**Requirements for API Inbound Setup and How to Test**

Last update date: **26-Aug-2019 |** Version: **1.7**

**Setup**

**Step 1: Database Tables and Initial Configuration Data Required**

1. Ensure that the database tables APIInbound, APIInboundDetail and APIInboundEvent are available in ASI database
2. Ensure that the database tables APIInbound and APIInboundEvent have initial configuration data

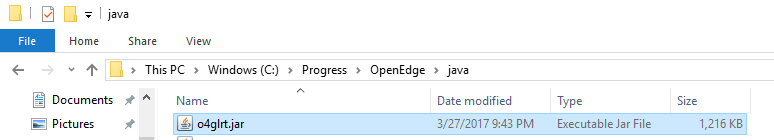
**Step 2: Programs Required**

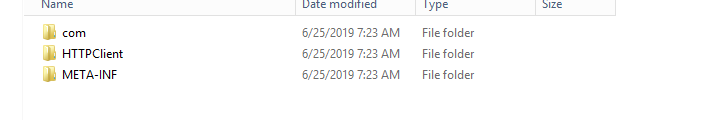
The latest code changes for API Inbound **must be** available in the repository and the same is required to be compiled in the targeted environment using the “fsInstaller”.

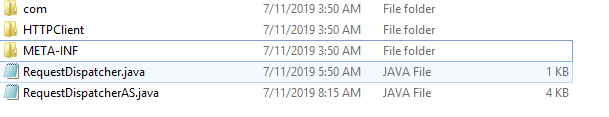
The “fsInstaller” compiles the code base and copies the compiles program to “Programs” folder.

**Step 3: Java Folder Setup**

* A folder named “java”(under Programs\api\node) will be created as the result of “fsInstaller” installer run.
* **Open Client Java OpenAPI** is required to enable making AppServer calls and running Progress procedures from a java class. Progress provides the required libraries as part of Progress OpenEdge installation. A java archive file named **o4glrt.jar** can be located under Progress Installation Folder as below:



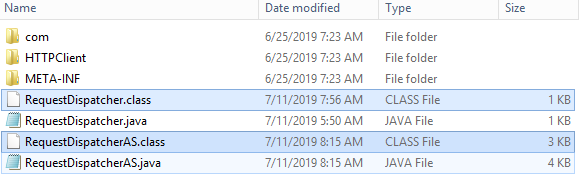
* Copy the required java packages from **<Progress Installation Folder>\java** into the folder Programs\api\node\java following the below steps:
* Go to the path **<Progress Installation Folder>\java** and extract the contents from **o4glrt.jar** into Programs\api\node\java folder and verify that the following sub-folders are available under Programs\api\node\java 
* Compile java programs as per the steps given below:
* Java programs “RequestDispatcherAS.java” and “RequestDispatcher.java” (under Programs\api\node\java) will be created as result of “fsInstaller” installer run as shown in below screenshot.



* Open command prompt
* Change current directory to Programs\api\node\java
* Execute below commands to compile java programs

|  |
| --- |
| **<Progress Installation Folder>**\jdk\bin\javac RequestDispatcher.java  **<Progress Installation Folder>**\jdk\bin\javac RequestDispatcherAS.java |

**Note:** If the java programs are compiled successfully, then files “RequestDispatcherAS.class**”** and **“**RequestDispatcher.class**”** should get generated in the same folder Programs\api\node\java as highlighted in the screen shot below:

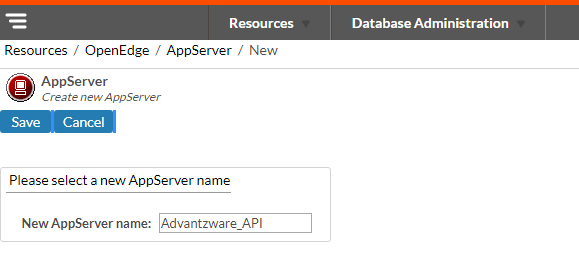


* **Once the .class files are generated, delete the .java files (“RequestDispatcherAS.java” and “RequestDispatcher.java”) from Programs\api\node\java folder.**
* Make sure “logs” folder under Programs\api\node\java is available. A CSV file will be generated with the failed requests data which can be later imported into APIInboundEvent table and processed further when AppServer/Progress Layer is down.

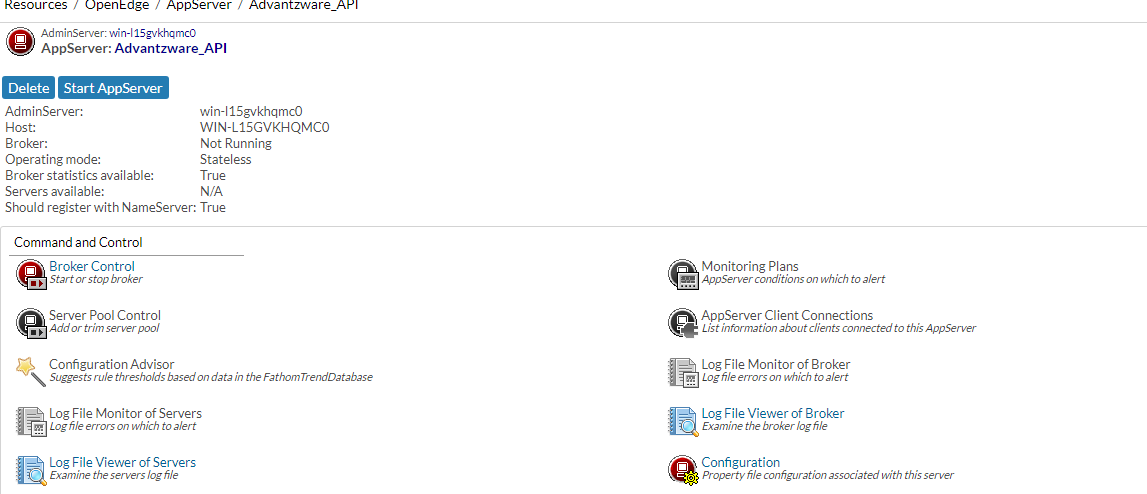
**Step 4: AppServer Setup for Development and Test Environments**

**Please note – this broker is ONLY required for Development and Test Environments – this is assumed that there is an existing AppServer broker available in Prod environment and the same can be used in node start program – please refer to the steps in this document under Node Server Setup section for more details.**

* Open Management Console
* Create new App Server broker **Advantzware\_API**, if not already available



* After creation of AppServer broker **Advantzware\_API**, click on “**Configuration**” to setup the AppServer.



* Create a .pf file to setup the databases (ASI and AUDIT) and server start up parameters

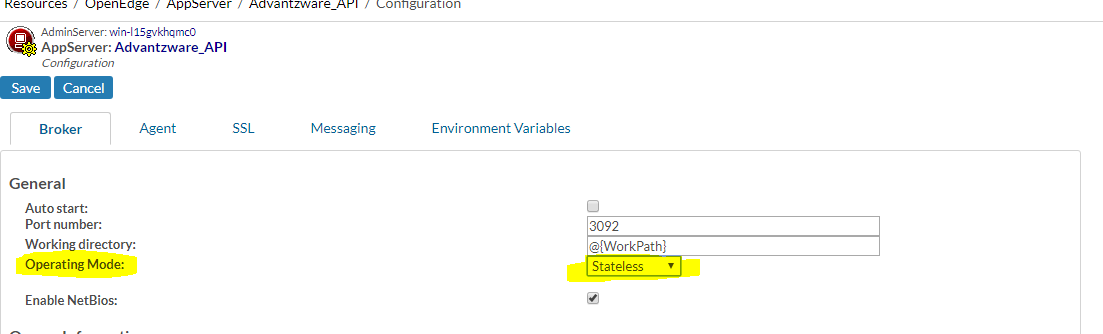
-s to 200 and -strictEntityResolution to 0 to setup in **Advantzware\_API** broker

A Sample code for .pf file creation is shown below:

The tags <path> , <asi-db-name> and <audit-db-name> are to be replaced with the appropriate values.

|  |
| --- |
| -db <path>\Prod\<asi-db-name>.db  -H localhost  -S 2821  -ld ASI  -cpinternal ISO8859-1  -cpstream ISO8859-1  -cpcoll Basic  -cpcase Basic  -d mdy  -numsep 44  -numdec 46  -db <path>\Audit\<audit-db-name>.db  -H localhost  -S 2831  -ld AUDIT  -cpinternal ISO8859-1  -cpstream ISO8859-1  -cpcoll Basic  -cpcase Basic  -d mdy  -numsep 44  -numdec 46  -s 200  -strictEntityResolution 0 |

* Go to the **Broker** tab to setup **“Operating Mode”** as **“Stateless”** as shown below

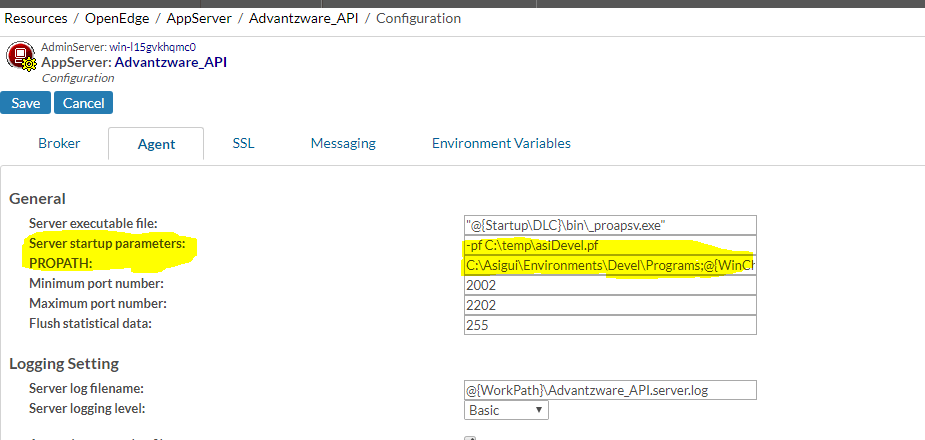


* Go to the **Agent** tab to setup the server startup parameters and the PROPATH
* Setup .pf file in the **server startup parameters** field and application’s PROPATH in the **PROPATH** field

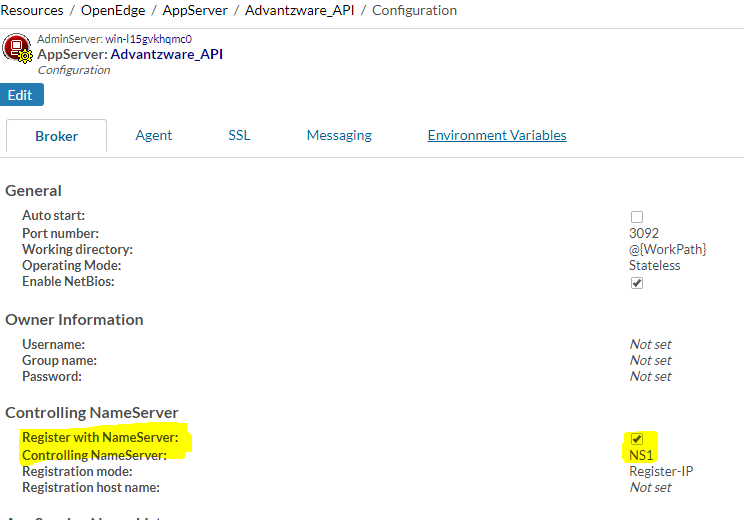
Sample setup of AppServer startup parameters and the PROPATH is shown below

Add the below path to the PROPATH field as shown below:

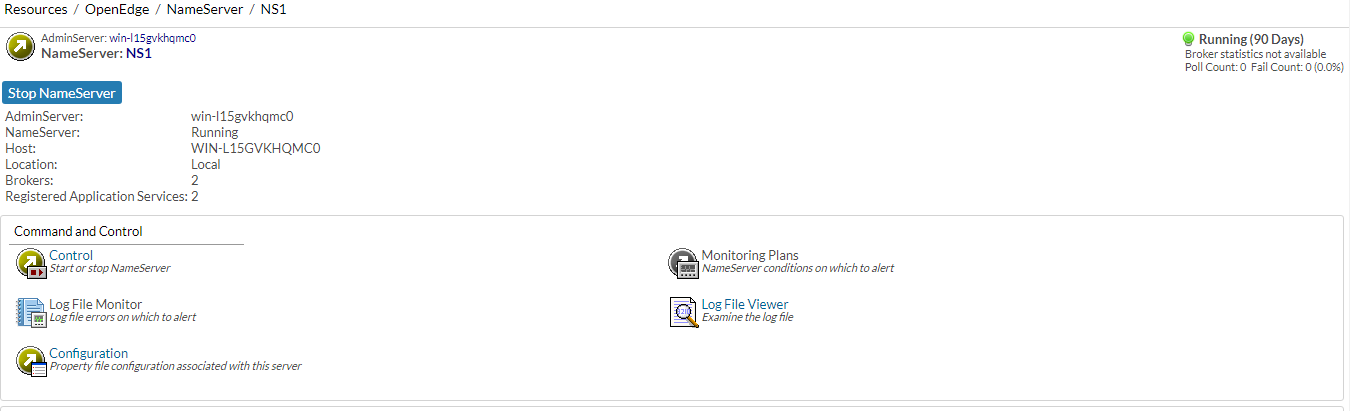
C:\Asigui\Environments\Devel\Programs



* Save the configuration
* Now, go to **Broker** tab and check whether any **NameServer** is registered or not.

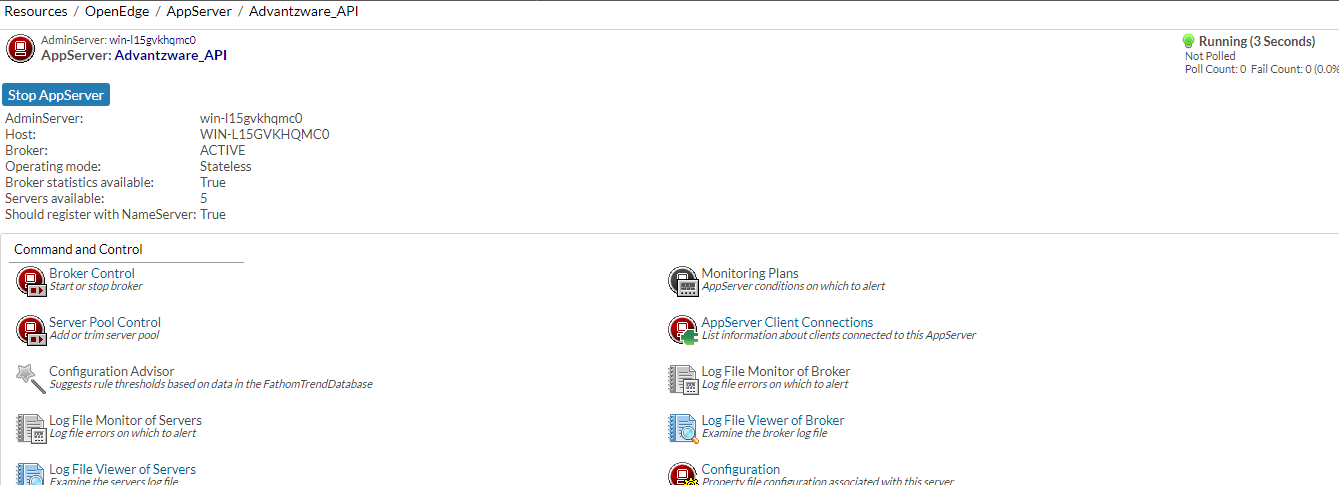


* If there is no registered NameServer,then check for any available NameServer and register it with AppServer
* If there are no available NameServers, then create new NameServer and register it with AppServer
* Now start registered NameServer if it is not running.
* If registered NameServer started without any issues, we can see its status as **“Running”** as shown in the screen shot below.



**Note:** When any AppServer is created, by default AppServer registers with the NameServer **NS1**.We can register AppServer with another NameServer if required. But, registered NameServer should be running before we start AppServer

* Now, Start the AppServer **Advantzware\_API**.
* If the AppServer starts without any issues, we can see its status as **“Running”** as shown in the screen shot below.



**Step 5: Progress Programs Folder Setup**

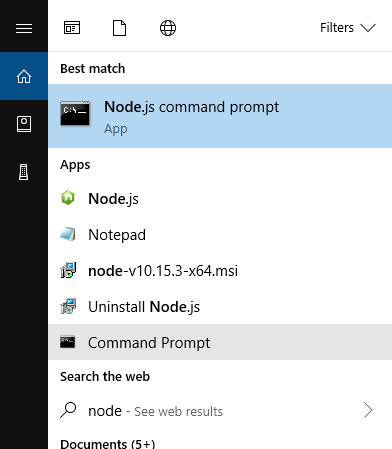
* A Folder named “inbound” (under Programs\api) will be created as result of “fsInstaller” installer run. This is where the Inbound API specific programs are available as compiled code

**Step 6: Node Server Setup**

1. **Install Node Server by using the link below:**

<https://nodejs.org/en/>

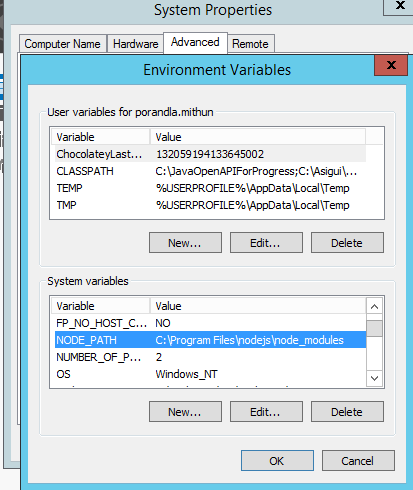
**Note:** If Node Server is installed successfully, then its icons should appear in windows search as shown below.

****

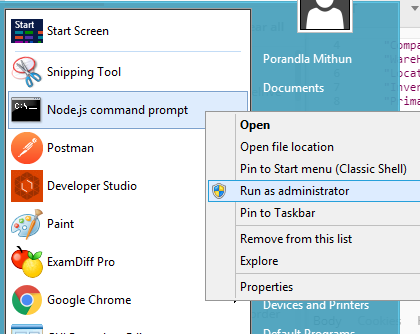
1. **Add NODE\_PATH in system environment variables:**

NODE\_PATH: <path where node is installed>\node\_modules

**Example**: C:\Program Files\nodejs\node\_modules



1. **Run Node.js command prompt in admin mode**

****

1. **Go to NODE\_PATH by executing below command**

|  |
| --- |
| cd %NODE\_PATH% |

1. **Install Required Node modules:**

* Open command prompt
* Run below commands in the same order as given below:

|  |
| --- |
| npm install --save express  npm install --save body-parser  npm install --save body-parser-xml  npm install --save node-jre  npm install --save csv-write-stream  npm install --save fs  npm install --save xml  npm install --save xmldom |

1. **Set environment variables as below:**

**The environment variables should not be enclosed in double quotes(“)**

|  |  |  |
| --- | --- | --- |
| **Environment variable** | **Value** | **Purpose** |
| NODE\_ENV | Production/Development | Set “Production” for production environment and “Development” for development environment |
| API\_JAVA\_PROGRAMS\_DIR | java | Path for java programs |
| API\_JAVA\_LOGS\_DIR | java\logs | Path where requests are logged to a CSV file when the AppSever is down. This file can be loaded using a program and such inbound API requests can be executed outside Node |
| API\_JAVA\_PROCESSOR | RequestDispatcher | The name of the java program which dispatches the requests from Node to AppServer |
| API\_APPSERVER\_URL | AppServer://<IPAddress>:<NameServer port number>/ <AppServer broker name> | URL to Progress AppServer |
| API\_APPSERVER\_REQUEST\_ROUTER | api/inbound/APIRequestRouterAS.p | A Progress program which runs behind AppServer and makes calls to API specific process hanlders |
| API\_IP\_ADDRESS | <IPAddress> | The IP address of the machine to be used by Node server |
| API\_PORT | <PORT> | The port to be used by Node server |

**Please note:**

* Value for the environment variable **API\_APPSERVER\_URL** should be updated with:
* AppServer broker and NameServer port number as configured in the Step 4.
* If AppServer and Node are running on the same machine, then update IPAddress with "localhost"
* If AppServer and Node are running on different machines, then update “IPAddress" with "IPAddress of the machine on which AppServer is running”
* Value for the environment variable **API\_IP\_ADDRESS** should be updated with “IP address of the machine” on which Node needs to run
* Value for the environment variable **API\_PORT** should be updated with PORT on which Node needs to listen for incoming requests

**6. Start Node Server:**

* Open command prompt.
* Change current working directory to Programs\api\node folder
* Run below command

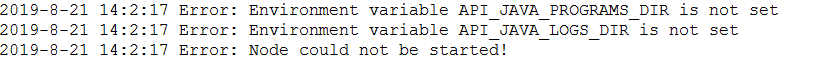
|  |
| --- |
| node InboundAPIStart.js >> node.server.log 2>> node.server.error.log |

**Note:**

* If Node starts without any errors, “node.server.log” file will have the text as shown below. Here, values for the appserverURL, serverIP and serverPort will change based on the updated values in environment variables

## 

* If environment variables are not set properly, then “node.server.log” will get the below errors and Node server stops. Here, errors may change based on the number of invalid environment variables



* If there is any problem with Node server start, then “node.server.error.log” gets update with the error.
* Move node folder from “Programs\api\inbound” to “CustFiles” folder of the respective environment

**How to test Inbound APIs**

1. Inbound API “**getinventory**”:

* Postman Setup
* Launch Postman
* Select request type as **POST**
* Enter “http://<IPAddress>:<PORT>/api/getinventory” as API URL

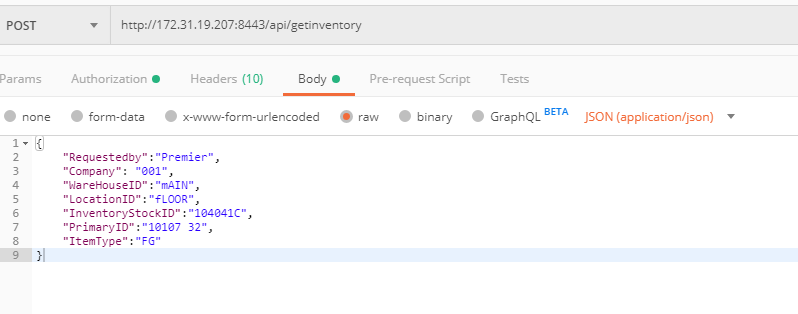
Here, IPAddress and PORT should be same as provided in the environment variables (Programs\api\node\config.js)

In the **Step 6.6**

* Go to “Authorization” tab and select Type as **Basic**
* Enter username and password
* Go to “Body” tab
* Enter below sample **JSON** text as request data

|  |
| --- |
| {  "Requestedby":"Premier",  "Company": "001",  "WareHouseID":"mAIN",  "LocationID":"fLOOR",  "InventoryStockID":"104041C",  "PrimaryID":"10107 32",  "ItemType":"FG"  } |

* Select “raw” and “JSON” as data type as shown below



* Now click on **Send** button to call API “getinventory”
* If API “getinventory” call is successful, then below JSON text is returned as the response

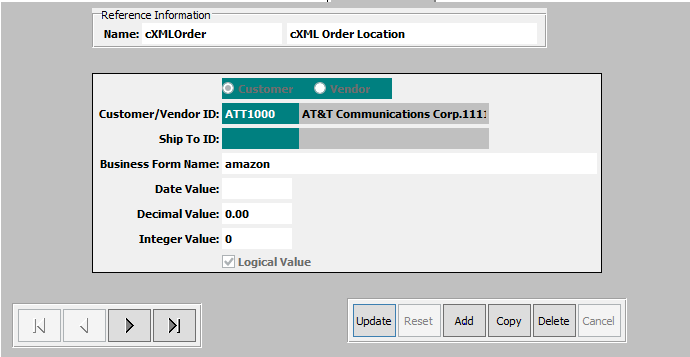


1. Inbound API “**cxmlorder**”:

* Setting up “System Control Parameter “ for “cXMLOrder”
* Go to “System Control Parameters” (N-K-1) screen
* Search for name “cXMLOrder”. Here, we are assuming that “cXMLOrder” is already created
* Create new form (sys-ctrl-shipto) by following below steps

1. Select radio button “Customer”
2. Enter “Customer/Vendor ID” field for which we tag the customer name
3. Enter customer name in “Business Form Name” field. Customer name should be same as “Identity” tag value in request XML provided by customer
4. Check “Logical Value” toggle box

Find the below sample form created for customer “amazon” linked to Customer ID “ATT1000”

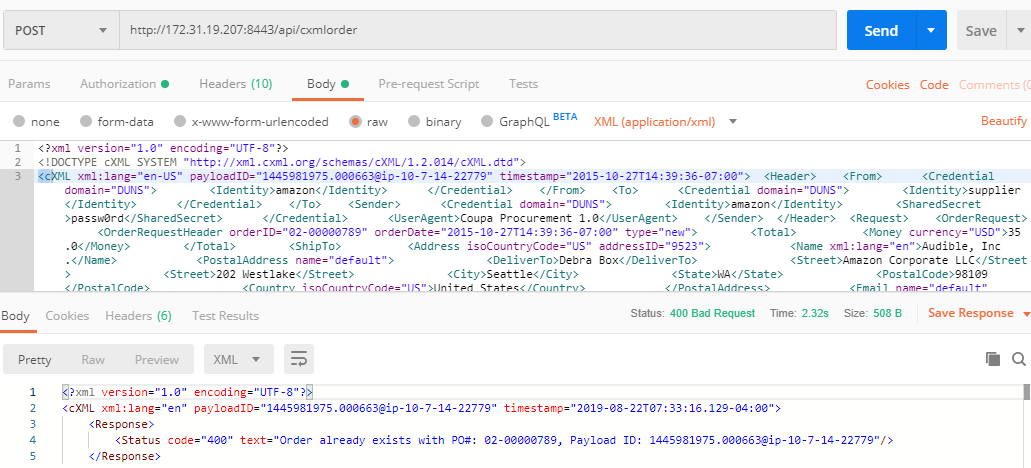


* Modify “XML request data” provided by customer as below.
* Modify “SupplierPartID” tag value in request XML with any valid item for the customer.
* Modify “orderID” attribute in “OrderRequestHeader” tag with a unique value for each call
* Setup Postman
* Launch Postman
* Select request type as **POST**
* Enter “http://<IPAddress>:<PORT>/api/cxmlorder” as API URL

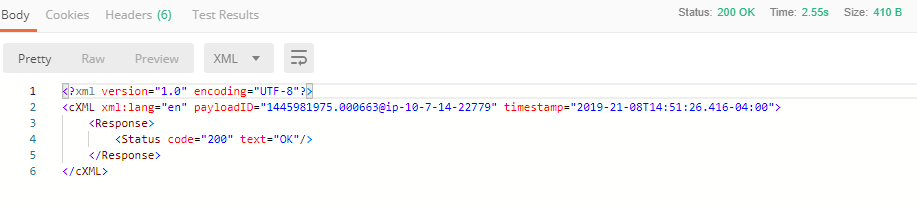
Here, IPAddress and PORT should be same as provided in the environment variables (Programs\api\node\config.js)

In the **Step 6.6**

* Go to “Authorization” tab and select Type as **Basic**
* Enter username and password
* Go to “Body” tab and paste modified request cXML
* Select “raw” and “XML” as data type as shown below



* Now click on **Send** button to call API “cxmlorder”
* If API “cxmlorder” call is successful, then below XML is returned as the response



* Order created can be checked in **“Order Entry” (O-U-1)** screen as show below

