TGH

Making Integrations Simpler

boomi Partner

XML TO PDF Conversion

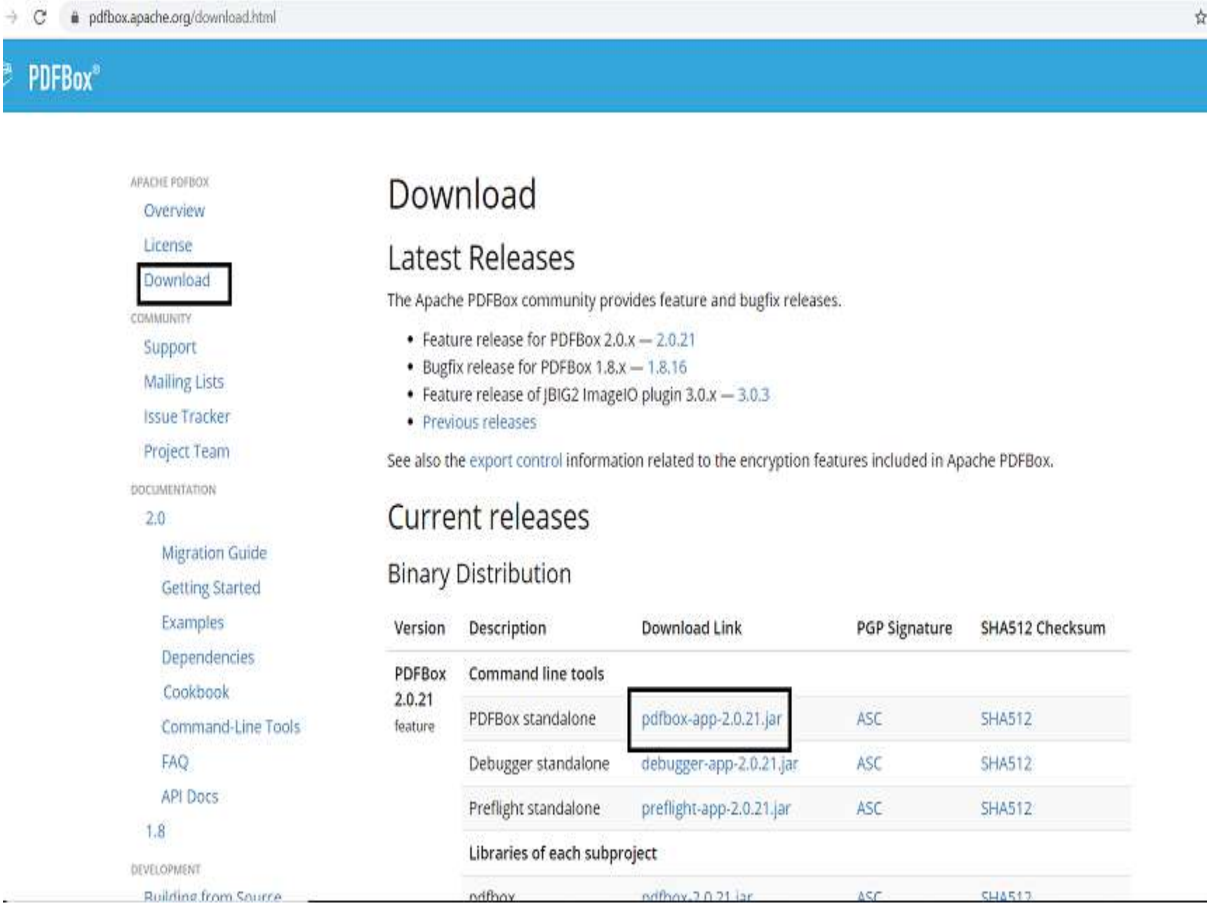
XML to PDF Conversion

In this blog, we will develop a process which will convert XML into PDF in Dell Boomi.

Boomi does not support PDF conversion directly, but we can do it by importing PDF libraries and using custom scripting to convert documents into PDF Format.

First, let us see how to import PDF Libraries and deploy them to the atom. In this Use Case, we are importing two PDF libraries i.e. Apache PDFBox and iText.

Step 1: To import the Apache PDFBox library, go to https://pdfbox.apache.org/ > Download and choose the jar as shown. The jar gets downloaded.



pdfbox.apache.org/download.html

PDFBox®

APACHE PDFBOX
- Overview
- License
- Download

COMMUNITY
- Support
- Mailing Lists
- Issue Tracker
- Project Team

DOCUMENTATION
- 2.0
  - Migration Guide
  - Getting Started
  - Examples
  - Dependencies
  - Cookbook
  - Command-Line Tools
  - FAQ
  - API Docs
- 1.8

DEVELOPMENT
- Building from Source

Download

Latest Releases

The Apache PDFBox community provides feature and bugfix releases.

- Feature release for PDFBox 2.0.x — 2.0.21
- Bugfix release for PDFBox 1.8.x — 1.8.16
- Feature release of JBIG2 ImageIO plugin 3.0.x — 3.0.3
- Previous releases

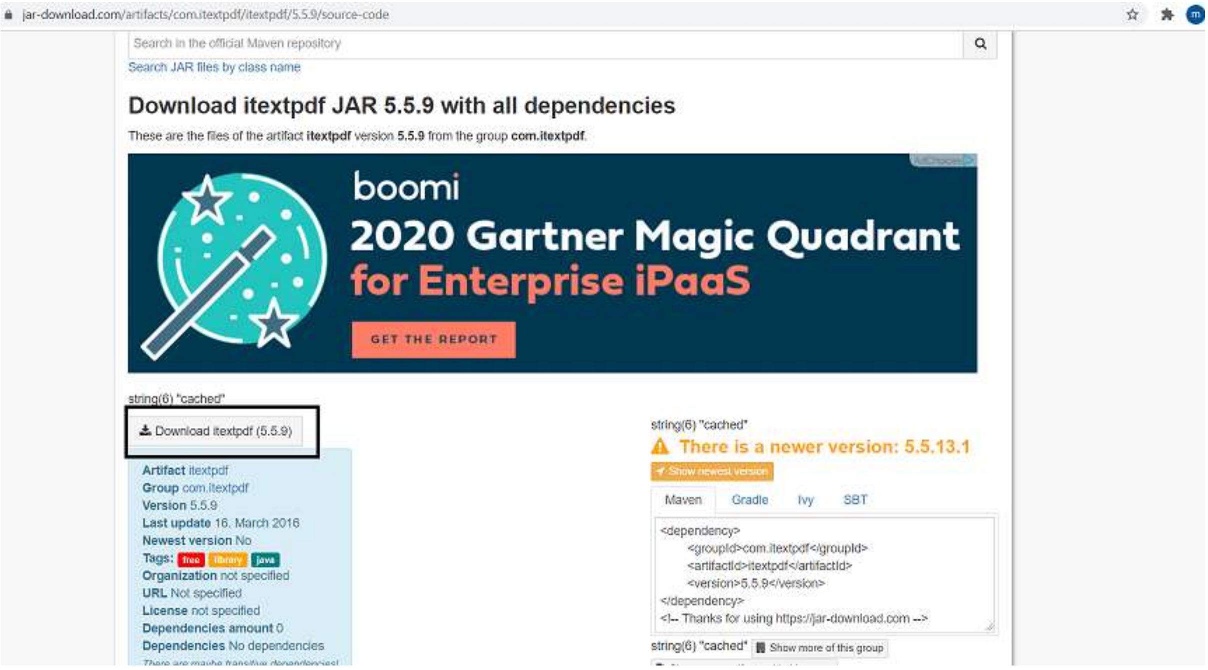
See also the export control information related to the encryption features included in Apache PDFBox.

Current releases

Binary Distribution

Version	Description	Download Link	PGP Signature	SHA512 Checksum
PDFBox 2.0.21 feature	Command line tools			
	PDFBox standalone	pdfbox-app-2.0.21.jar	ASC	SHA512
	Debugger standalone	debugger-app-2.0.21.jar	ASC	SHA512
	Preflight standalone	preflight-app-2.0.21.jar	ASC	SHA512
	Libraries of each subproject			
	pdfbox	pdfbox-2.0.21.jar	ASC	SHA512

Step 2: To import the iText library, go to https://jardownload.com/artifacts/com.itextpdf/itextpdf/5.5.9/source-code and choose the jar as shown. Once we choose it, the jar gets downloaded.



jar-download.com/artifacts/com.itextpdf/itextpdf/5.5.9/source-code

Search in the official Maven repository

Search JAR files by class name

Download itextpdf JAR 5.5.9 with all dependencies

These are the files of the artifact **itextpdf** version **5.5.9** from the group **com.itextpdf**.

string(6) "cached"

Download itextpdf (5.5.9)

Artifact itextpdf
Group com.itextpdf
Version 5.5.9
Last update 16. March 2016
Newest version No
Tags: free library java
Organization not specified
URL Not specified
License not specified
Dependencies amount 0
Dependencies No dependencies
There are maybe transitive dependencies!

string(6) "cached"

There is a newer version: 5.5.13.1

Show newest version

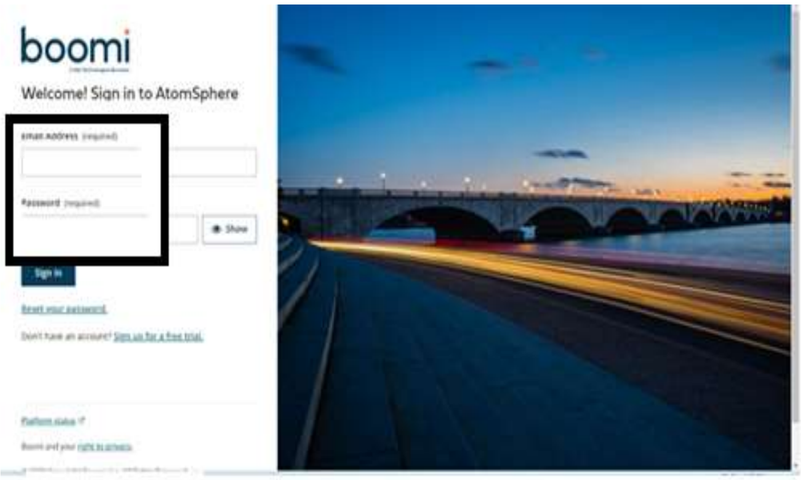
Maven Gradle Ivy SBT

```
<dependency>
    <groupId>com.itextpdf</groupId>
    <artifactId>itextpdf</artifactId>
    <version>5.5.9</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->
```

string(6) "cached" Show more of this group

Now, we will upload the downloaded jar files in the Account Library to the Boomi Account.

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



boomi

Welcome! Sign in to AtomSphere

Email Address (required)

Password (required)

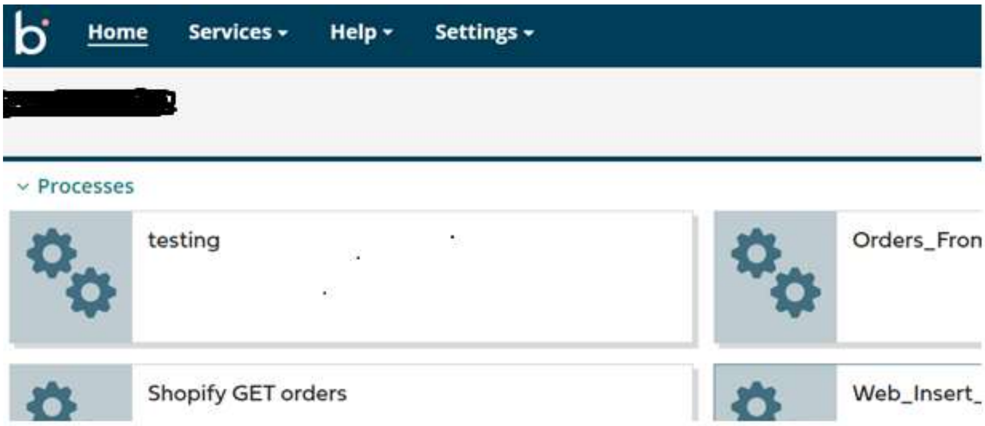
Reset your password.

Don't have an account? Sign up for a free trial.

Platform status

Boomi and your right to privacy.

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



[redacted]

Processes

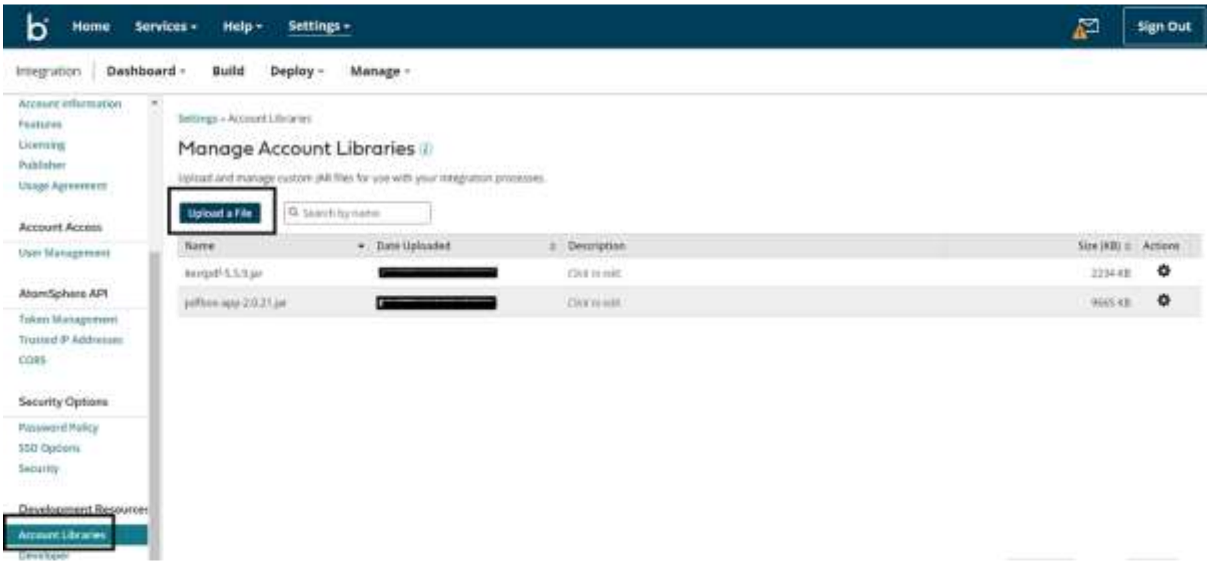
testing

Orders_Fron

Shopify GET orders

Web_Insert_

Step 3: Go to Settings > Account Information and Setup > Account Libraries and upload the jar files which we have downloaded by choosing Upload a File.



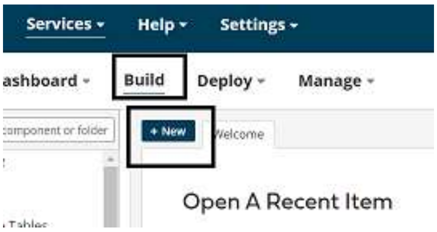
Settings › Account Libraries

Manage Account Libraries

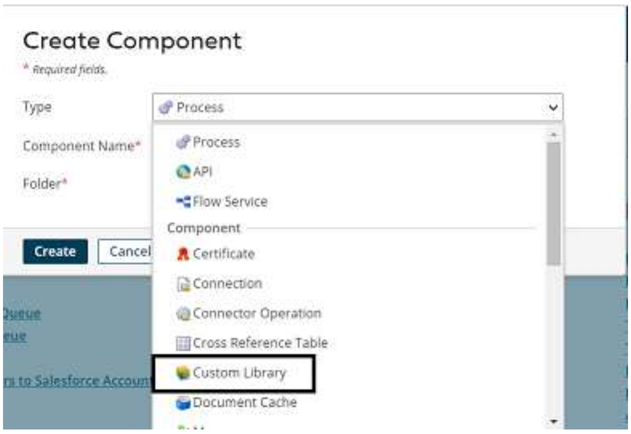
Upload and manage custom JAR files for use with your integration processes.

Name	Date Uploaded	Description	Size (KB)	Actions
itextpdf-5.5.9.jar	[redacted]	Click to edit.	2234 KB	⚙
pdfbox-app-2.0.21.jar	[redacted]	Click to edit.	9665 KB	⚙

Step 4: Now, create a custom library component and add the uploaded JAR files in the account library to the component. Navigate to the Build tab and click on New as shown.



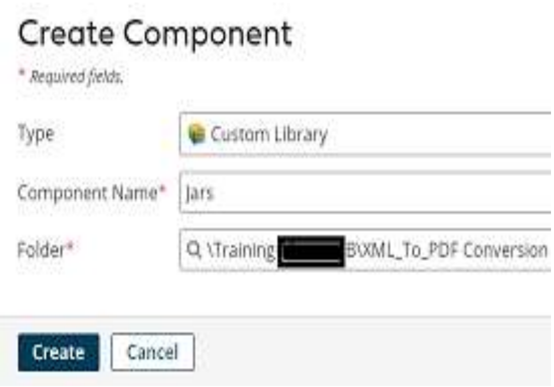
Step 5: Choose Type as Custom Library. Component Name and Folder can be given based on your choice. Click on Create.



Create Component

* Required fields.

Type: Process



Create Component

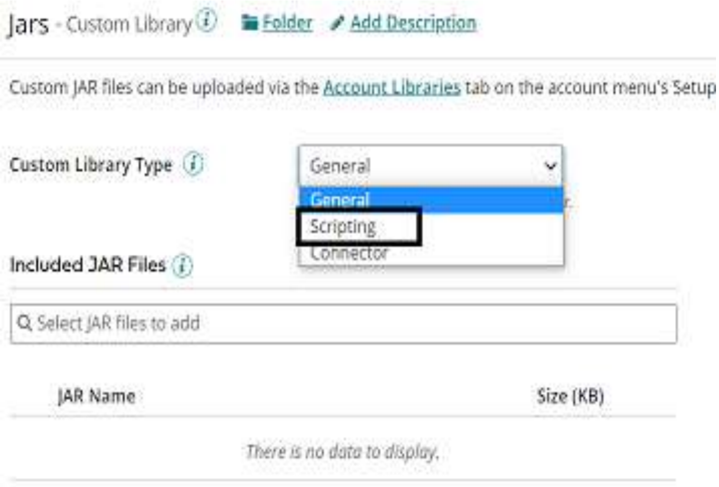
* Required fields.

Type: Custom Library

Component Name*: Jars

Folder*: \Training[redacted]B\XML_To_PDF Conversion

Step 6: Choose Custom Library Type as Scripting from the drop down and these jars will be available for scripting.



Jars - Custom Library

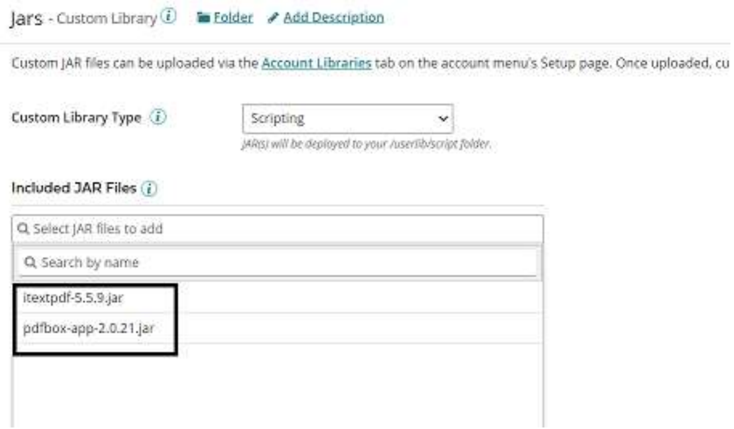
Custom JAR files can be uploaded via the Account Libraries tab on the account menu's Setup

Custom Library Type: General

Included JAR Files

JAR Name	Size (KB)
There is no data to display.	

- Select Jars which we have uploaded in Account Library. Click save.



Jars - Custom Library ⓘ Folder Add Description

Custom JAR files can be uploaded via the Account Libraries tab on the account menu's Setup page. Once uploaded, cu

Custom Library Type ⓘ: Scripting

JAR(s) will be deployed to your /userlib/script folder.

Included JAR Files ⓘ

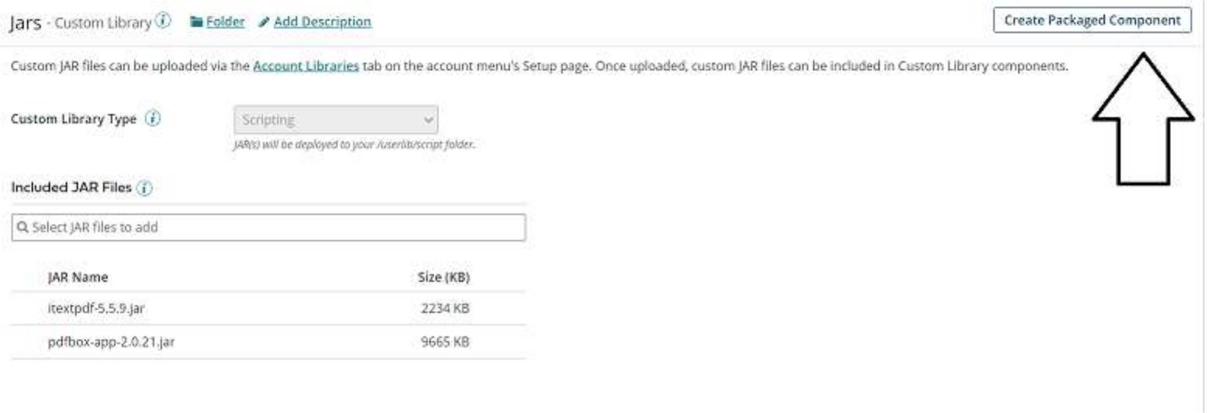
Select JAR files to add

Search by name

itextpdf-5.5.9.jar

pdfbox-app-2.0.21.jar

Step 7: Deploy the Custom Library component to an atom because these are not explicitly referenced by a process or any other component, we must deploy them independently. Click on Create Packaged Component.



Jars - Custom Library ⓘ Folder Add Description

Custom JAR files can be uploaded via the Account Libraries tab on the account menu's Setup page. Once uploaded, custom JAR files can be included in Custom Library components.

Custom Library Type ⓘ: Scripting

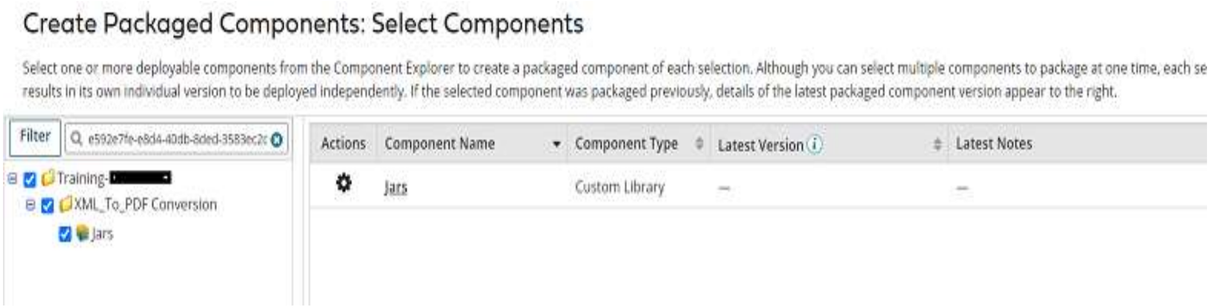
JAR(s) will be deployed to your /userlib/script folder.

Included JAR Files ⓘ

Select JAR files to add

JAR Name	Size (KB)
itextpdf-5.5.9.jar	2234 KB
pdfbox-app-2.0.21.jar	9665 KB

Step 8: The process gets selected automatically. Click Add Details.



Create Packaged Components: Select Components

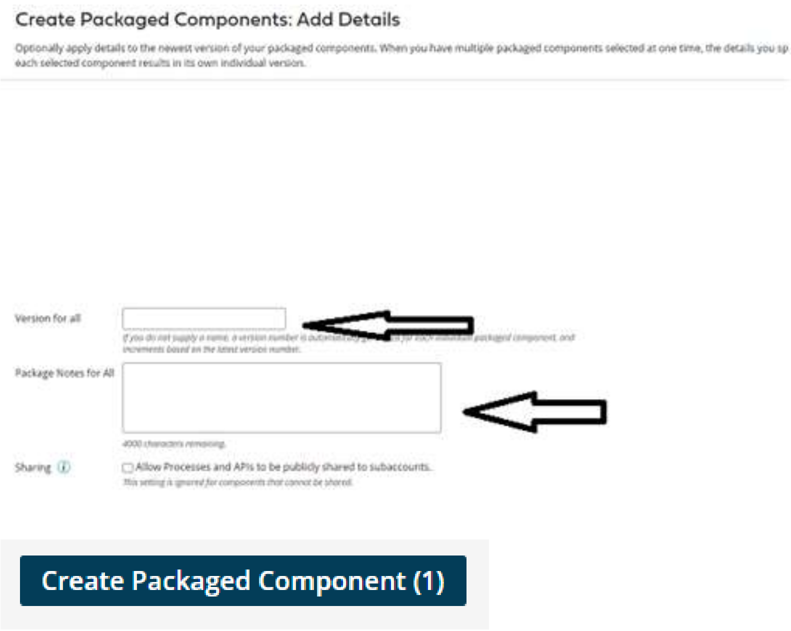
Select one or more deployable components from the Component Explorer to create a packaged component of each selection. Although you can select multiple components to package at one time, each se results in its own individual version to be deployed independently. If the selected component was packaged previously, details of the latest packaged component version appear to the right.

Filter: e592e7fe-e8d4-40db-8ded-3583ec2c

☑ Training-[redacted]
☑ XML_To_PDF Conversion
☑ Jars

Actions	Component Name	Component Type	Latest Version ⓘ	Latest Notes
⚙	Jars	Custom Library	–	–

- Next, select the version and write notes if you have any. Click on Create Packaged Component.



Create Packaged Components: Add Details

Optionally apply details to the newest version of your packaged components. When you have multiple packaged components selected at one time, the details you sp each selected component results in its own individual version.

Version for all

If you do not supply a name, a version number is automatically generated for each individual packaged component, and increments based on the latest version number.

Package Notes for All

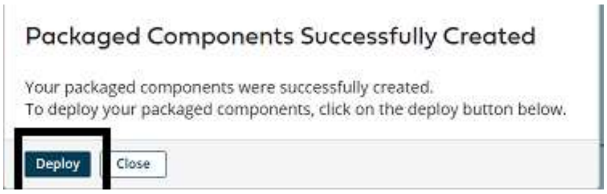
4000 characters remaining.

Sharing

□ Allow Processes and APIs to be publicly shared to subaccounts.

This setting is ignored for components that cannot be shared.

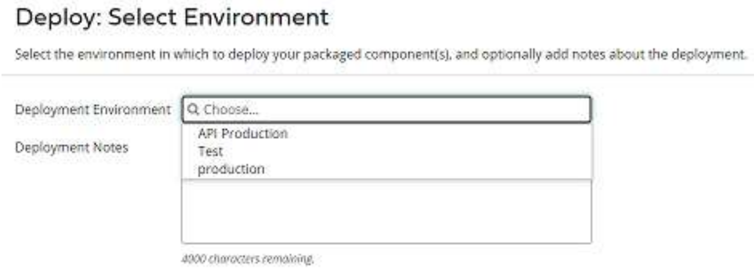
- Now, we see that the package has been created successfully and click on deploy.



Packaged Components Successfully Created

Your packaged components were successfully created.
To deploy your packaged components, click on the deploy button below.

- We will then have to select the environment. Choose production and click select version and review.



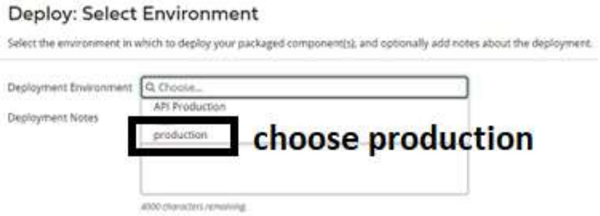
Deploy: Select Environment

Select the environment in which to deploy your packaged component(s), and optionally add notes about the deployment.

Deployment Environment

Deployment Notes

4000 characters remaining.



Deploy: Select Environment

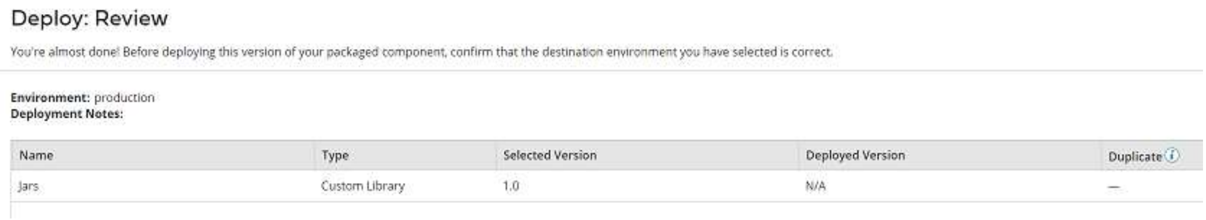
Select the environment in which to deploy your packaged component(s), and optionally add notes about the deployment.

Deployment Environment

Deployment Notes

4000 characters remaining.

- We will be asked to cross-check the environment which we have configured in the deployment tab.



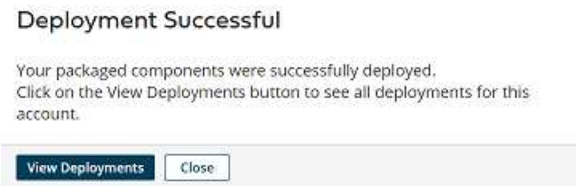
Deploy: Review

You're almost done! Before deploying this version of your packaged component, confirm that the destination environment you have selected is correct.

Environment: production
Deployment Notes:

Name	Type	Selected Version	Deployed Version	Duplicate ⓘ
Jars	Custom Library	1.0	N/A	—

- Once, we click on deploy we will be able to see that deployment is done successfully. Click save and close.

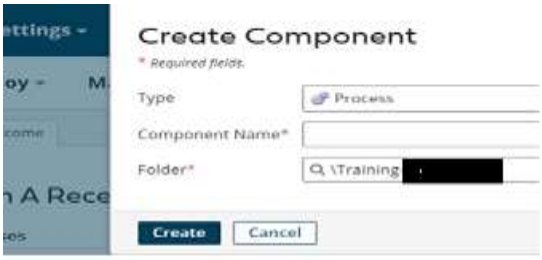


Deployment Successful

Your packaged components were successfully deployed.
Click on the View Deployments button to see all deployments for this account.

Now, we will create a process which reads JSON data from the Disk directory and converts it into XML. Then, XML data will be covered in PDF Format using scripting.

Step 9: Click on New to create a process. we will be able to see three fields i.e. Type, Component Name and Folder.



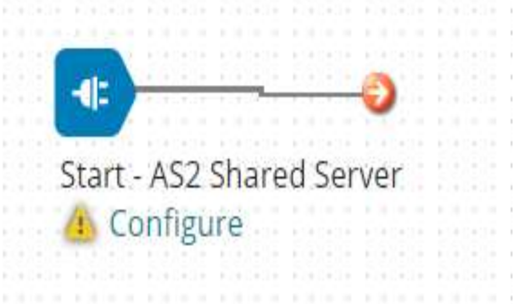
Create Component

* Required fields.

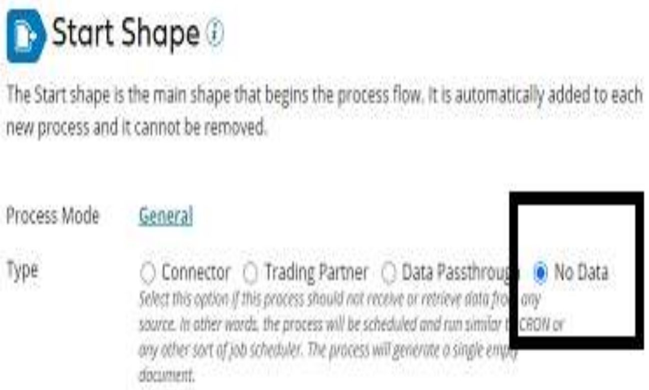
Type: Process
Component Name*:
Folder*: \Training [redacted]

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



Step 11: Select the start shape and choose No Data. Click ok.



Start Shape

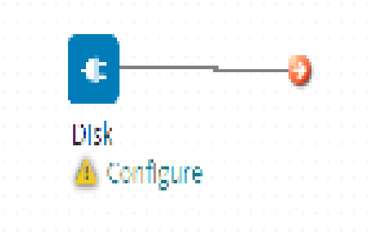
The Start shape is the main shape that begins the process flow. It is automatically added to each new process and it cannot be removed.

Process Mode General

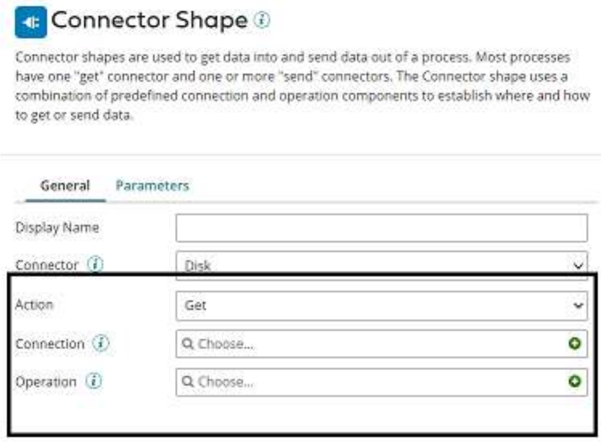
Type ○ Connector ○ Trading Partner ○ Data Passthrough ● No Data

Select this option if this process should not receive or retrieve data from any source. In other words, the process will be scheduled and run similar to CRON or any other sort of job scheduler. The process will generate a single empty document.

Step 12: Drag and drop the disk connector shape onto the process canvas to read a file from a specific directory.



- We have to configure 3 fields in connector i.e. Action, Connector and Operation.



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

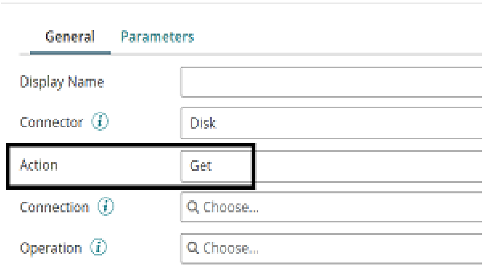
Display Name	
Connector	Disk
Action	Get
Connection	Choose...
Operation	Choose...

- We see 2 actions i.e. Get and Send in Actions.

  Get – To get the data from a disk location.

  Send – To send to the disk location.

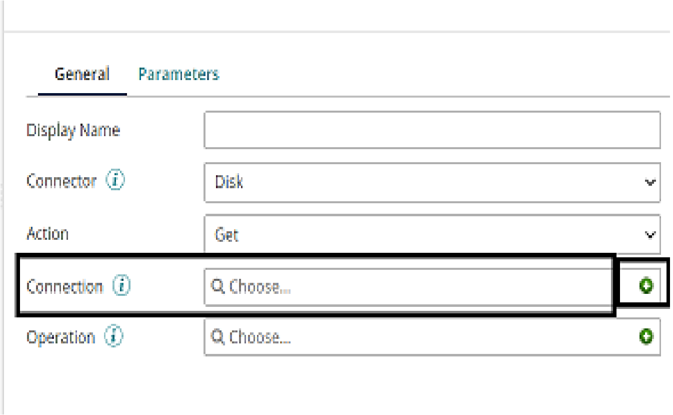
- Here, we will choose action as **GET** as we are reading the file.



General | Parameters

Display Name	
Connector	Disk
Action	Get
Connection	Choose...
Operation	Choose...

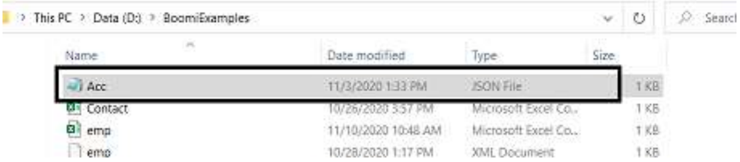
- Click + on connection to create a new one.



General | Parameters

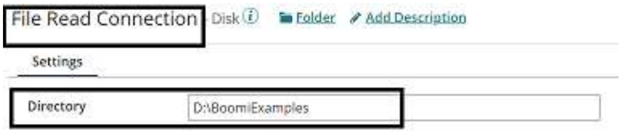
Display Name	
Connector	Disk
Action	Get
Connection	Choose...
Operation	Choose...

- Name the file and give the directory from where we want to read the file. Here, we are reading JSON files from D drive and Boomi Examples folder as shown in the screenshot.



This PC > Data (D:) > BoomiExamples

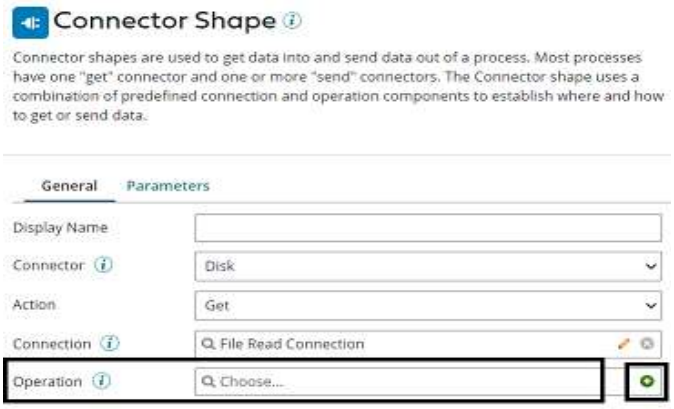
Name	Date modified	Type	Size
Acc	11/3/2020 1:33 PM	JSON File	1 KB
Contact	10/26/2020 3:57 PM	Microsoft Excel Co...	1 KB
emp	11/10/2020 10:48 AM	Microsoft Excel Co...	1 KB
emp	10/28/2020 1:17 PM	XML Document	1 KB



File Read Connection Disk

Directory: D:\BoomiExamples

- Click save and close
- Now, we will configure the operation. Click + on operation to create a new one.



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.

Display Name:

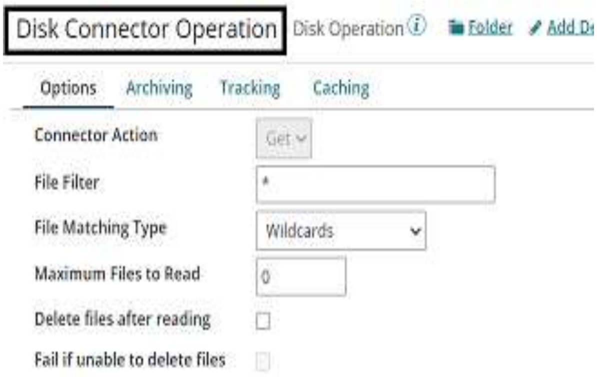
Connector: Disk

Action: Get

Connection: File Read Connection

Operation: Choose...

- Name the operation and configure the following.



Disk Connector Operation Disk Operation

Connector Action: Get

File Filter: *

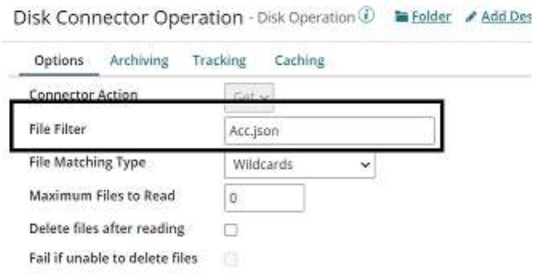
File Matching Type: Wildcards

Maximum Files to Read: 0

Delete files after reading: □

Fail if unable to delete files: □

- **File Filter**: Read-only files with a file name that matches the file filter. Here, it will be **Acc.json**



Disk Connector Operation - Disk Operation

Connector Action: Get

File Filter: Acc.json

File Matching Type: Wildcards

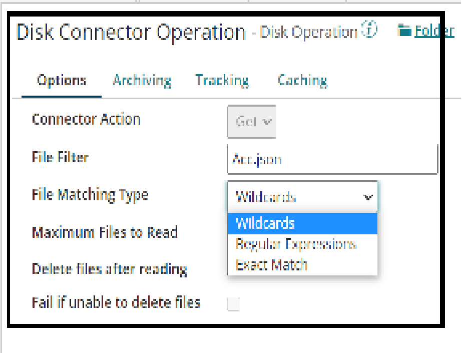
Maximum Files to Read: 0

Delete files after reading □

Fail if unable to delete files □

File Matching Type:

- **Wildcards** uses simple file filters like * and. * represent multiple characters and ? represents a single character.
- **Regular Expressions** can include complex regular expressions.
- **Exact Match** includes the filename that we are reading.



Disk Connector Operation - Disk Operation

Connector Action: Get

File Filter: Acc.json

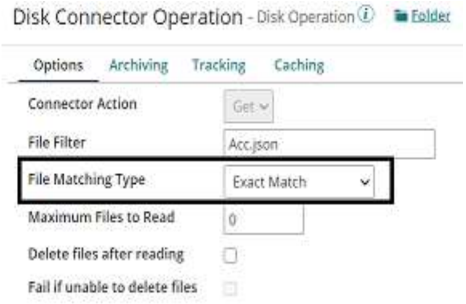
File Matching Type: Wildcards

Maximum Files to Read

Delete files after reading

Fail if unable to delete files □

- Here, the file matching type would be **Exact Match** as we are giving the file name.



Disk Connector Operation - Disk Operation

Connector Action: Get

File Filter: Acc.json

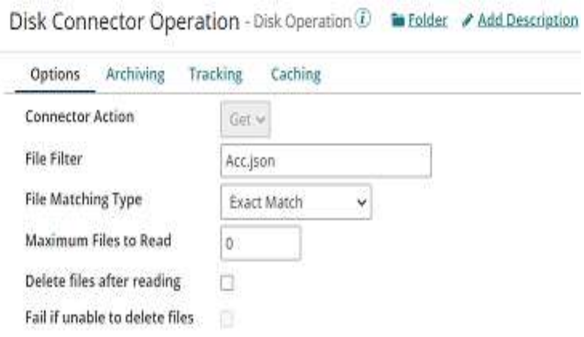
File Matching Type: Exact Match

Maximum Files to Read: 0

Delete files after reading □

Fail if unable to delete files □

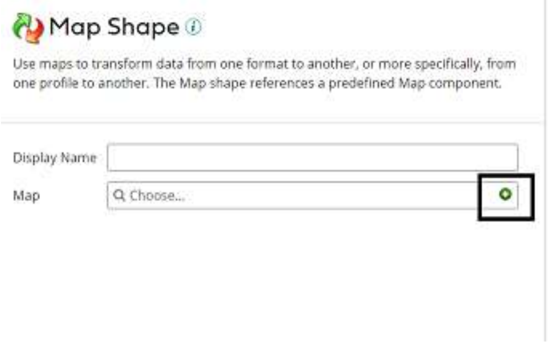
- **Maximum files to read**: It sets the maximum number of files to be read at one time. Let it be default.
- **Delete files after reading**: If we want the file to be read and deleted, we can check this option. Here, we are leaving it to default. Click save and close
- The complete disk operation looks like this,



Disk Connector Operation - Disk Operation

Connector Action	Get
File Filter	Acc.json
File Matching Type	Exact Match
Maximum Files to Read	0
Delete files after reading	□
Fail if unable to delete files	□

Step 13: Drag and drop the Map shape onto the process canvas. Click + and Name the map.



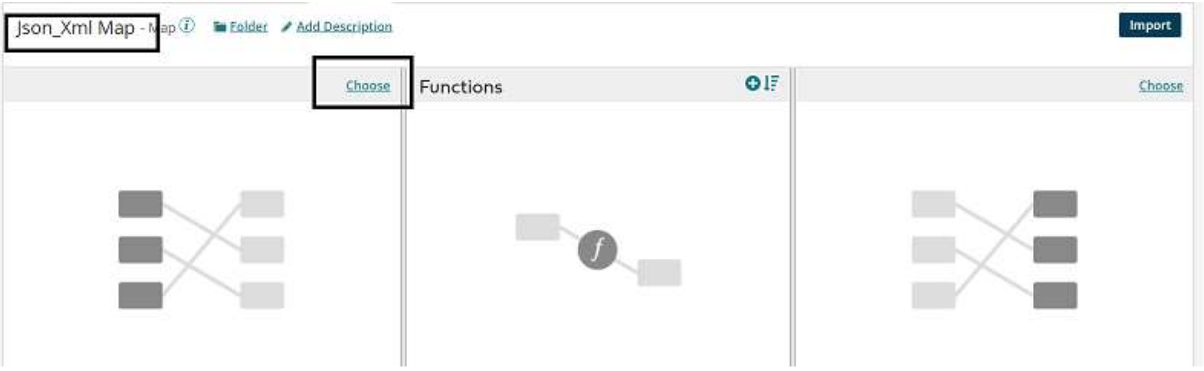
Map Shape

Use maps to transform data from one format to another, or more specifically, from one profile to another. The Map shape references a predefined Map component.

Display Name

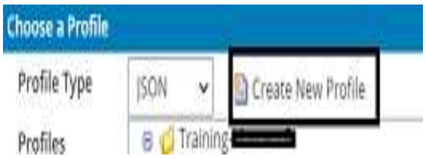
Map: Choose...

- Add source side and Destination side profiles to the map. Here, the source side would be the JSON profile and the Destination side would be the XML profile. First, we will add a Source profile. Click choose and select the folder where we saved the profile.



Json_Xml Map - Map

- Here, the Profile Type will be JSON and click on Create New Profile.

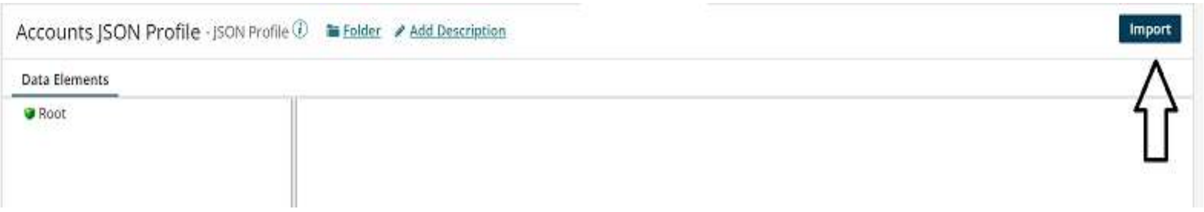


Choose a Profile

Profile Type JSON Create New Profile

Profiles Training [redacted]

- Name the profile and click on import. In this example, we used the below file as an input.

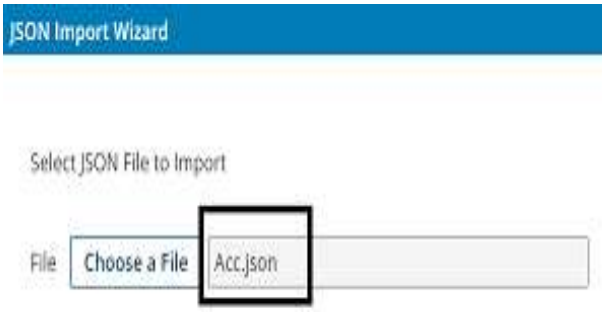


Accounts JSON Profile - JSON Profile Folder Add Description Import

Data Elements

Root

- Choose the file from the directory where we have saved the file. Click on Next.

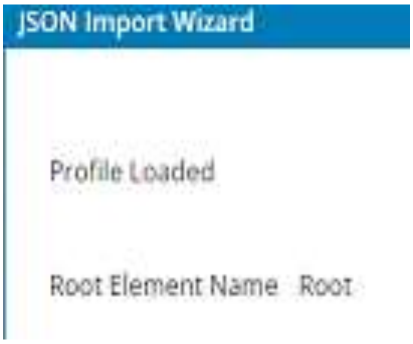


JSON Import Wizard

Select JSON File to Import

File Choose a File Acc.json

- We see that the profile is imported.

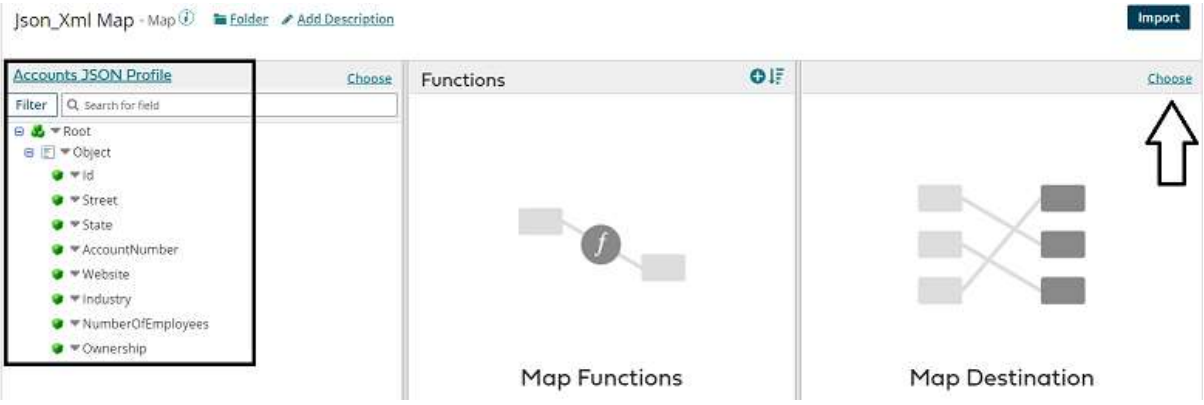


JSON Import Wizard

Profile Loaded

Root Element Name Root

Step 14: Now, add the destination profile. Click choose on the right side as shown.



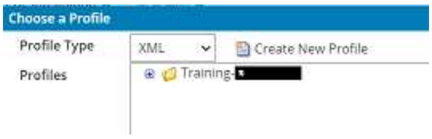
Json_Xml Map - Map

Accounts JSON Profile

Filter: Search for field

Root
Object
Id
Street
State
AccountNumber
Website
Industry
NumberOfEmployees
Ownership

- Select Profile Type as XML and click on Create New Profile.



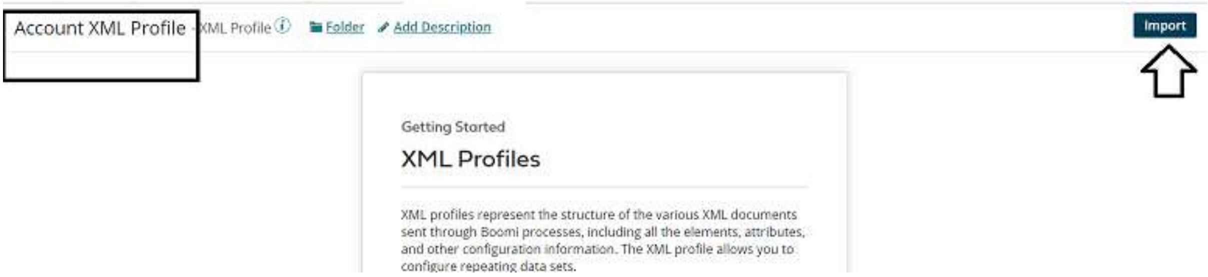
Choose a Profile

Profile Type: XML

Profiles: Training-[redacted]

- Name the profile and click on import. In this example, we used the below file as an input.

Account.xml



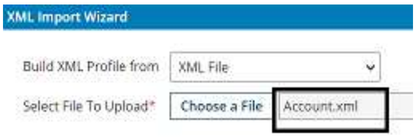
Account XML Profile - XML Profile

Getting Started

XML Profiles

XML profiles represent the structure of the various XML documents sent through Boomi processes, including all the elements, attributes, and other configuration information. The XML profile allows you to configure repeating data sets.

- Choose a file from the directory where we saved it. Click Next.

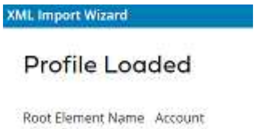


XML Import Wizard

Build XML Profile from: XML File

Select File To Upload*: Account.xml

- We see that the profile has been imported. Click finish, save and close.

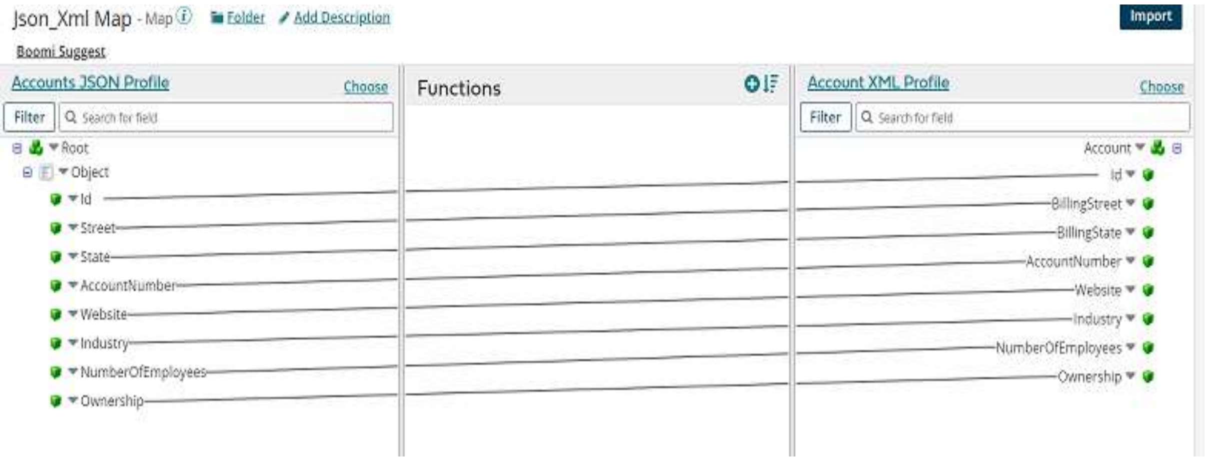


XML Import Wizard

Profile Loaded

Root Element Name Account

- We see that source and destination profiles have been imported. Provide one-to-one mapping from source to destination. Click save and close.



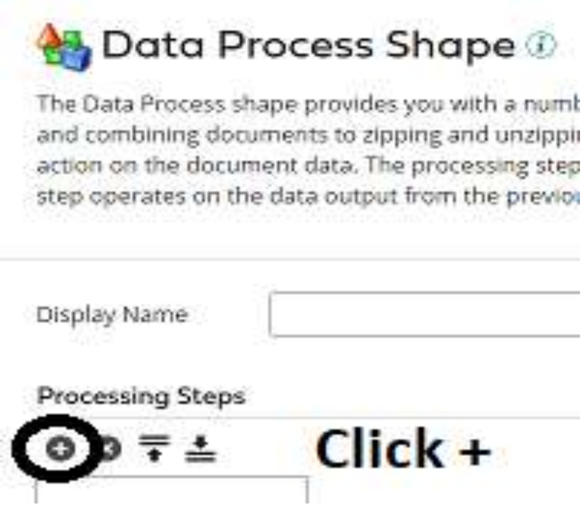
Json_Xml Map - Map
Folder
Add Description
Import
Boomi Suggest

Accounts JSON Profile
Choose
Filter
Search for field
Root
Object
Id
Street
State
AccountNumber
Website
Industry
NumberOfEmployees
Ownership

Functions

Account XML Profile
Choose
Filter
Search for field
Account
Id
BillingStreet
BillingState
AccountNumber
Website
Industry
NumberOfEmployees
Ownership

Step 15: To write a Groovy Script, drag and drop the data process shape onto the process canvas and click on + as shown.



Data Process Shape

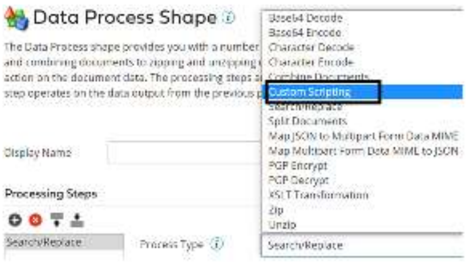
The Data Process shape provides you with a numb
and combining documents to zipping and unzippi
action on the document data. The processing step
step operates on the data output from the previo

Display Name

Processing Steps

Click +

- Choose Process Type as custom scripting from the drop-down. Click on edit.



Data Process Shape

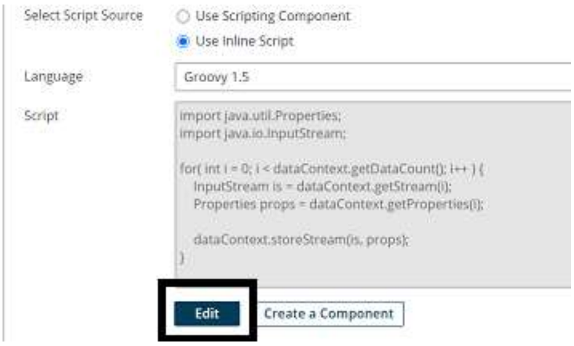
The Data Process shape provides you with a number
and combining documents to zipping and unzipping
action on the document data. The processing steps a
step operates on the data output from the previous p

Display Name

Processing Steps

Search/Replace

Process Type: Search/Replace



Select Script Source: ○ Use Scripting Component ● Use Inline Script

Language: Groovy 1.5

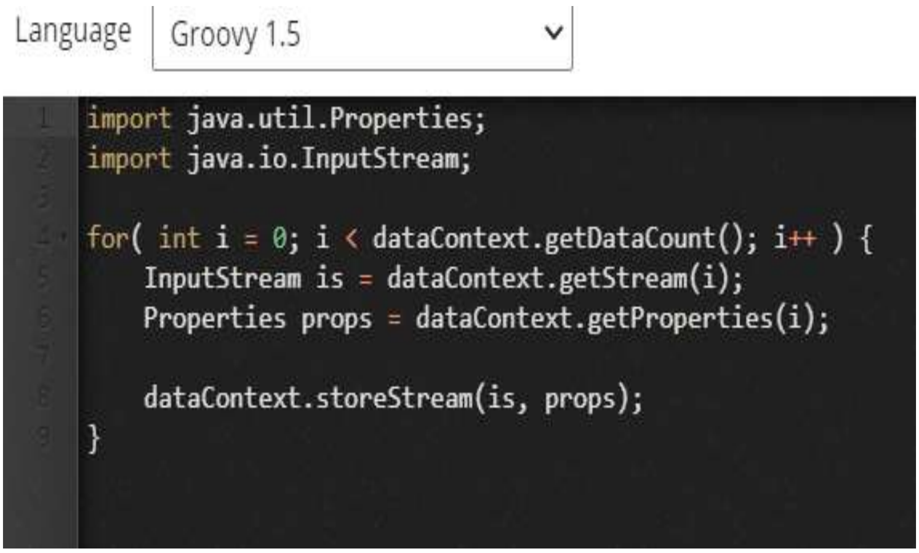
Script:

```
import java.util.Properties;
import java.io.InputStream;

for( int i = 0; i < dataContext.getDataCount(); i++ ) {
  InputStream is = dataContext.getStream(i);
  Properties props = dataContext.getProperties(i);

  dataContext.storeStream(is, props);
}
```

Step 16: We will be able to see the command prompt as shown in the screenshot.



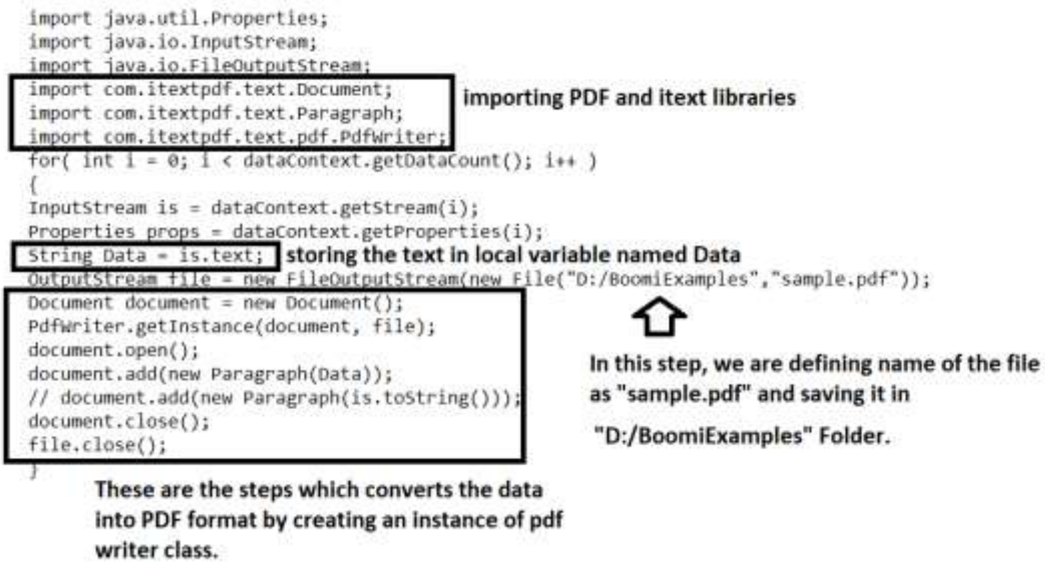
Language: Groovy 1.5

```
import java.util.Properties;
import java.io.InputStream;

for( int i = 0; i < dataContext.getDataCount(); i++ ) {
    InputStream is = dataContext.getStream(i);
    Properties props = dataContext.getProperties(i);

    dataContext.storeStream(is, props);
}
```

Step 17: Now, we will write a groovy script to convert XML into PDF Format as shown below.



```
import java.util.Properties;
import java.io.InputStream;
import java.io.FileOutputStream;
import com.itextpdf.text.Document;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;
for( int i = 0; i < dataContext.getDataCount(); i++ )
{
InputStream is = dataContext.getStream(i);
Properties props = dataContext.getProperties(i);
String Data = is.text;
OutputStream file = new FileOutputStream(new File("D:/BoomiExamples","sample.pdf"));
Document document = new Document();
PdfWriter.getInstance(document, file);
document.open();
document.add(new Paragraph(Data));
// document.add(new Paragraph(is.toString()));
document.close();
file.close();
}
```

importing PDF and itext libraries

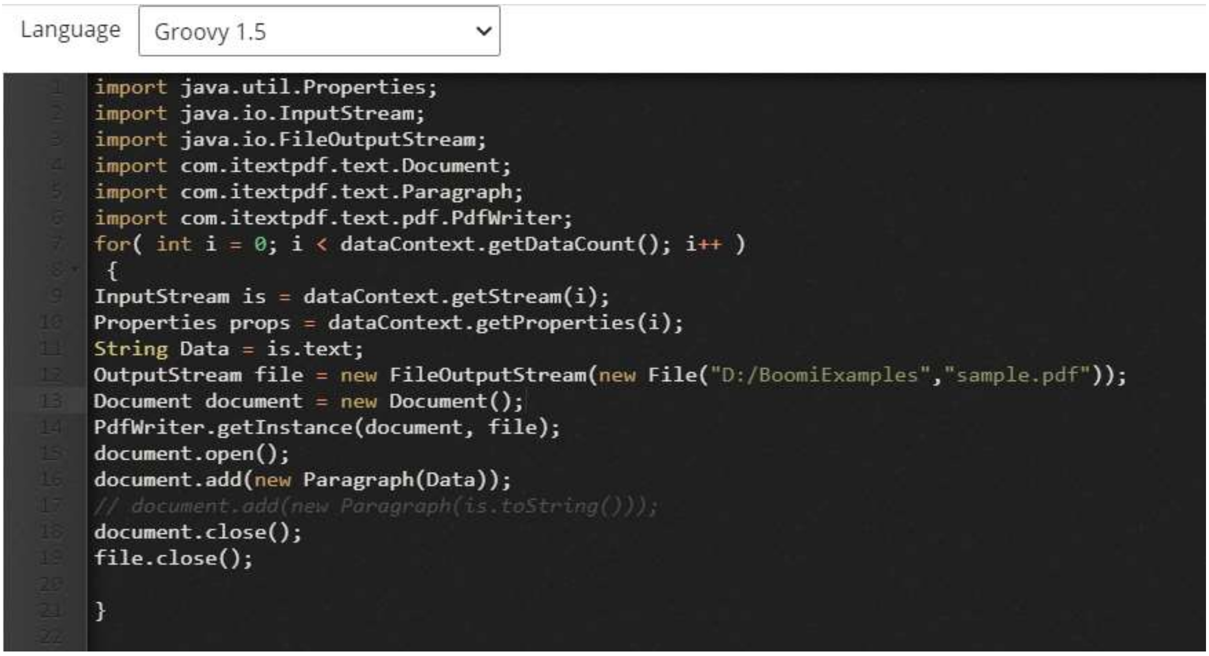
storing the text in local variable named Data

In this step, we are defining name of the file as "sample.pdf" and saving it in "D:/BoomiExamples" Folder.

These are the steps which converts the data into PDF format by creating an instance of pdf writer class.

- Add the below script in the command prompt and click OK.

```
import java.util.Properties;
import java.io.InputStream;
import java.io.FileOutputStream;
import com.itextpdf.text.Document;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;
for( int i = 0; i < data context.getDataCount(); i++ )
{
InputStream is = dataContext.getStream(i);
Properties props = dataContext.getProperties(i);
String Data = is.text;
OutputStream file = new FileOutputStream(new File("D:/BoomiExamples","sample.pdf"));
Document document = new Document();
PdfWriter.getInstance(document, file);
document.open();
document.add(new Paragraph(Data));
// document.add(new Paragraph(is.toString()));
document.close();
file.close();
}
```



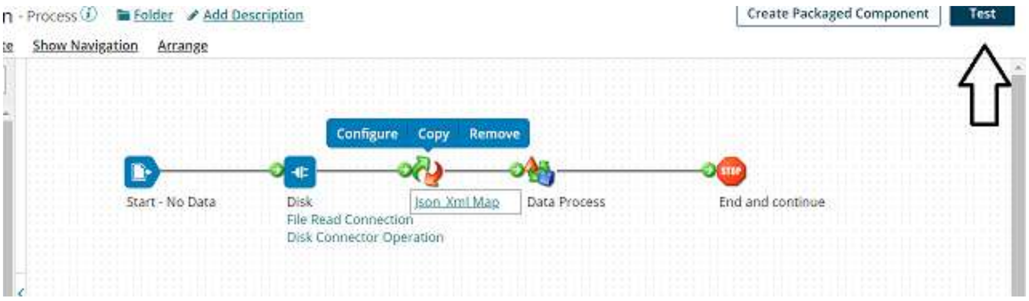
Language Groovy 1.5

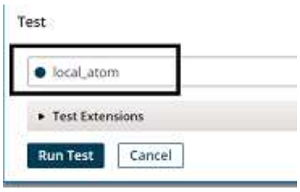
```
import java.util.Properties;
import java.io.InputStream;
import java.io.FileOutputStream;
import com.itextpdf.text.Document;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;
for( int i = 0; i < dataContext.getDataCount(); i++ )
 {
InputStream is = dataContext.getStream(i);
Properties props = dataContext.getProperties(i);
String Data = is.text;
OutputStream file = new FileOutputStream(new File("D:/BoomiExamples","sample.pdf"));
Document document = new Document();
PdfWriter.getInstance(document, file);
document.open();
document.add(new Paragraph(Data));
// document.add(new Paragraph(is.toString()));
document.close();
file.close();

}
```

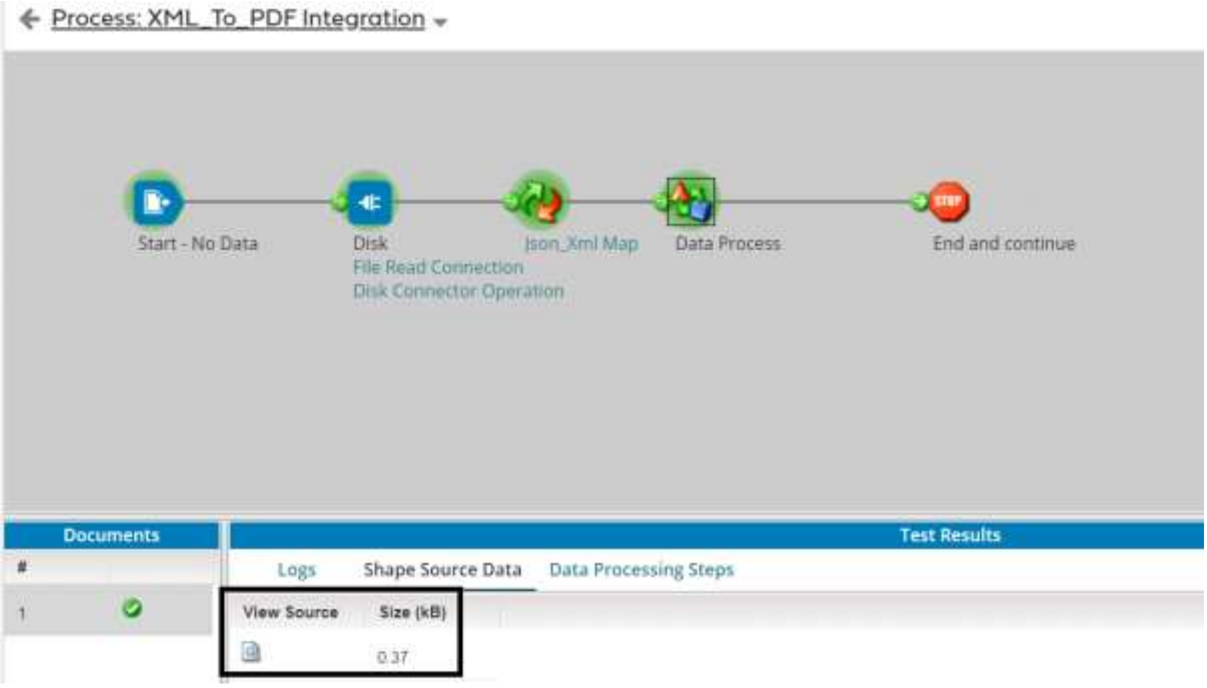
Step 18: Drag and drop the stop shape onto the process canvas to indicate the end of the flow.

Step 19: Arrange all the shapes in order and test them by configuring the local atom as shown below.

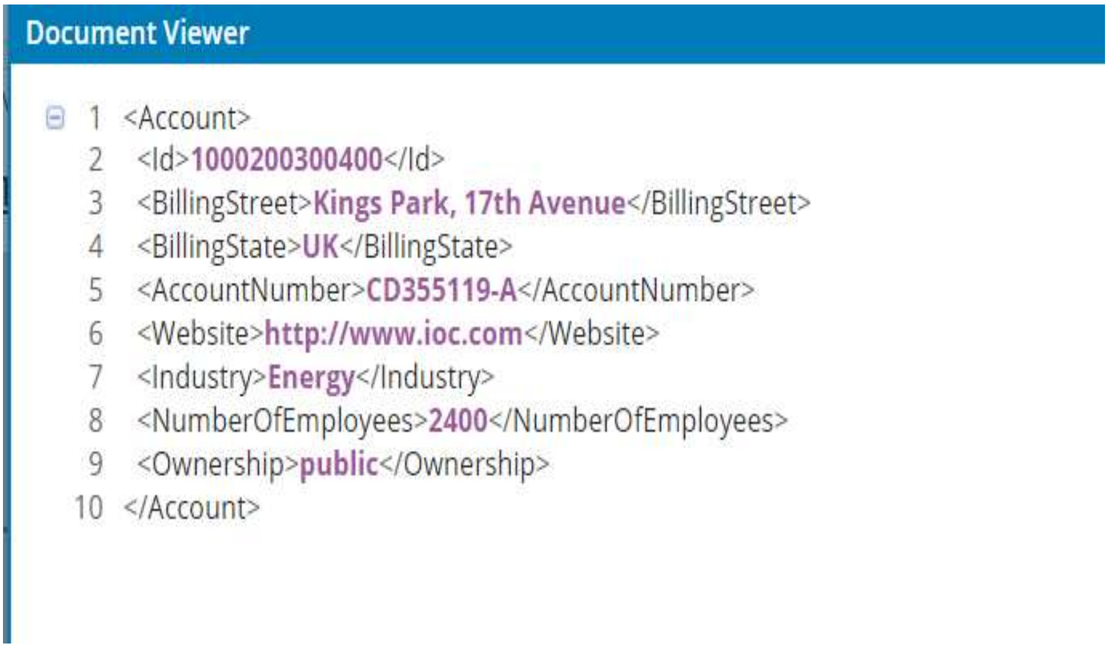




Step 20: We see that the process has been executed. Click on view source to see the output.

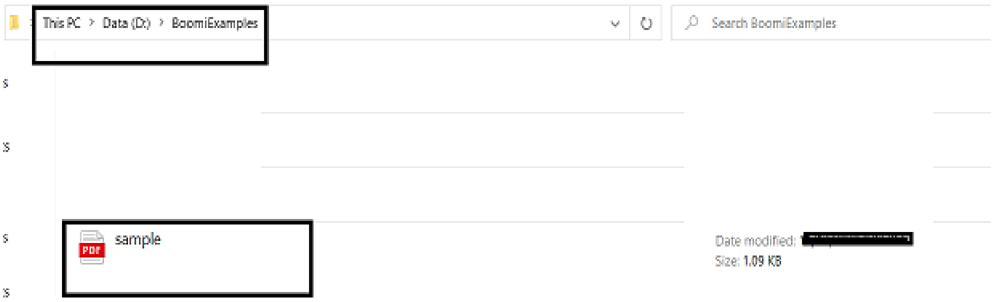


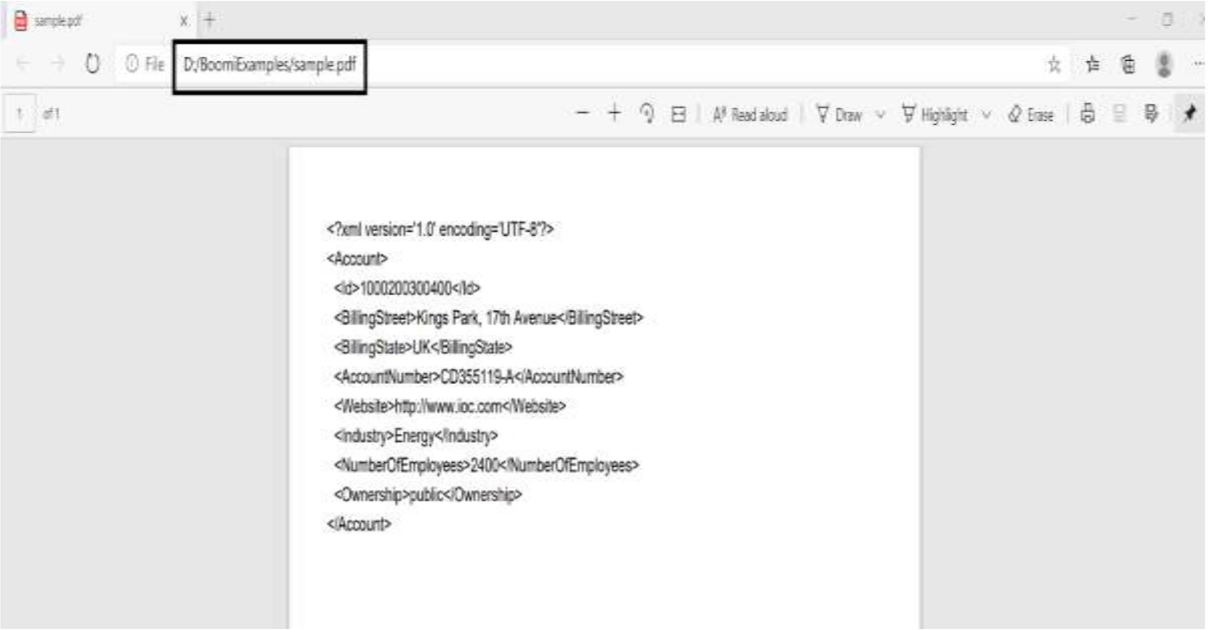
- Once, we open the document, it looks like



```
<Account>
  <Id>1000200300400</Id>
  <BillingStreet>Kings Park, 17th Avenue</BillingStreet>
  <BillingState>UK</BillingState>
  <AccountNumber>CD355119-A</AccountNumber>
  <Website>http://www.ioc.com</Website>
  <Industry>Energy</Industry>
  <NumberOfEmployees>2400</NumberOfEmployees>
  <Ownership>public</Ownership>
</Account>
```

Step 21: Navigate to the "**D:/BoomiExamples**" folder and we see that a file has been created with the name "sample" in pdf format.


This PC > Data (D:) > BoomiExamples
Search BoomiExamples
sample
Date modified: [redacted]
Size: 1.09 KB



```
<?xml version='1.0' encoding='UTF-8'?>
<Account>
  <Id>1000200300400</Id>
  <BillingStreet>Kings Park, 17th Avenue</BillingStreet>
  <BillingState>UK</BillingState>
  <AccountNumber>CD355119-A</AccountNumber>
  <Website>http://www.ioc.com</Website>
  <Industry>Energy</Industry>
  <NumberOfEmployees>2400</NumberOfEmployees>
  <Ownership>public</Ownership>
</Account>
```

TGH

Making Integrations Simpler

boomi Partner

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