



TGH

Making Integrations Simpler



Integration With AtomSphere Api in Boomi



INTEGRATION WITH ATOMSPHERE API IN BOOMI

In this blog, we will see how to create a package component and deploy it to an environment using AtomSphere API in Boomi.

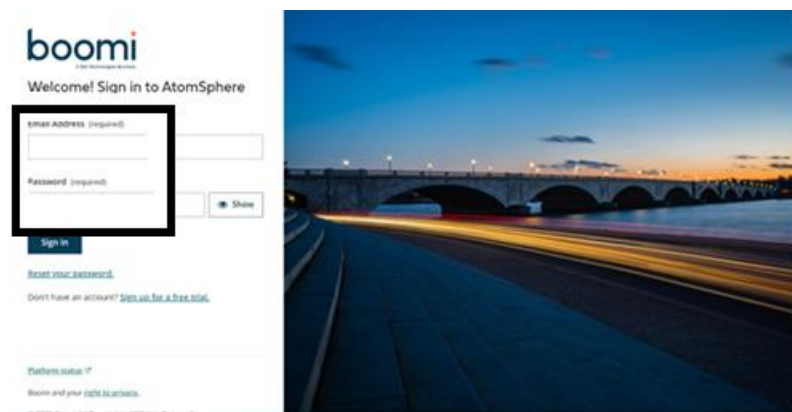
What is AtomSphere API?

It is an API in Boomi which automates the packaging, deployment of integration processes and other components. All AtomSphere API calls are authenticated by a user name or API token and an account ID that is associated with an active AtomSphere account. Deploying an AtomSphere API connector does not affect your license count.

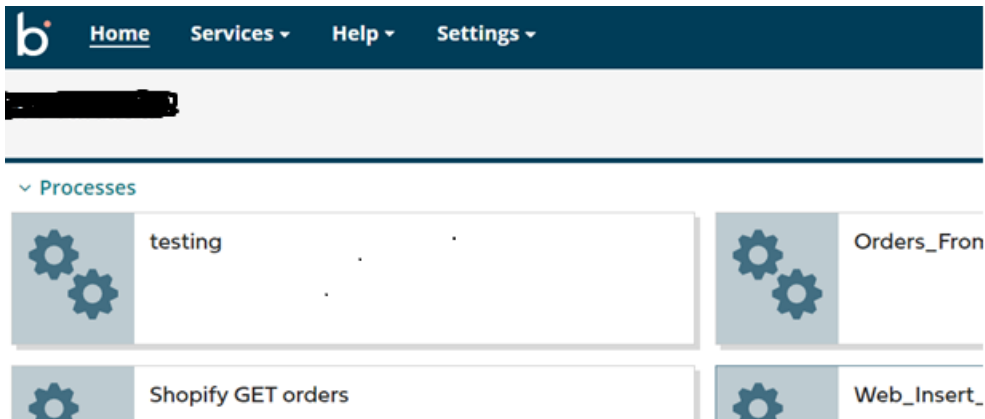
In this Use Case, we will create a packaged component and deploy the process called “Shopify GET orders” using the Atomsphere API connector.

Let us begin with the steps by integrating AtomSphere API in Boomi.

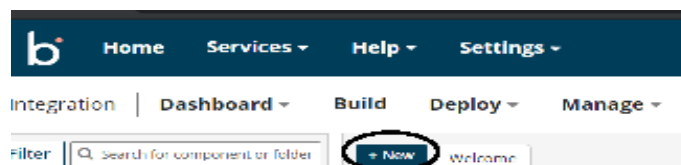
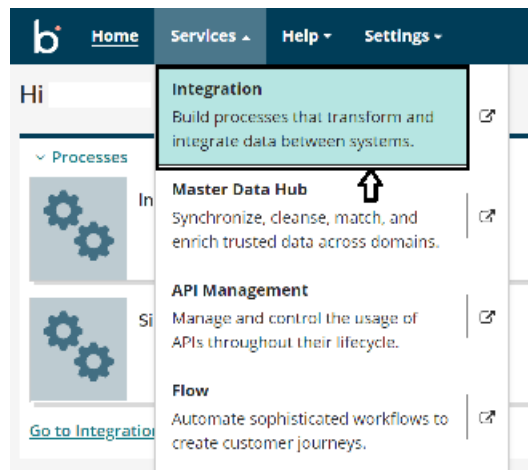
Step 1: Log on to the Boomi platform (<https://platform.boomi.com/>) with the required credentials i.e. Email Address and Password.



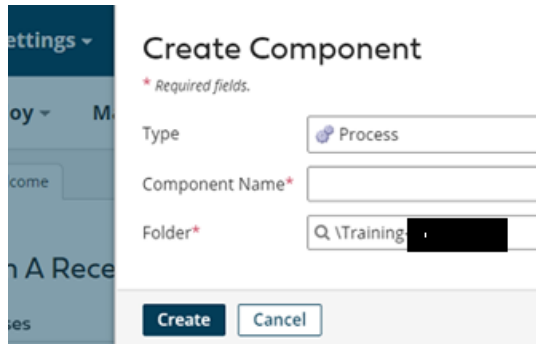
Step 2: Once logged into the Boomi platform, we will be able to view the Home page.



Step 3: Now, click on Services followed by Integration. We will see the Build page. Click on New.

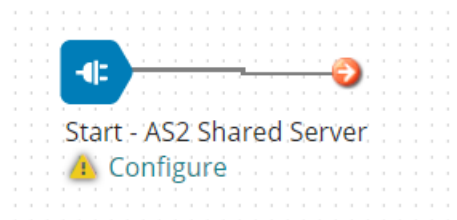


Step 4: Once, click on New, we will be able to see three fields i.e. Type, Component Name and Folder.

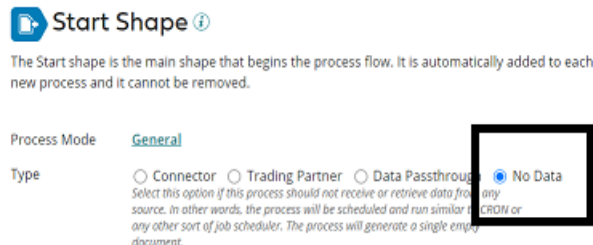


- Select Type as process as we are building a process. Component Name and Folder can be given based on your choice (i.e. which name to be given and where do we want to create the process). Click on Create.

Step 5: We see that the process gets created with a start shape which is configured with AS2 Shared Server by default.



Step 6: Select the start shape and choose No Data. Click ok.



Step 7: Drag and drop the Branch shape onto the process canvas. Number of branches will be 2. Branch 1 will create the package component and branch 2 will deploy the process.

Branch Shape ⓘ

The Branch shape is used when you have several actions that you want to execute in sequence. Each branch consists of a separate path that is executed in sequential order. A branch's path is executed to completion before executing the next branch.

Display Name

Number of Branches
Specify a number between 2 and 25.

Step 8: Drag and drop the AtomSphere connector onto the process canvas and configure it.

Connector Shape ⓘ


Connector shapes are used to get data into and send data out of a process. Most processes have one "get" connector and one or more "send" connectors. The Connector shape uses a combination of predefined connection and operation components to establish where and how to get or send data.


General Parameters

Display Name

Connector ⓘ AtomSphere API

Action GET

Connection ⓘ 

Operation ⓘ 


- Here, we choose Action as **CREATE** as we are creating the package component. Click + on connection.


General Parameters

Display Name

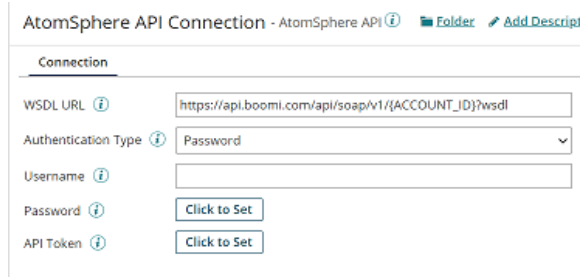
Connector ⓘ AtomSphere API

Action **CREATE**

Connection ⓘ 

Operation ⓘ 

- Name the connection and configure the details.



AtomSphere API Connection - AtomSphere API ⓘ Folder Add Description

Connection

WSDL URL ⓘ

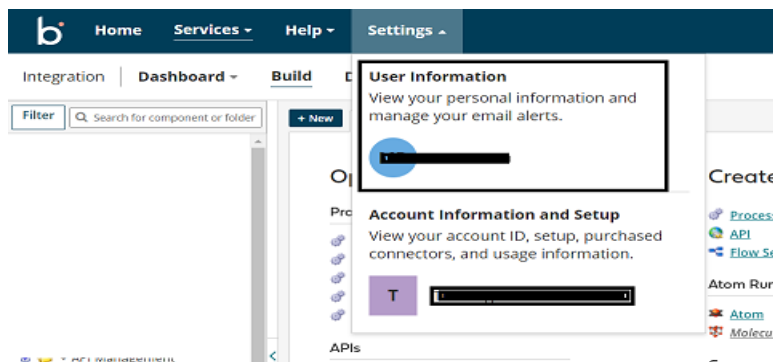
Authentication Type ⓘ Password

Username ⓘ

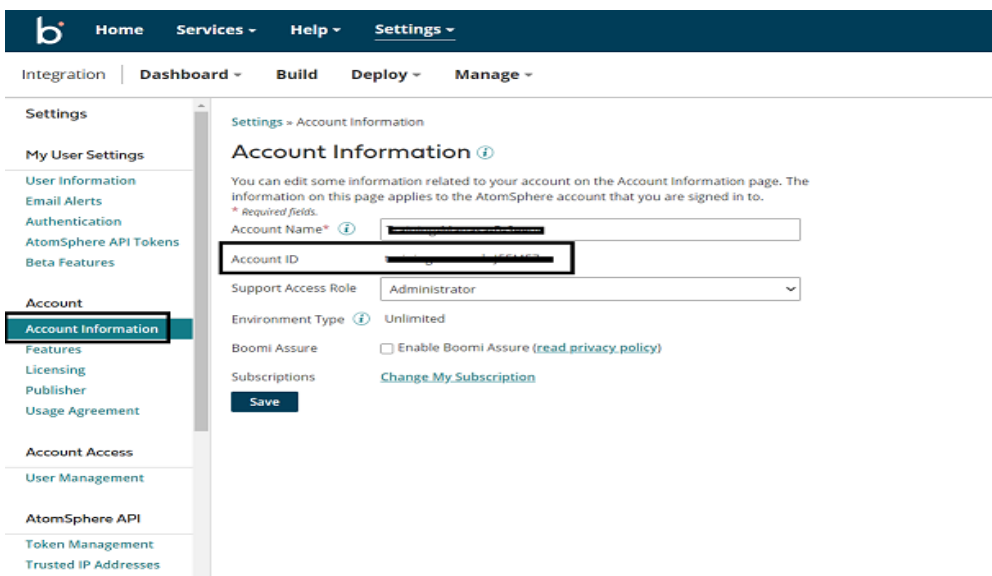
Password ⓘ

API Token ⓘ

- In the WSDL URL, give your account ID instead of ACCOUNT_ID.
- To get a Boomi Account ID, go to settings and click on User Information.



- Select Account Information on the left and We will find Account ID



Home Services Help Settings

Integration Dashboard Build Deploy Manage

Settings

My User Settings
User Information
Email Alerts
Authentication
AtomSphere API Tokens
Beta Features

Account
Account Information
Features
Licensing
Publisher
Usage Agreement

Account Access
User Management

AtomSphere API
Token Management
Trusted IP Addresses

Settings > Account Information

Account Information ⓘ

You can edit some information related to your account on the Account Information page. The information on this page applies to the AtomSphere account that you are signed in to.
* Required fields.

Account Name* ⓘ

Account ID

Support Access Role Administrator

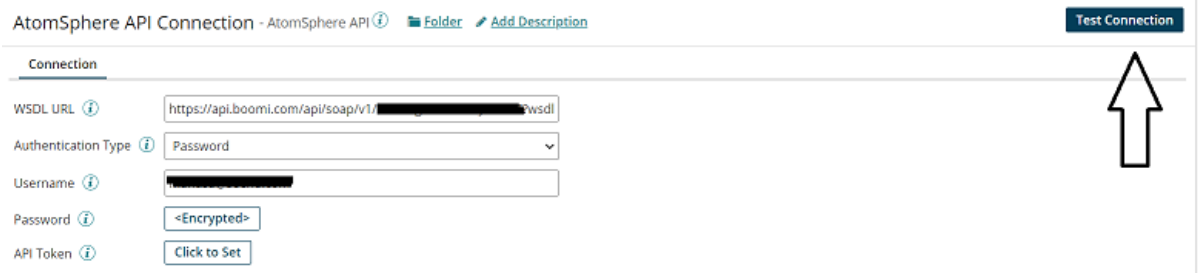
Environment Type ⓘ Unlimited

Boomi Assure Enable Boomi Assure (read privacy policy)

Subscriptions [Change My Subscription](#)

- Authentication Type will be Password.
- Give the name and password of your Boomi account.

Step 9: Now, test the connection.



AtomSphere API Connection - AtomSphere API ⓘ Folder Add Description

Test Connection

Connection

WSDL URL ⓘ

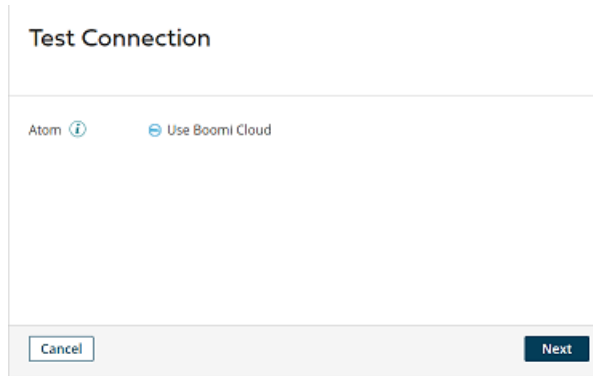
Authentication Type ⓘ Password

Username ⓘ

Password ⓘ

API Token ⓘ

- Click on Next.

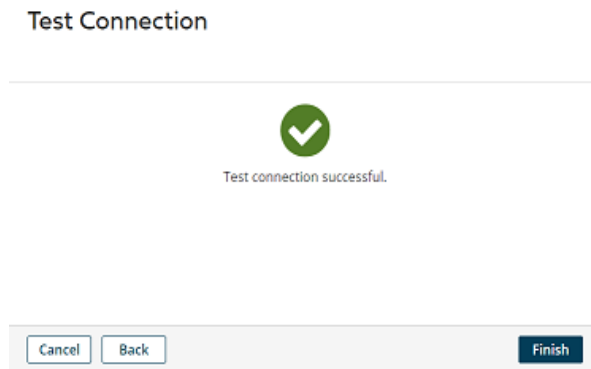


Test Connection


Atom ⓘ Use Boomi Cloud

Cancel Next

- We see that the Test connection is successful. Click Finish, Save and Close.



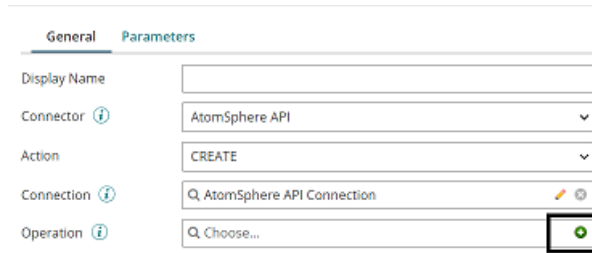
Test Connection



Test connection successful.

Cancel Back Finish

Step 10: Click + on operation and name it.




General Parameters

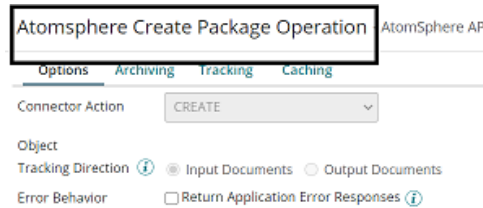
Display Name

Connector ⓘ AtomSphere API

Action CREATE

Connection ⓘ AtomSphere API Connection

Operation ⓘ 



AtomSphere Create Package Operation AtomSphere AP

Options Archiving Tracking Caching

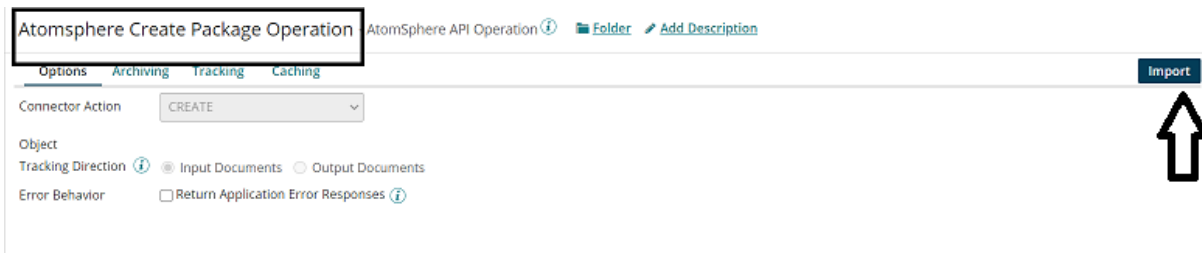
Connector Action CREATE

Object

Tracking Direction ⓘ Input Documents Output Documents

Error Behavior Return Application Error Responses ⓘ

- Click on Import.



AtomSphere Create Package Operation AtomSphere API Operation ⓘ Folder Add Description

Options Archiving Tracking Caching **Import**

Connector Action CREATE

Object

Tracking Direction ⓘ Input Documents Output Documents

Error Behavior Return Application Error Responses ⓘ

- Choose the connection which we have configured in AtomSphere API Connection. Click next.



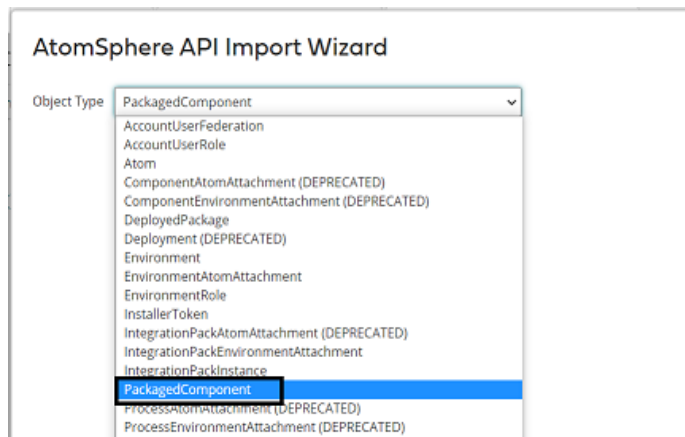
AtomSphere API Import Wizard

* Required fields

Connection*

Filter ⓘ

- Select Object type as package component. Click next.



- We see that the profiles get imported. Click finish, save and close.

AtomSphere API Import Wizard

Operation Loaded

Object Name PackagedComponent
Request Profile AtomSphere API PackagedComponent CREATE Request
Response Profile AtomSphere API PackagedComponent CREATE Response

AtomSphere Create Package Operation - AtomSphere API Operation ⓘ

Options Archiving Tracking Caching

Connector Action CREATE

Object	PackagedComponent
Request Profile	AtomSphere API PackagedComponent CREATE Request ✎ ⓧ
Response Profile	AtomSphere API PackagedComponent CREATE Response ✎ ⓧ

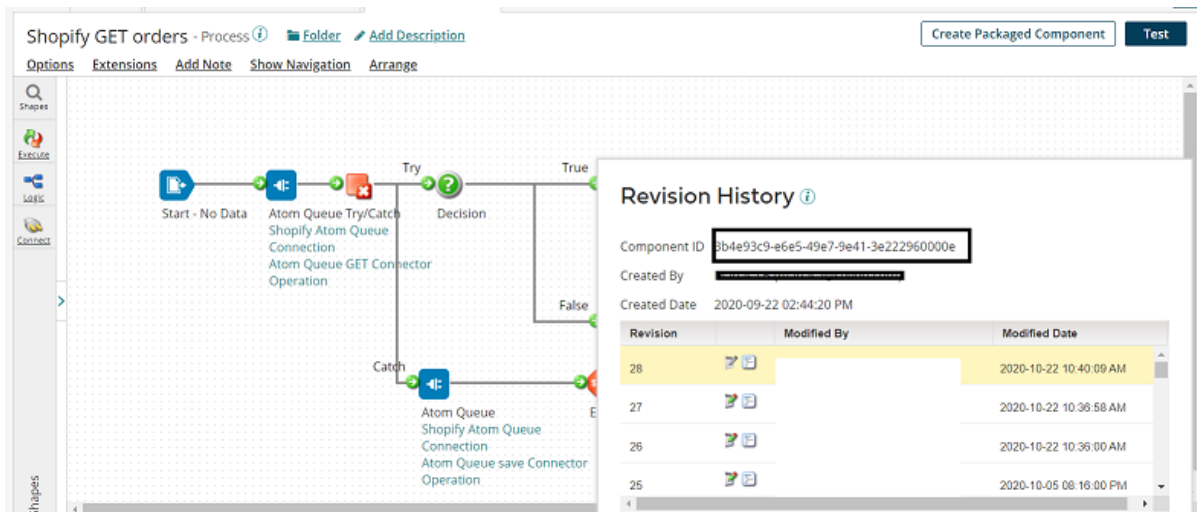
Tracking Direction ⓘ Input Documents Output Documents

Error Behavior Return Application Error Responses ⓘ

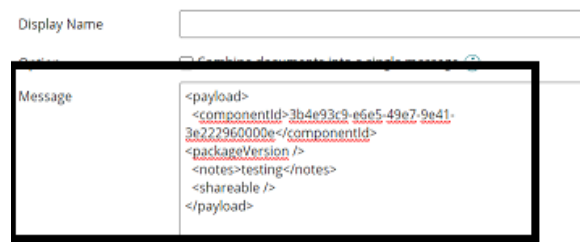
Step 11: Drag and drop message shape to add static XML payload containing component ID, notes, packageVersion and shareable.

In component id, we will add the id of the process for which we are going to create the package component.

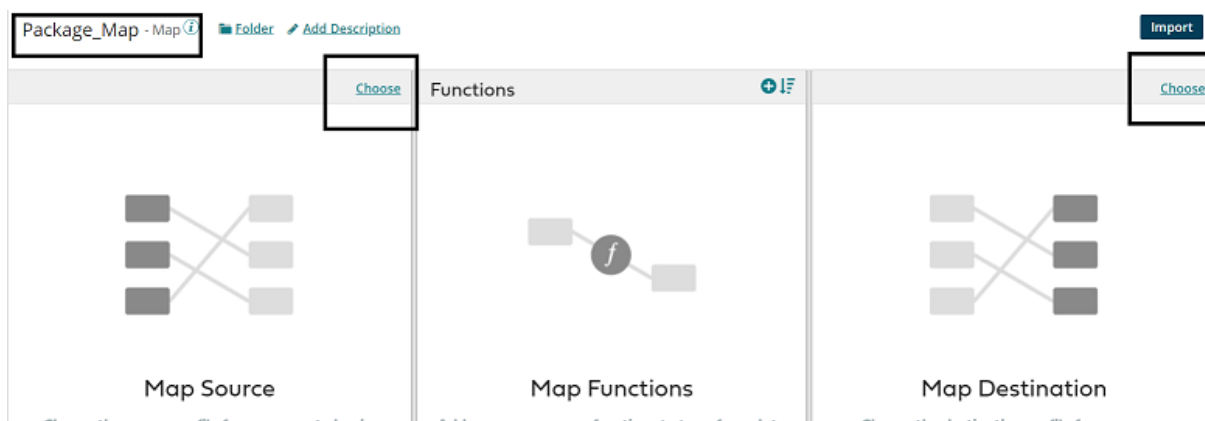
To get the Component ID, select the process for which the packaged component has to be created. Click on Revision History which is at the right corner as shown below. Copy the ID and add it to the component ID of the XML payload in the message



- Navigate to the process which you are building. Click on the message shape and add the data in the message section. Click ok.

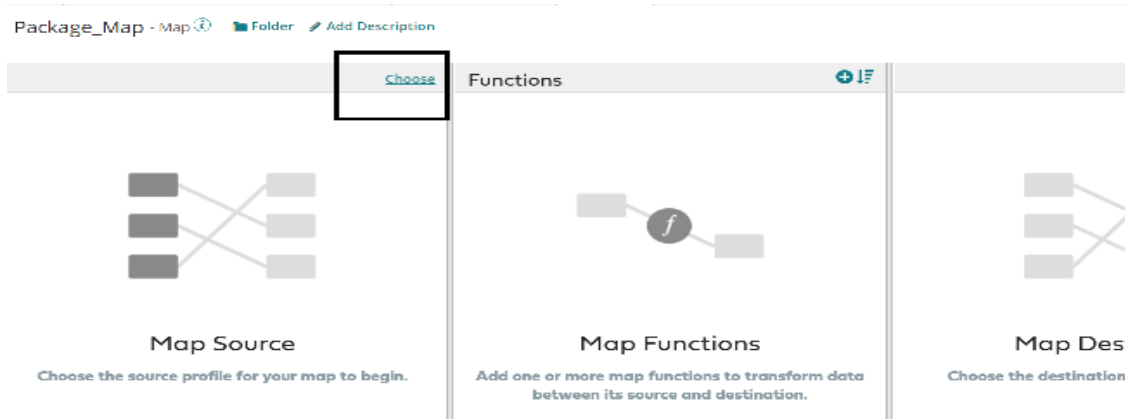


Step 12: Drag and drop the map shape onto the process canvas and name it.

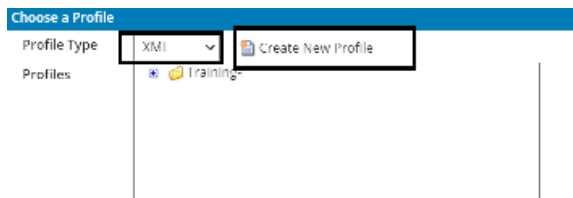


- Add profiles on both sides. The left side will be the XML profile which we send as a payload in message shape. The right side will be the Packaged Object request profile

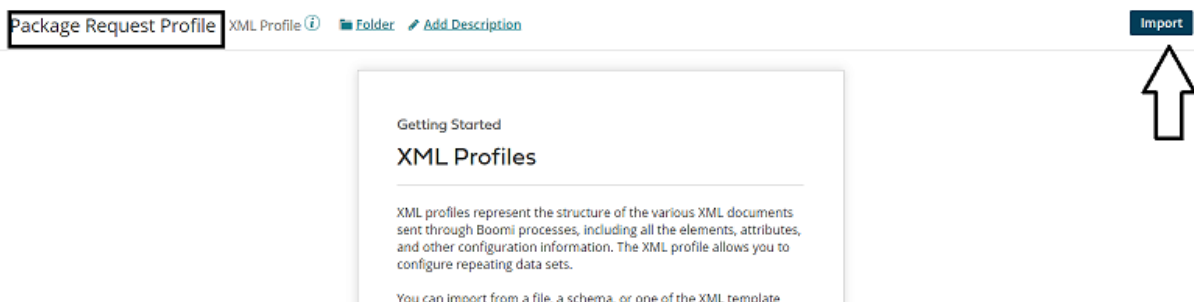
which we have imported in Atomsphere API. Click on Choose on the left side and add the XML profile.



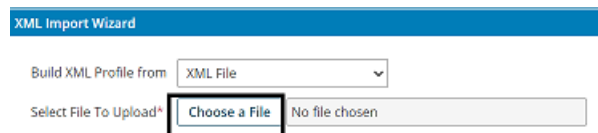
- Choose profile type as XML and click on Create a new profile.



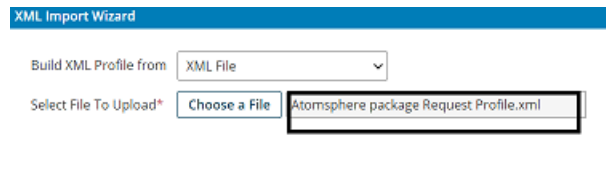
- Name it and click on import as shown.



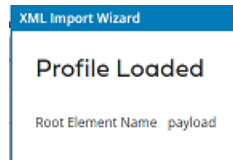
- Choose a file from the location where we have saved it.



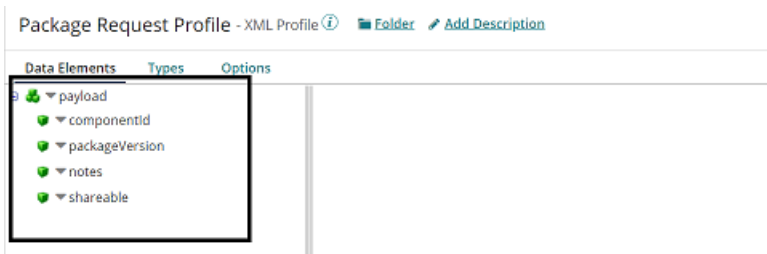
- Once, we choose the file, we can see the file name.



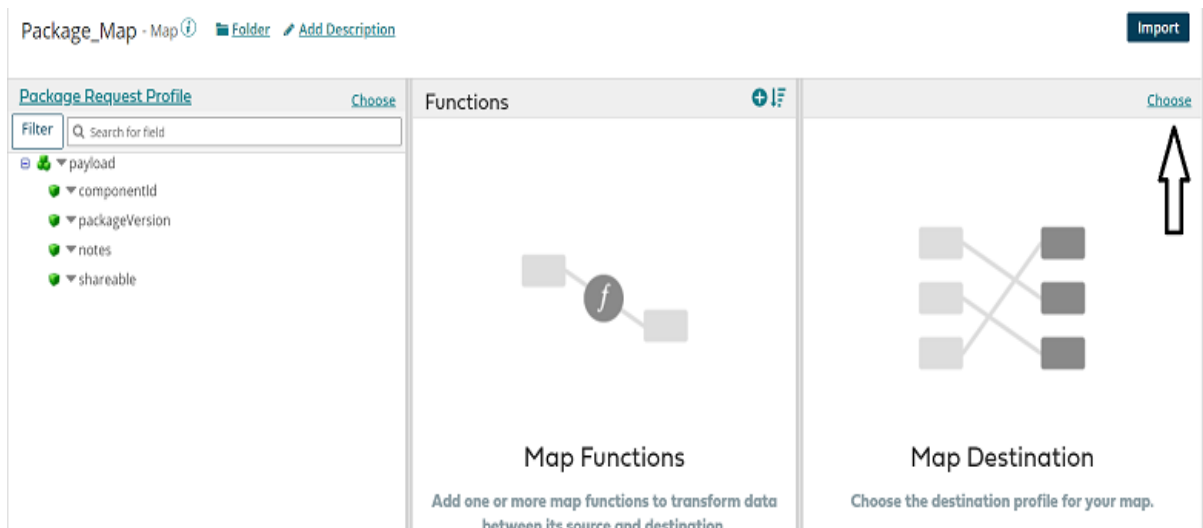
- Click next. We see the profile has been loaded. Click on finish.



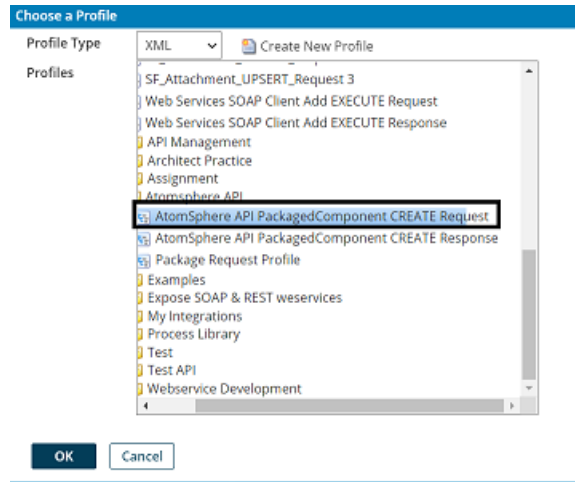
- The Request Profile looks like,



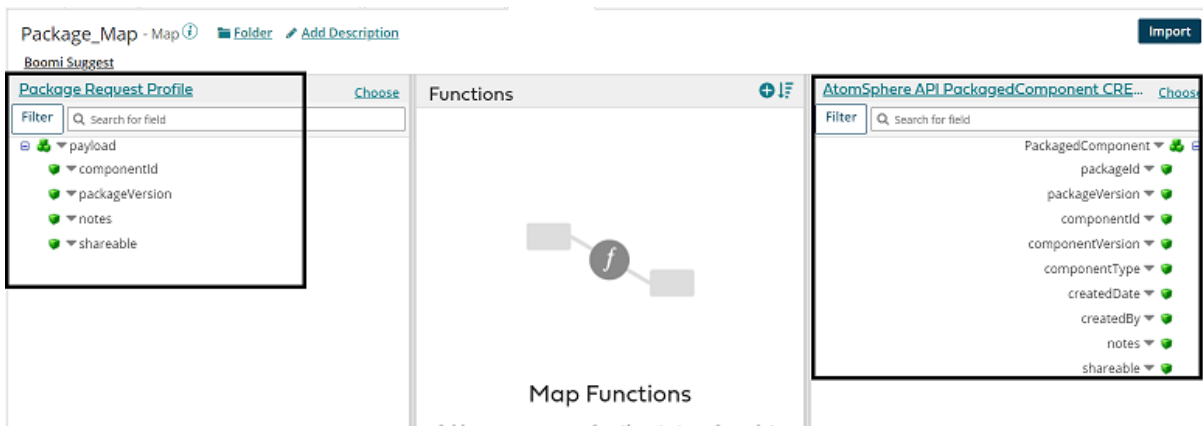
- Click save and close. Click on Choose on the right side of the map to add the destination profile.



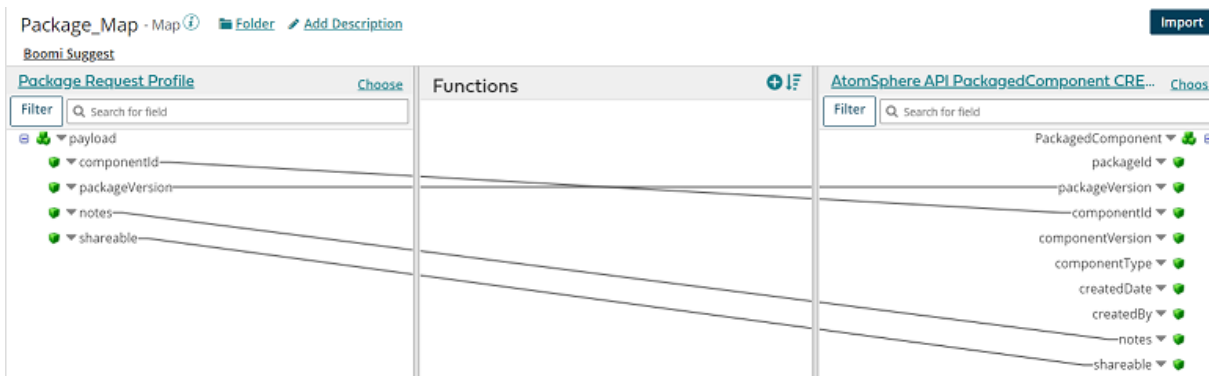
- Select the profile type as XML and Choose the imported Packaged Component Response Profile. Click ok.



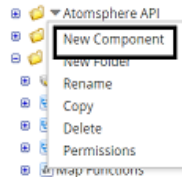
- We see that both the profiles have been imported.



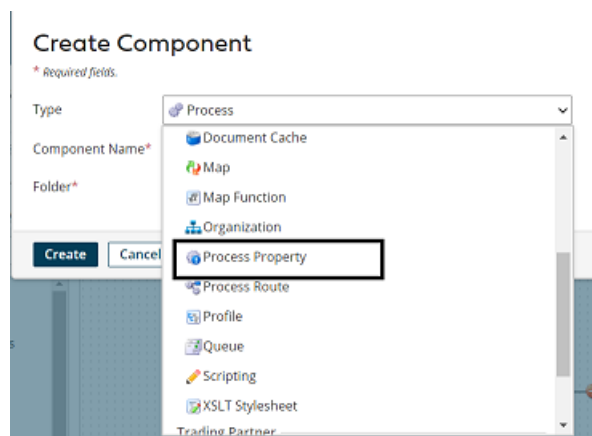
Step 13: Provide one-to-one mapping. Click Save and close.



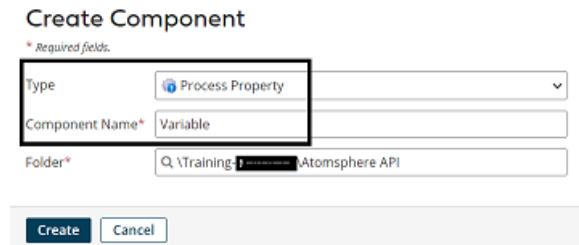
Step 14: Now, we will create a Process Property to store the package ID which comes as a response. Click the arrow beside the folder in which we have created the Process and select New Component.



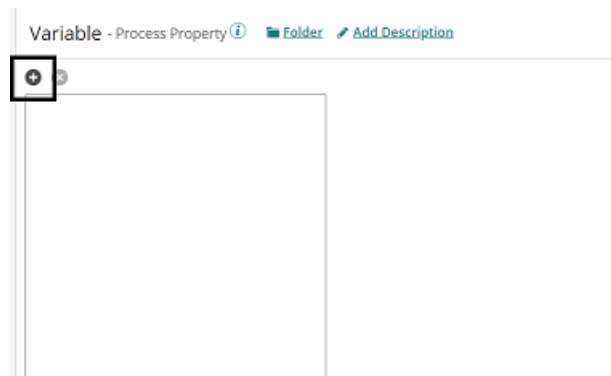
- Choose process property.



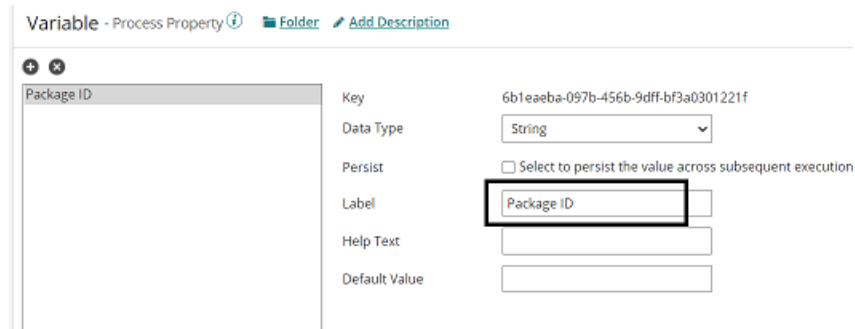
- Add the Component name as a Variable. Click on Create.



- Select + to add the property name as shown below.



- Add the label as Package Id. Click save and close.



Variable - Process Property ⓘ Folder Add Description

Package ID

Key 6b1eaeba-097b-456b-9dff-bf3a0301221f

Data Type String

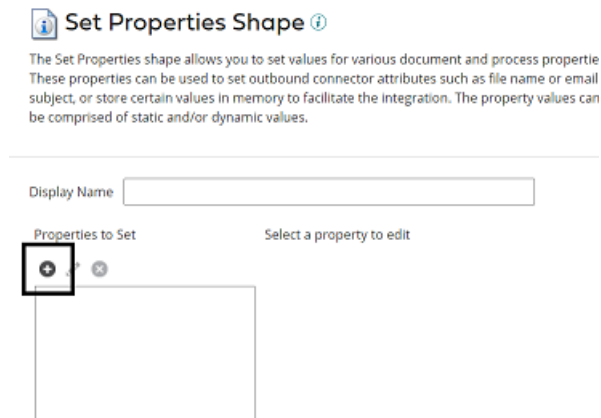
Persist Select to persist the value across subsequent execution

Label **Package ID**

Help Text

Default Value

Step 15: Drag and drop set properties shape onto process canvas. Select + to add the property name.



Set Properties Shape ⓘ

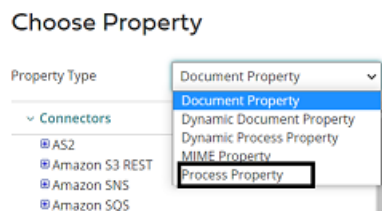
The Set Properties shape allows you to set values for various document and process properties. These properties can be used to set outbound connector attributes such as file name or email subject, or store certain values in memory to facilitate the integration. The property values can be comprised of static and/or dynamic values.

Display Name

Properties to Set Select a property to edit

+

- Select process property. Choose the property name as a variable and process the property name as package ID.



Choose Property

Property Type

Connectors

- AS2
- Amazon S3 REST
- Amazon SNS
- Amazon SQS

Document Property

Dynamic Document Property

Dynamic Process Property

MIME Property

Process Property



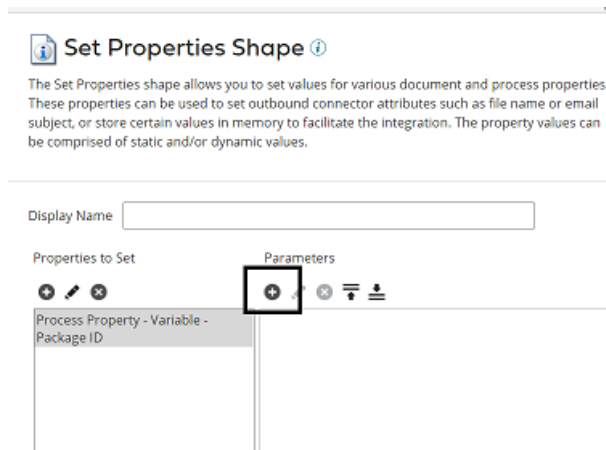
Choose Property

Property Type Process Property

Process Property* Variable

Process Property Name Package ID

- Click + on parameters to assign the value as shown.



Step 16: Select type as profile element. Choose Profile type as XML, Profile as the Packaged Component response profile which we have imported while configuring AtomSphere API and elements as package ID.

Parameter Value

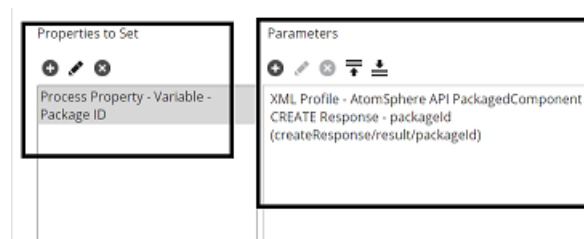
Type:

Profile Type:

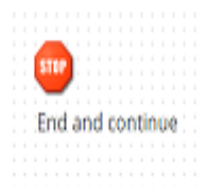
Profile:

Element:

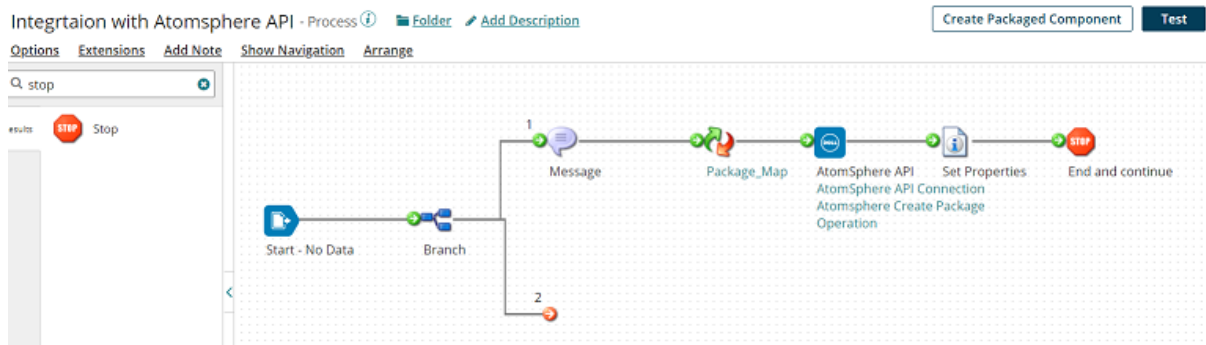
- After configuring the set properties shape, it looks like.



Step 17: Now append the top shape at the end.

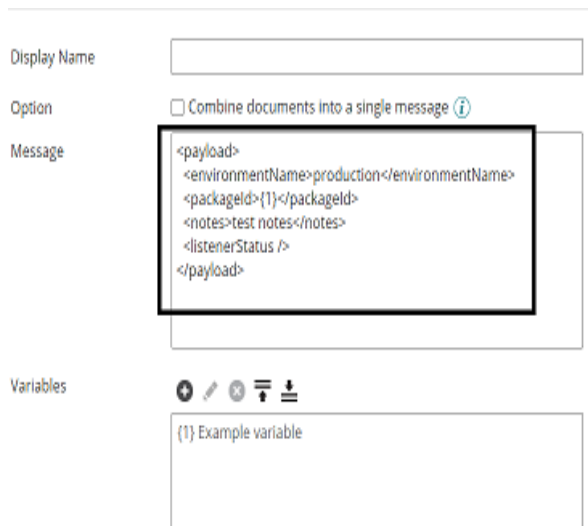


Step 18: Arrange all the shapes to branch 1 as shown in the screenshot.



Step 19: Drag and drop message shape to add static XML payload containing Environment Name, PackageId, Notes and listener status.

- Here, the environmental value will be Production.
- Package ID will be {1} which means we are getting the value from the process property to which Package ID has been assigned in Branch 1.

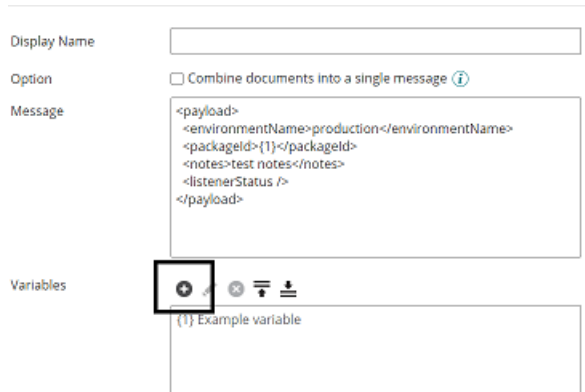


The screenshot shows a configuration dialog for a message shape. It has fields for 'Display Name', 'Option' (with a checkbox 'Combine documents into a single message'), and 'Message'. The 'Message' field contains the following XML payload:

```
<payload>
  <environmentName>production</environmentName>
  <packageId>{1}</packageId>
  <notes>test notes</notes>
  <listenerStatus />
</payload>
```

Below the 'Message' field is a 'Variables' section with a list containing '{1} Example variable'.

- To assign the value, click on variables.



Display Name:

Option: Combine documents into a single message ⓘ

Message:

```
<payload>
<environmentName>production</environmentName>
<packageId>{1}</packageId>
<notes>test notes</notes>
<listenerStatus />
</payload>
```

Variables: + ✖ 📄 📥

{1} Example variable

- Select Type as process property. Choose process property as Variable and process property name as package ID. Click ok.

Parameter Value

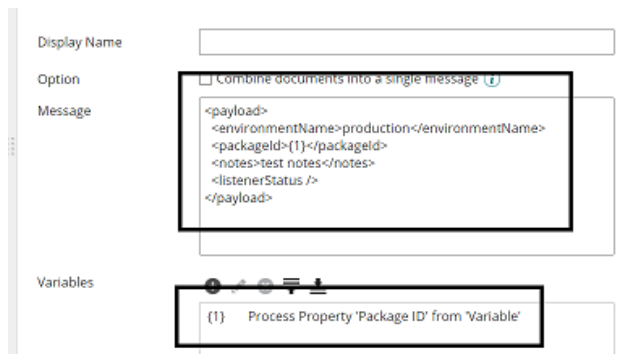


Type: Process Property

Process Property*: Q Variable

Process Property Name: Package ID

- After configuring the message shape, it looks like



Display Name:

Option: Combine documents into a single message ⓘ

Message:

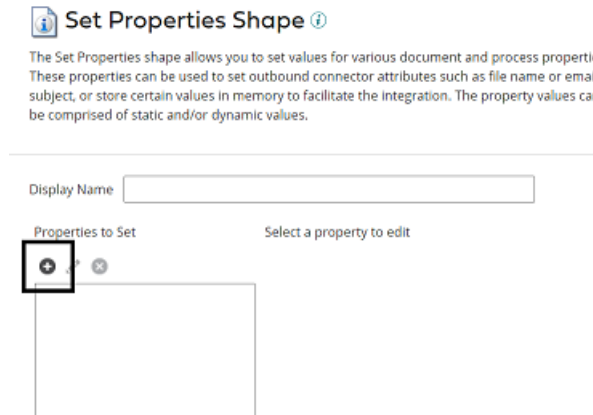
```
<payload>
<environmentName>production</environmentName>
<packageId>{1}</packageId>
<notes>test notes</notes>
<listenerStatus />
</payload>
```

Variables: + ✖ 📄 📥

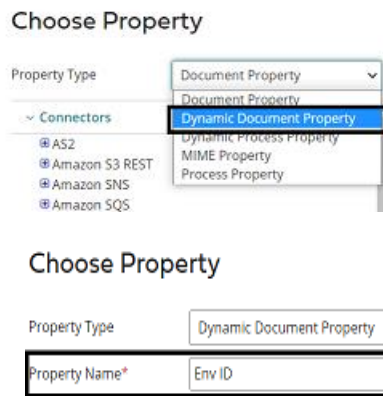
{1} Process Property 'Package ID' from 'Variable'

- Click ok.

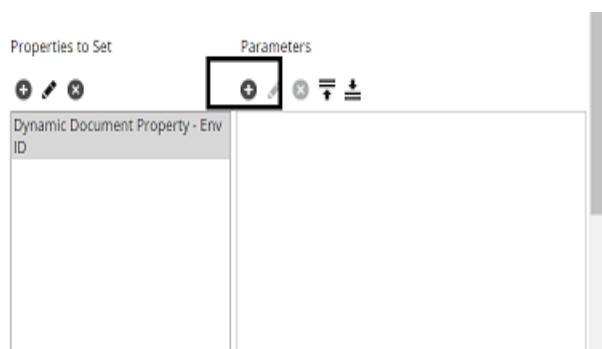
Step 20: Drag and drop set properties shape onto the process canvas to set the environment ID in Dynamic Document Property. Select + to set the property name as below.



- Choose Document property and give the property name as Env ID. Click ok.

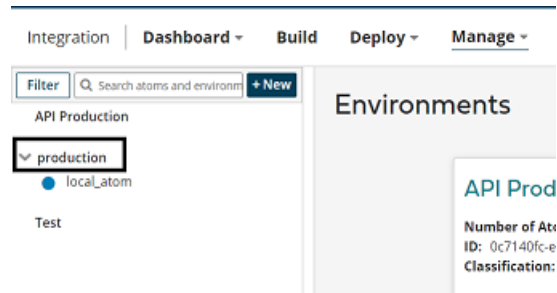


Step 21: Select + on parameters as shown below.

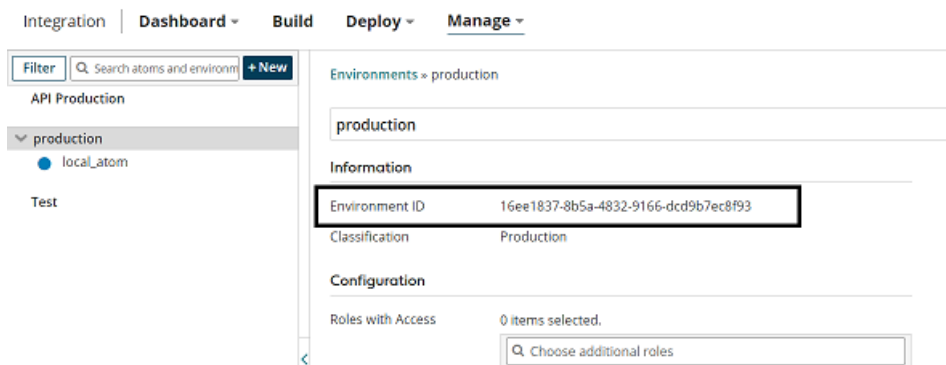


Step 22: Give an ID of the environment to which you want to deploy the process.

- To get the ID, go to Manage > Atom Management > Click on the environment.



- We can see the environment id on the right side.



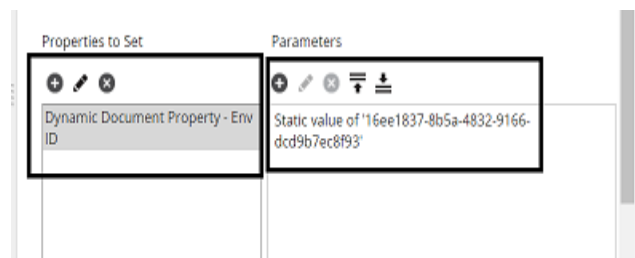
- Copy the ID and add in the static value of the document property. Click ok.

Parameter Value

Type:

Static Value:

- After configuring the set properties shape, it looks like



Step 23: Drag and drop the AtomSphere connector onto the process canvas and configure it.

Connector Shape ⓘ

Connector shapes are used to get data into and send data out of a process. Most processes have one "get" connector and one or more "send" connectors. The Connector shape uses a combination of predefined connection and operation components to establish where and how to get or send data.

General Parameters

Display Name

Connector ⓘ AtomSphere API ▼

Action GET ▼

Connection ⓘ Ⓞ

Operation ⓘ Ⓞ

- Configure the same Action and Connection as mentioned in Step 8.

Step 24: Click + on operation and name it.

General Parameters

Display Name

Connector ⓘ AtomSphere API ▼

Action CREATE ▼

Connection ⓘ Ⓞ

Operation ⓘ Ⓞ

Deployment Operation AtomSphere API Operation ⓘ

Options Archiving Tracking Caching

Connector Action CREATE ▼

Object

Tracking Direction ⓘ Input Documents Output Documents

Error Behavior Return Application Error Responses ⓘ

- Click on Import.

Deployment Operation AtomSphere API Operation ⓘ Folder Add Description


Options Archiving Tracking Caching **Import**

Connector Action CREATE ▼

Object

Tracking Direction ⓘ Input Documents Output Documents

Error Behavior Return Application Error Responses ⓘ



- Choose the connection which we have configured in Connection. Click next.

AtomSphere API Import Wizard

* Required fields.

Connection*

Filter

- Select the Object type as Deployed Package. Click next.

AtomSphere API Import Wizard

Object Type

- AccountUserFederation
- AccountUserRole
- Atom
- ComponentAtomAttachment (DEPRECATED)
- ComponentEnvironmentAttachment (DEPRECATED)
- DeployedPackage**
- Deployment (DEPRECATED)
- Environment
- EnvironmentAtomAttachment
- EnvironmentRole
- InstallerToken

- We see that profiles get imported. Click finish, save and close.

AtomSphere API Import Wizard

Operation Loaded

Object Name	DeployedPackage
Request Profile	AtomSphere API DeployedPackage CREATE Request
Response Profile	AtomSphere API DeployedPackage CREATE Response

Deployment Operation AtomSphere API Operation

Options Archiving Tracking Caching

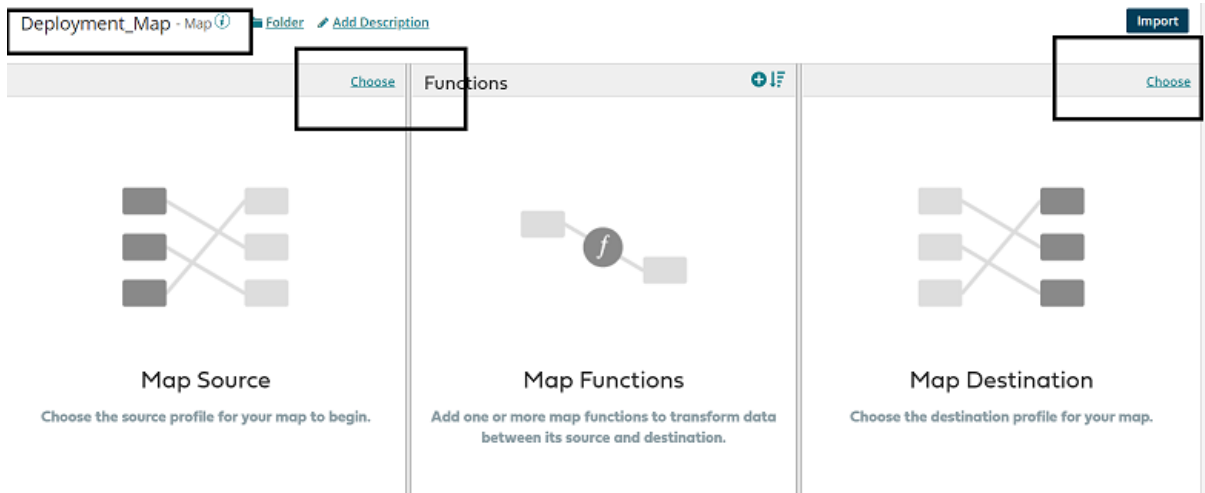
Connector Action

Object	DeployedPackage
Request Profile	<input type="text" value="AtomSphere API DeployedPackage CREATE Request"/>
Response Profile	<input type="text" value="AtomSphere API DeployedPackage CREATE Response"/>

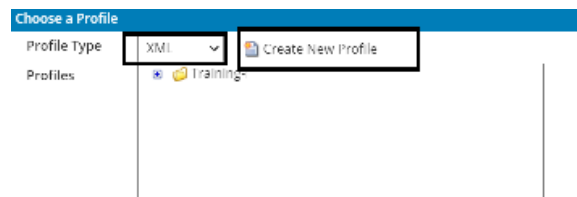
Tracking Direction Input Documents Output Documents

Error Behavior Return Application Error Responses

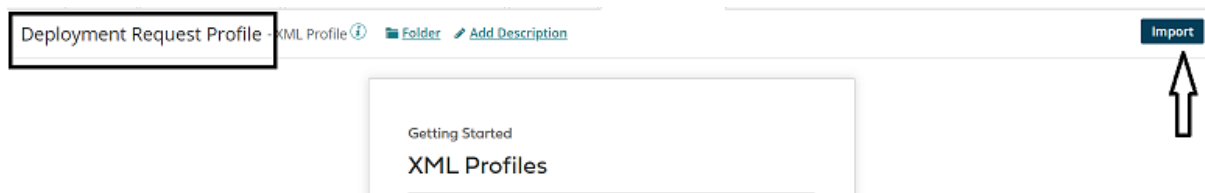
Step 25: Drag and drop Map Shape onto the process canvas and name it. The left side will be the XML profile which we send as a payload in message shape. The right side will be the request profile which we have imported as a Deployed Object Request Profile in atomsphere API. Click on to left side and add the XML profile.



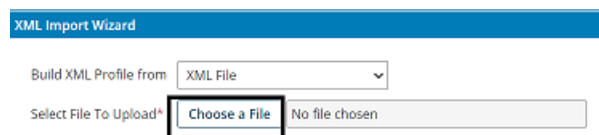
- Choose profile type as XML and click on Create a new profile.



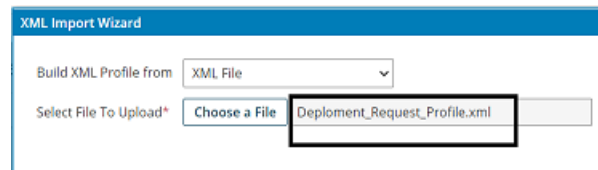
- Name it and click on import.



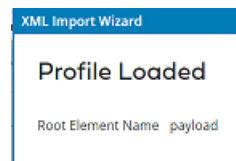
- Choose a file from the location where we have saved it.



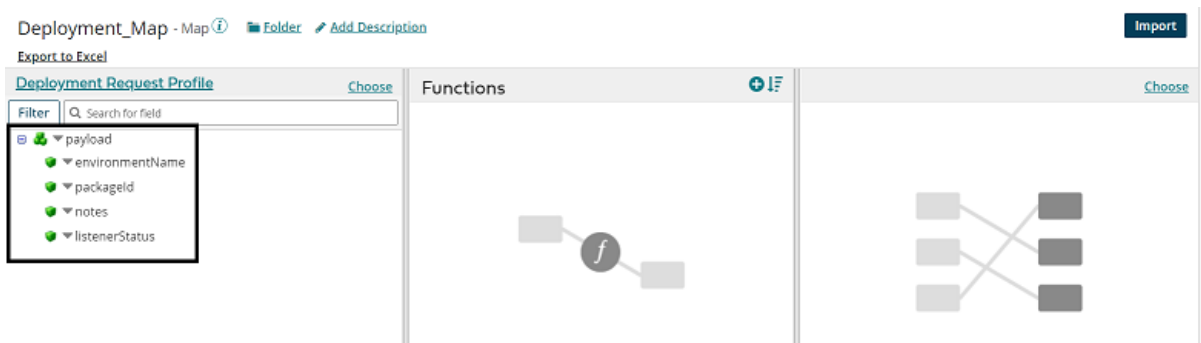
- Once, we choose the file, we can see the file name.



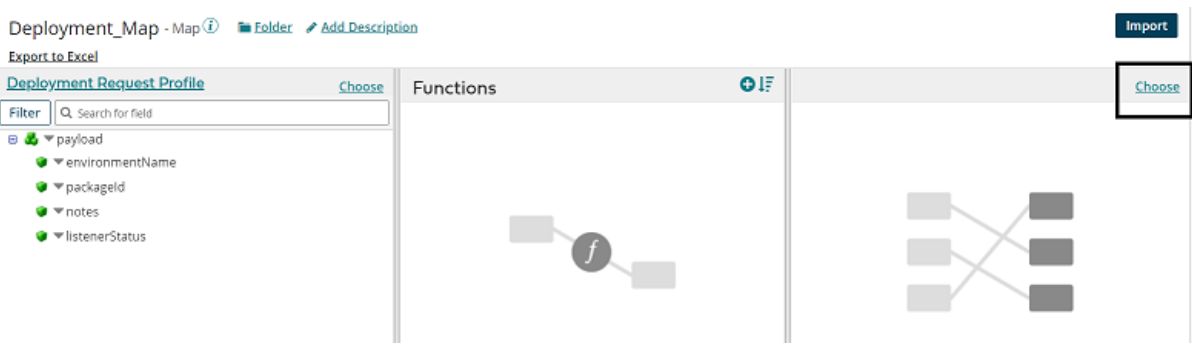
- Click next. We see the profile has been loaded. Click finish.



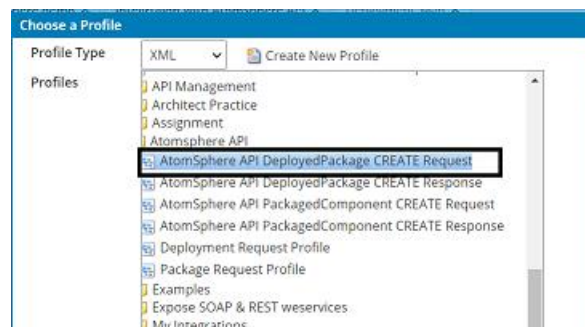
- The request profile looks like this,



Step 27: Click save and close. Click on the right side of the map to add the destination profile.



- Select the profile type as XML and select the imported profile. Click ok

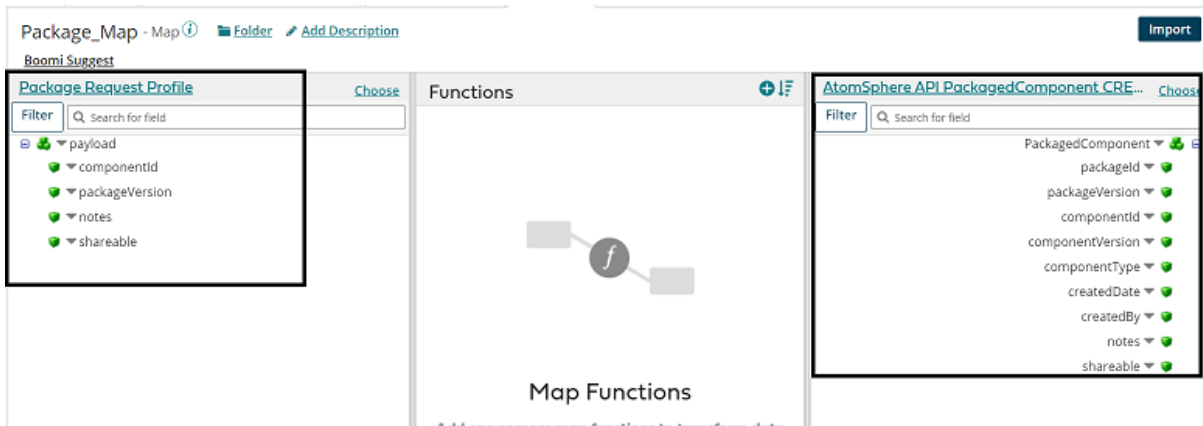


©TGH Software Solutions Pvt. Ltd.

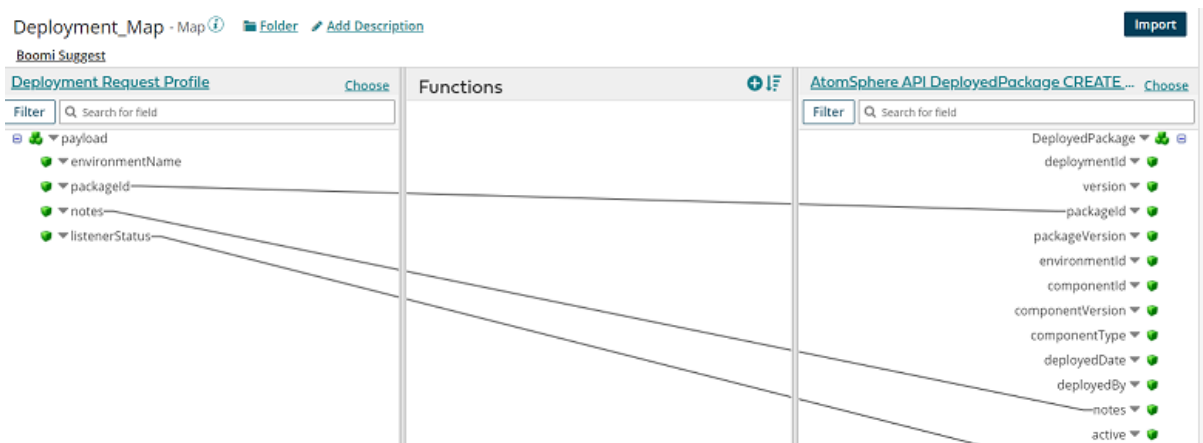
No part of this document may be copied, reproduced, republished, uploaded, posted, publicly displayed, encoded, translated, transmitted or distributed in any way to any other computer, server, website or other medium for publication or distribution, without TGH's prior written consent



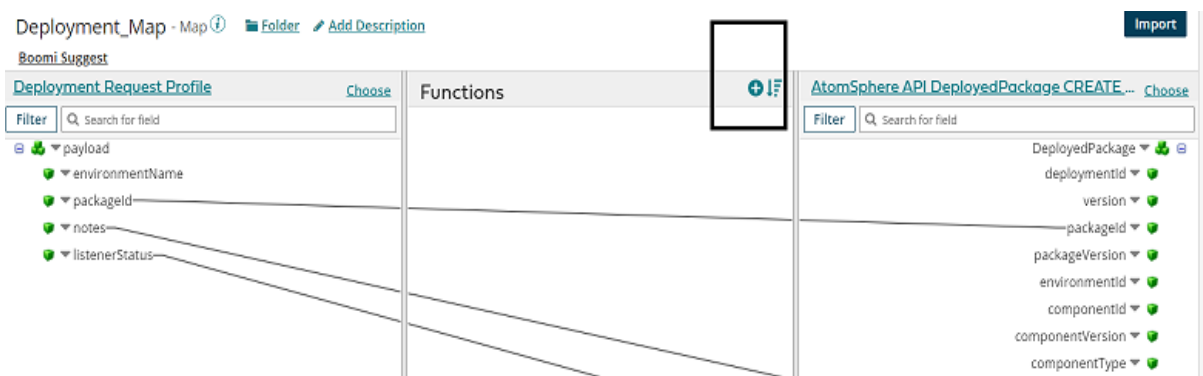
- We see that both the profiles have been imported.



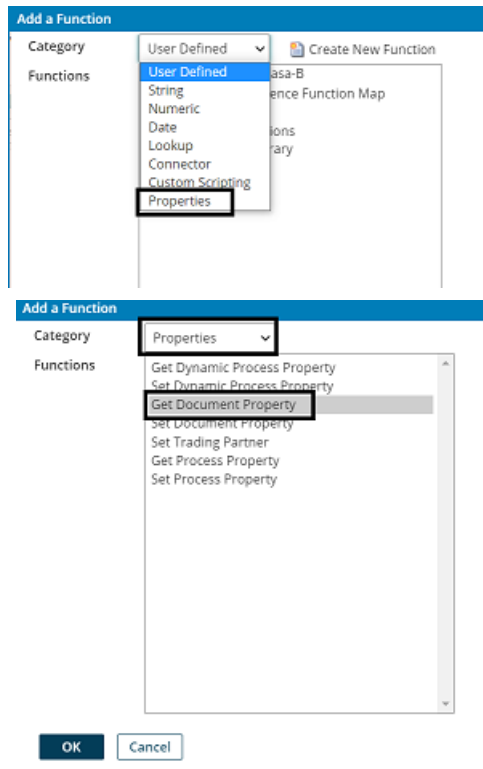
Step 28: Provide one-to-one mapping for notes, listener and package ID.



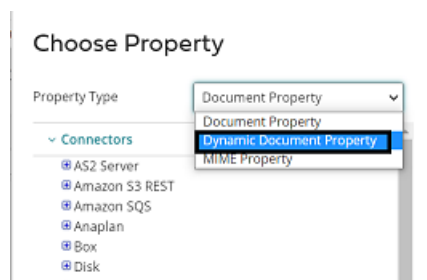
Step 29: To set the environment ID, get the ID from the document property which we have configured in the set properties shape. Click on functions.



- We can see a few categories in Function i.e. String, Numeric, Date, Look up and so on. Here, we will select Category as properties and function as Get Document Property. Click ok.



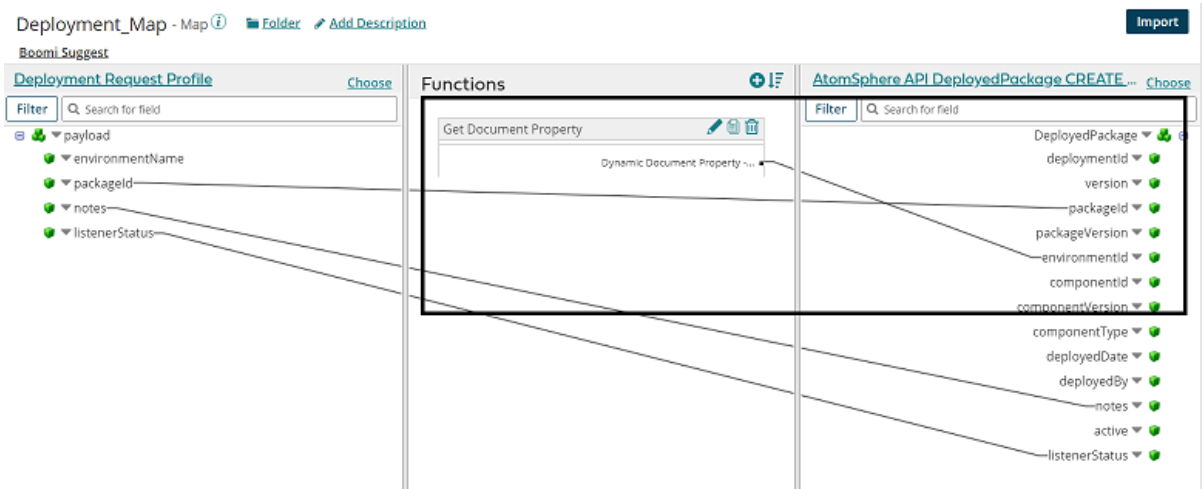
- Choose Dynamic Document Property from the drop-down as shown.



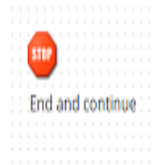
- Here, the Property Type would be Dynamic Document Property and the Property Name would be Env ID. Click ok.



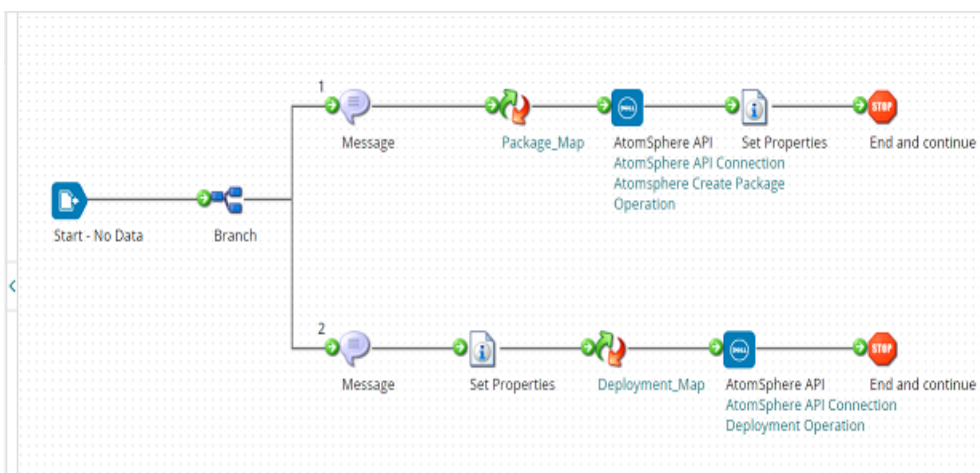
- We see that we can retrieve dynamic document Properties in the map. Now, set the property environment id as shown in the screenshot. Click save and close.



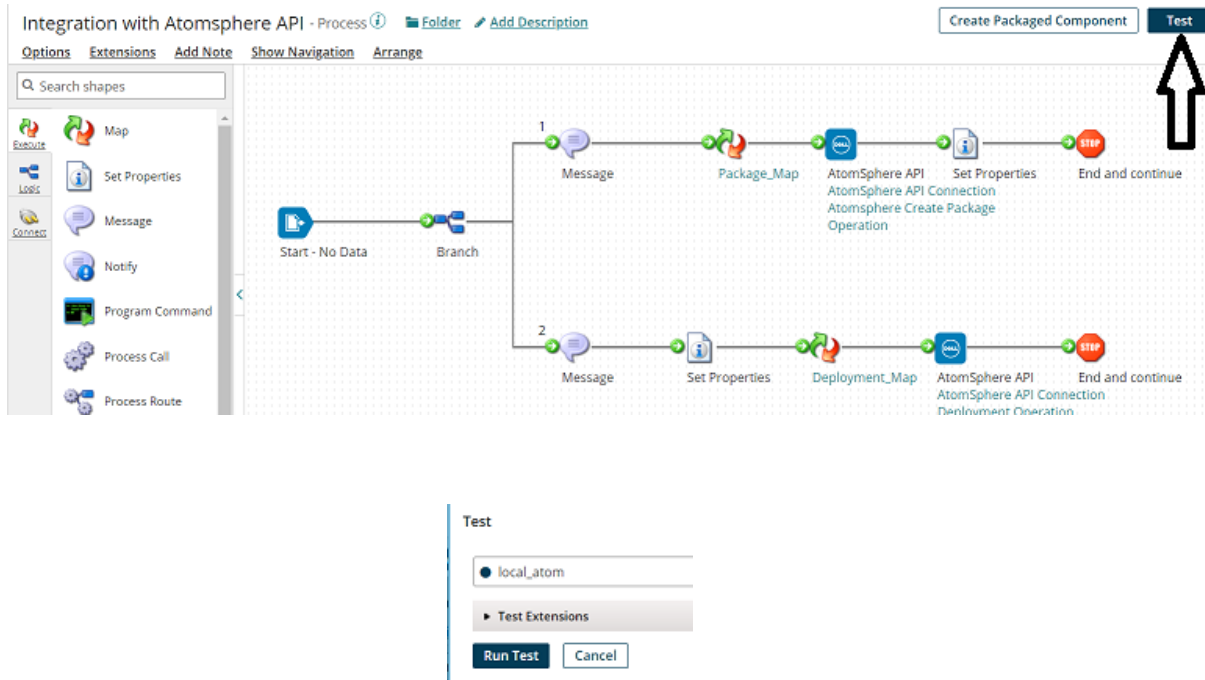
Step 30: Now append the stop shape at the end.



Step 31: Arrange all shapes in the order and complete the process looks like,



Step 32: Run the test by clicking on Run Test and configuring the local atom.



Step 33: We see that the process has been executed.



Step 34: Select the stop shape in branch 1 to check package component details.

← Process: Integration with Atomsphere API ▾

Documents		Test Results	
#		Logs	Shape Source Data
1	✓	View Source	Size (kB)
			0.67

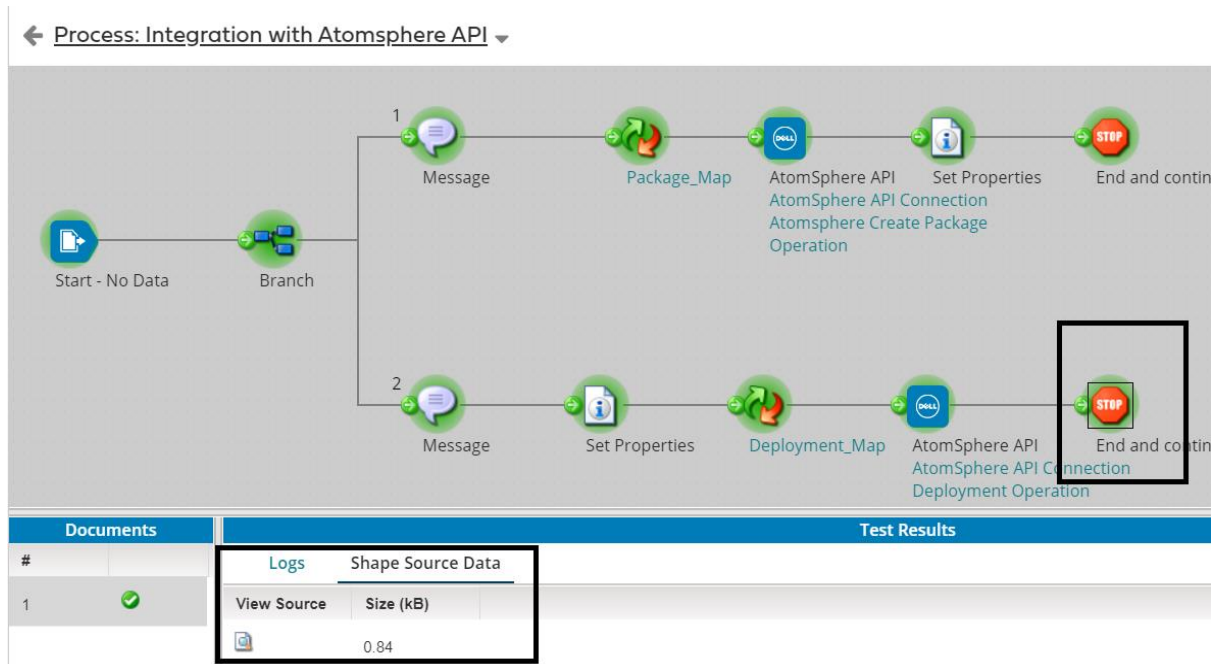
Document Viewer

```

1 <createResponse>
2 <result>
3 <packageId>e7171fdb-9218-415b-a29f-7f2aab2c0ae2</packageId>
4 <packageVersion>3.0</packageVersion>
5 <componentId>3b4e93c9-e6e5-49e7-9e41-3e222960000e</componentId>
6 <componentVersion>28</componentVersion>
7 <componentType>process</componentType>
8 <createdDate>2020-11-07T12:23:55Z</createdDate>
9 <createdBy>[REDACTED]</createdBy>
10 <notes>testing</notes>
11 <shareable>>false</shareable>
12 </result>
13 </createResponse>

```

Step 35: Select the stop shape in branch 2 to see the deployment details.

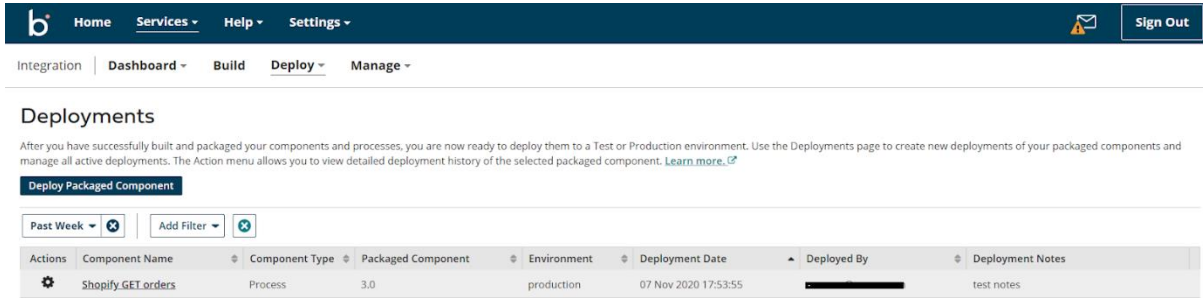


Document Viewer

```

1 <createResponse>
2 <result>
3 <deploymentId>84b21bac-6eb5-40df-a408-3156c44fdaca</deploymentId>
4 <version>2</version>
5 <packageId>e7171fdb-9218-415b-a29f-7f2aab2c0ae2</packageId>
6 <packageVersion>3.0</packageVersion>
7 <environmentId>16ee1837-8b5a-4832-9166-dcd9b7ec8f93</environmentId>
8 <componentId>3b4e93c9-e6e5-49e7-9e41-3e222960000e</componentId>
9 <componentVersion>28</componentVersion>
10 <componentType>process</componentType>
11 <deployedDate>2020-11-07T12:23:55Z</deployedDate>
12 <deployedBy>[REDACTED]</deployedBy>
13 <notes>test notes</notes>
14 <active>true</active>
15 </result>
16 </createResponse>
  
```

Step 36: If we go to Deploy > Deployments, we see that the process “Shopify get Orders “is deployed using AtomSphere API in Boomi.



The screenshot shows the Boomi user interface. At the top, there is a navigation bar with 'Home', 'Services', 'Help', and 'Settings'. Below this, a breadcrumb trail shows 'Integration' > 'Dashboard' > 'Build' > 'Deploy' > 'Manage'. The main heading is 'Deployments'. Below the heading, there is a sub-heading 'Deployments' and a paragraph of text: 'After you have successfully built and packaged your components and processes, you are now ready to deploy them to a Test or Production environment. Use the Deployments page to create new deployments of your packaged components and manage all active deployments. The Action menu allows you to view detailed deployment history of the selected packaged component. [Learn more](#).' Below this text is a button labeled 'Deploy Packaged Component'. Underneath the button, there are two filters: 'Past Week' and 'Add Filter'. The main content is a table with the following columns: 'Actions', 'Component Name', 'Component Type', 'Packaged Component', 'Environment', 'Deployment Date', 'Deployed By', and 'Deployment Notes'. The table contains one row with the following data: 'Shopify.GET_orders', 'Process', '3.0', 'production', '07 Nov 2020 17:53:55', a redacted name, and 'test notes'.

Actions	Component Name	Component Type	Packaged Component	Environment	Deployment Date	Deployed By	Deployment Notes
	Shopify.GET_orders	Process	3.0	production	07 Nov 2020 17:53:55	[REDACTED]	test notes



TGH

Making Integrations Simpler



TGH Software Solutions Pvt. Ltd.

www.techygeekhub.com

At TGH, we specialize in driving digital transformation through seamless Integration Technologies.

Operating as an INTEGRATION FACTORY, we serve as a one-stop shop for all your integration needs. Our expert team is well-versed in enterprise software and legacy system integration, along with leading iPaaS technologies like Boomi, MuleSoft, Workato, OIC, and more.

We're committed to enhancing business processes and solving problems through our integration expertise.



Email address

connect@techygeekhub.com



Phone number

+ 011-40071137
+ 91-8810610395



Our offices

Noida Office

iThum
Plot No -40, Tower A,
Office No: 712,
Sector-62, Noida,
Uttar Pradesh, 201301

Hyderabad Office

Plot no: 6/3, 5th Floor,
Techno Pearl Building,
HUDA Techno Enclave,
HITEC City, Hyderabad,
Telangana 500081

