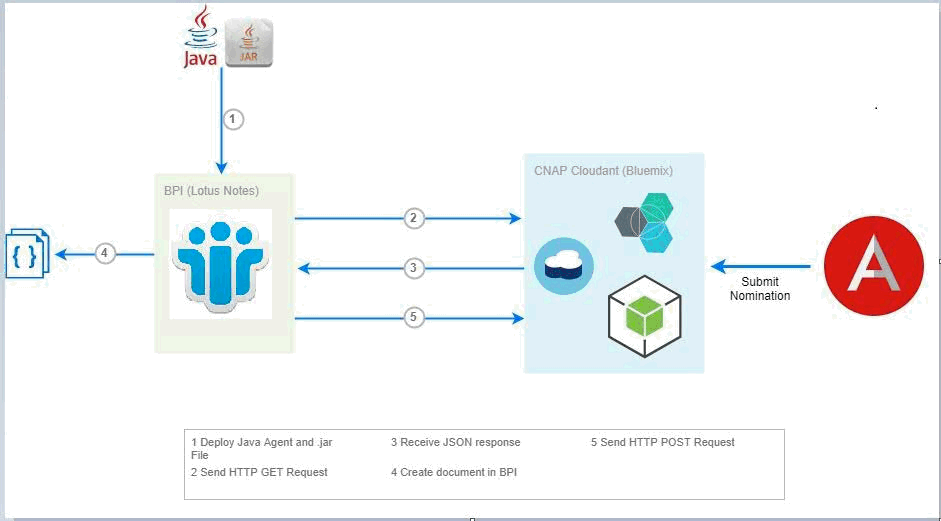
**Solution for receiving HTTP JSON GET/PUT request against a RESTful URL Cloudant data source from Lotus Notes/Domino and create document in Lotus Notes database using returned JSON data**

This solution can be used by any Lotus Notes/Domino application to send HTTP GET/PUT request to RESTful URL Cloudant data source. It returns a JSON which can be used to create document in Lotus Notes/Domino database. This solution can be very useful for Lotus Notes applications which are planning to be re-written in modern technology and deploy in Bluemix.

Here is the high-level Architecture diagram of the solution,



**Step 1: Create Cloudant Read API and generate API Key**

Login to “bluemix.cloudant.com”. Go to “Permissions” tab and Generate API Key for a Username. Grant “\_reader” access to that user. Note down the Username and API Key. It will be used for sending HTTP request.

**Step 2: Create View in Cloudant Design Documents (Optional)**

Creating a view is optional. If you filter documents in a view and send HTTP request to fetch document from the view instead of database, response will be faster.

Open your Cloudant database. Click on “Design Documents” tab and select “New View”.

Name Design Document and give an Index Name. In the Map Function, write a JavaScript function to filter the documents based on the condition. Here’s a sample function,

function (doc) {

if(doc.Form == "ProgramDocv60")

emit(doc.\_id, doc.Form);

}

This function will filter the documents which has “Form = ProgramDocv60”. You can write your own Map function. Then click on “Save Document and then Build Index”. It will create the view.

**Step 3: Add external jar file**

Download “java-json.jar” file and add it to “C:\Notes\jvm\lib\ext” path. Restart your system.

**Step 4: Create Java agent in Lotus Notes/Domino database**

Open your database in designer and create a Java agent. Here’s the agent code,

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**import** java.util.Base64;

**import** org.json.JSONObject; // External jar to work with JSON objects in Java

**import** lotus.domino.\*;

/\*\*

\* Method for receiving HTTP JSON GET request against a RESTful URL data source.

\* **@author** KirtiJha/India/IBM

\* **@param** url the URL of the REST endpoint

\* **@return** JsonObject containing the data from the REST response.

\*/

**public** **class** JavaAgent **extends** AgentBase {

**public** **void** NotesMain() {

**try** {

Session session = getSession();

AgentContext agentContext = session.getAgentContext();

// (Your code goes here)

Database db = agentContext.getCurrentDatabase();

String url = “https://<Cloudant URL>/<DB Name>/\_design/<Design Documents Name>/\_view/<View Index Name>?key=\”<Document ID>”\"&include\_docs=true";

// If you have not created view and querying the database for a specific docID, the url will be –

“https://<Cloudant URL>/<DB Name>/Document ID>"&include\_docs=true";

URL obj = **new** URL(url);

String usernameColonPassword = "<API\_KEY>:<PASSWORD>";

String basicAuthPayload = "Basic " + Base64.*getEncoder*().encodeToString(usernameColonPassword.getBytes());

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

// HTTP GET Request

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", "Mozilla/5.0");

con.setRequestProperty ("Authorization", basicAuthPayload);

**int** responseCode = con.getResponseCode();

System.***out***.println("\nSending 'GET' request to URL : " + url);

System.***out***.println("Response Code : " + responseCode);

BufferedReader in = **new** BufferedReader(**new** InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = **new** StringBuffer();

**while** ((inputLine = in.readLine()) != **null**) {

response.append(inputLine);

}

in.close();

// This highlighted code is to reformat the JSON response to get the required Keys. This may change depending upon the HTTP response. The goal is to reach to “doc” JSON Object.

JSONObject myResponse = **new** JSONObject(response.toString());

JSONObject myResponse1 = **new** JSONObject(myResponse.getJSONArray("rows").get(0).toString());

JSONObject myResponse2 = **new** JSONObject(myResponse1.get("doc").toString());

// Create document in Lotus Notes database

Document newNotesDoc = db.createDocument();

newNotesDoc.replaceItemValue("Form", "IBMForm"); // Lotus Notes Form

newNotesDoc.replaceItemValue("docID", myResponse2.get("\_id"));

newNotesDoc.replaceItemValue("user\_program", myResponse2.get("User\_Programs"));

newNotesDoc.replaceItemValue("status", myResponse2.get("StatusStr"));

newNotesDoc.replaceItemValue("aq\_code", myResponse2.get("AQCode"));

newNotesDoc.computeWithForm(**true**, **false**);

newNotesDoc.save(**true**, **true**);

db.recycle();

// HTTP PUT Request to Cloudant to Update document

URL postUrl = **new** URL(“https://<Cloudant URL>/<DB Name>/<Document ID>”);

HttpURLConnection conn = (HttpURLConnection)postUrl.openConnection();

conn.setRequestMethod("PUT");

conn.setRequestProperty("Content-Type", "application/json; utf-8");

conn.setRequestProperty("Accept", "application/json");

conn.setRequestProperty("Authorization", basicAuthPayload);

conn.setDoOutput(**true**);

// Send \_rev and document JSON in PUT request to Update (Sending \_rev is mandatory)

myResponse2.put("\_rev", revID);

System.***out***.println(docResponse);

OutputStreamWriter json = **new** OutputStreamWriter(conn.getOutputStream());

json.write(myResponse2.toString());

json.flush();

json.close();

System.***out***.println("\nSending 'PUT' request to URL : " + postUrl);

} **catch**(Exception e) {

e.printStackTrace();

}

}

}

**Step 5: Add external jar file in the agent**

Click on “Project” tab in designer and go to Properties. Then go to Java Build Path and click on Libraries tab. Then click on “Add external JARs” and select “java-json.jar” file from “C:\Notes\jvm\lib\ext” path. Build project.

**Step 6: Run agent**

Run this agent from your Lotus Notes database either using a button or Action Menu. If successful, It should create a document with the Form specified in the agent and populate the corresponding field values. Please make sure the form and fields exist in lotus notes database.

**Step 7: Schedule agent to run on server**

To schedule this agent on server, follow these steps

**7a:** Change the agent schedule to run on the server. From agent properties, go to Securities tab and make sure to have a runtime security of "2. Allow restricted operations.

**7b.** In the Server Document from Domino Administrator, make sure the signer of the agent has permission to "Run unrestricted methods and operations" and "Run restricted LotusScript/Java agents"

**7c.** Copy “java-json.jar” file to “C:\Program Files (x86)\ibm\Domino\jvm\lib\ext” path on the server. Restart Server and Domino services.

**7d.** Enable agent.