IBM Notes to MongoDB Data Migration

With the divestiture of IBM Notes to HCL, it’s a need of this moment to move existing IBM Notes applications to some other platform. There are several options to choose from, but MEAN (MongoDB-Express-Angular-NodeJs) stack is the most appropriate choice because of following reasons,

1. MongoDB is the most popular NoSql database which offers several benefits like scalability, indexing, faster processing, Compass and Atlas etc.
2. NodeJS is JavaScript runtime which runs on Chrome V8 engine which is fast and secure.
3. Angular is full featured JavaScript framework to design feature rich UI.

In every rewrite or redesign project, the most critical part is the existing Data. Data retention, processing and migration are the steps which are critical and required for every application. IBM Notes (Lotus Notes) being a legacy tool with very small community, there are no or very less options available (as per my knowledge and R&D) to migrate data to MongoDB. There are 3rd party tools available, but that is costly and there is a risk of exposing confidential data to some 3rd party which is not secure.

We are redesigning our existing IBM Notes applications to MEAN stack and we faced these challenges during Data migrations. To overcome this, I have developed this solution that can easily migrate data from IBM Notes to MongoDB. This solution offers several benefits,

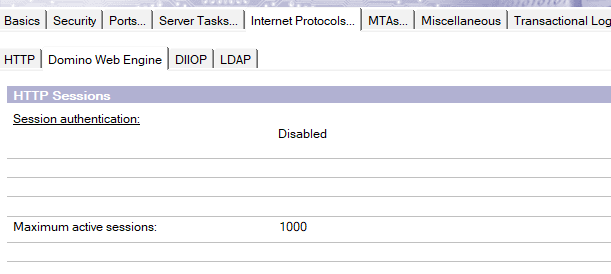
1. It’s easy to implement and it can be used with any IBM Notes database.
2. It’s based-on HTTP and REST, so its fast and secure.
3. Its written in Python which has excellent set of libraries to work with Data (Data Cleaning, Data Processing etc).
4. We are getting and processing data in JSON and its suitable for Python and MongoDB.
5. Most importantly, once you are out of IBM Notes it will give you lot of options and opportunities to scale your application using modern tools. You can implement DevOps, TDD, AI and many others (Opportunities are endless).

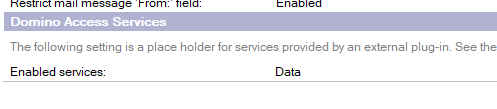
This solution is in few steps,

**Enable Domino Data Services in Domino server**

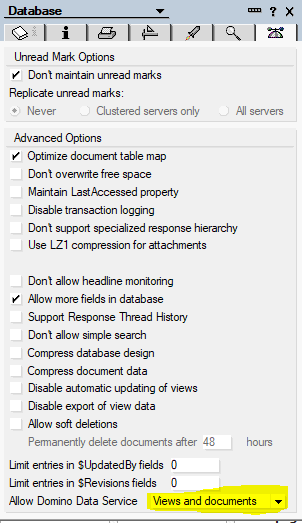
You need to enable Domino Data Service on your Domino server so that you can access your database with Domino API.

Open server in Domino Administrator. Go to “Configuration🡪Current Server Document🡪Internet Protocols🡪Domino Web Engine”. In “Domino Access Service” section, select “Data” under Enabled Services and Save.

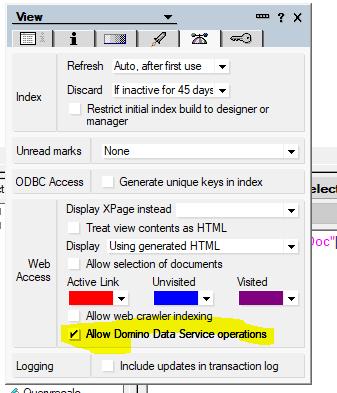




Go to Database properties and in the last tab, select “Views and documents” under Allow Domino Data Service,



Open the view which you need to migrate in Designer and in second last tab of View Properties, check “Allow Domino Data Service operations” under Web Access,



**Prepare View and CSV Data**

Next step is to prepare the data you need to migrate. We will access Document using Domino API and we will target each document based on its Document Unique ID (UNID). To get UNID, make sure to create a column in the view (preferably 1st column) with this formula,

@Text(@DocumentUniqueID)

This will populate document unique ID of every document in the view in that column. Once its done, Open this view in Notes client and Export it in a CSV file using File🡪Export option and save the file.

**Migrate data to MongoDB (Python Script)**

I have written this Python script which sends HTTP request to Domino API URL and get back notes data in JSON. Once JSON is received, you can do data cleaning or reformatting based on your requirement. Some common processes are,

1. Change multivalued field data in Domino to Array (List in Python)
2. Change Date values
3. Add any missing or new data

Most important and complicated part is to migrate Rich Text field data. Notes Rich Text fields can store several types of values like Attachments, in-line Images, Texts, Embedded Objects etc. MongoDB stores these types of data as Binary. So, you need to convert these data to base64 and then send it to MongoDB. In the attached script I have provided some sample codes to demonstrate some of these conditions, but finally it depends on the actual data that you have and the data that you need to store in MongoDB. Based on these, you can modify these codes.

I have used Pycharm Community Edition IDE to write and run the script. It’s a popular IDE for Python. But you can use any IDE which has Python support.

I am using Pymongo library to connect and work with MongoDB. It’s the official MongoDB library for Python. In addition to these I have used other libraries like urllibe3, Requests, BeautifulSoup, json, datetime etc to get, process and send the data. You can get more information about these packages and installation by following their documentations,

Here is the Python script to migrate data,

**import** json  
**import** urllib3  
**import** pymongo  
**import** csv  
**import** datetime  
**from** bs4 **import** BeautifulSoup  
  
documents = []  
  
**with** open(**'Mongo\_Migration\_Data\_Dev.csv'**, **'r'**) **as** csv\_file: # Provide path of exported CSV file   
 csv\_reader = csv.reader(csv\_file, delimiter=**','**)  
 **for** row **in** csv\_reader:  
 notes\_unid = row[0] # Get the row with UNID value (1st row in my case)  
 documents.append(row[0]) # Get all the UNID’s and store in List  
  
**for** document **in** documents[0:len(documents)]: #Loop through all the UNID’s  
 notes\_unid = document  
 http = urllib3.PoolManager()  
 headers = urllib3.util.make\_headers(  
 basic\_auth=**'<user\_name>:<password>'**) # Username and Password to access domino server  
 notesURL = **"<Database Path>/api/data/documents/unid/"** + notes\_unidr = http.request(**'GET'**, notesURL, headers=headers) # Send HTTP request  
 r\_json = json.loads(r.data.decode(**'utf-8'**))  
 **for** element **in** [**'@href'**, **'@authors'**, **'@created'**, **'@modified'**, **'@form'**, **'@noteid'**]: # These are notes specific fields. You can remove them if you want. Not needed in MongoDB   
 r\_json.pop(element)  
 propertyMap = [**'Brand'**, **'StgFedCWMWeb'**, **'brandTierDistributorDetails'**,  
 **'Competitors'**, **'StatusHist'**] # Multivalue fields  
 comments = [**'BPSOComments'**, **'SSMComments'**, **'BDEComments'**, **'AdminComments'**] # Array of Objects  
 **for** (key, value) **in** r\_json.items():

# Change multivalued fields data to List  
 **if** key **in** propertyMap **and** isinstance(value, str):  
 value = list(value.split(**','**))  
 r\_json[key] = value

# Create Array of Objects  
 **elif** key **in** comments **and** isinstance(value, str):  
 r\_json[key] = [  
 {  
 **"logdate"**: datetime.datetime.now().replace(microsecond=0).isoformat() + **'Z'**, # Format Date based on your requirement  
 **"logmessage"**: value,  
 **"user"**: **"Admin"** }  
 ]  
 r\_json[**'notesDocID'**] = r\_json.pop(**'@unid'**)  
 r\_json[**'Form'**] = **'nomination'** logdate = datetime.datetime.now().replace(microsecond=0).isoformat() + **'Z'** logmessage = **"Migrated from CNAP Notes by the"** user = **"Admin"** auditTrailData = [  
 {  
 **"logdate"**: logdate,  
 **"logmessage"**: logmessage,  
 **"user"**: user  
 }  
 ]  
  
 r\_json[**'auditTrailData'**] = auditTrailData  
 r\_json[**'Comments'**] = **"Migrated from Notes"**

**# This section processes the Rich Text field data i.e Attachemnts, in-line images, text, embedded objects etc** appssm = r\_json[**'appssm'**]  
 cont = appssm.get(**'content'**)  
 **for** item **in** cont:  
 **if** item.get(**'contentType'**) == **'image/jpeg'**:  
 image\_data = item.get(**'data'**)  
 image = **f'<p><img style="width: 539px;" src="data:image/png;base64,{**image\_data**}" data-filename="image.png"><br></p>'** r\_json[**'appssm'**] = image  
 **elif** item.get(**'contentType'**) == **'text/html; charset="US-ASCII"'**:  
 ssm\_text\_data = item.get(**'data'**)  
 r\_json[**'appssm'**] = ssm\_text\_data  
  
 revenue = r\_json[**'RevenueHistory\_1'**]  
 cont\_revenue = revenue.get(**'content'**)  
 revenue\_image = []  
 span\_data = []  
 revenue\_data = []  
 **for** item **in** cont\_revenue:  
 **if** item.get(**'contentType'**) == **'text/html; charset="US-ASCII"'**:  
 text\_data = item.get(**'data'**)  
 soup = BeautifulSoup(text\_data, **'html.parser'**)  
 spans = soup.find\_all(**'span'**)  
 **for** span **in** spans:  
 span\_data.append(span.text)  
 **elif** item.get(**'contentType'**) == **'image/gif'**:  
 revenue\_data.append(item.get(**'data'**))  
 revenue\_image\_data = [(span\_data[i], revenue\_data[i]) **for** i **in** range(len(revenue\_data))]  
 **for** rid **in** revenue\_image\_data:  
 revenue\_image.append(  
 **f'<span style=" font-size:8pt;font-family:sans-serif">{**rid[0]**}</span> <br><p><img style="width: 539px;" src="data:image/png;base64,{**rid[1]**}" data-filename="image.png"><br></p>'**)  
  
 r\_json[**'RevenueHistory\_1'**] = revenue\_image

#Connect to MongoDB and Insert data  
 client = pymongo.MongoClient(  
 **'<MONGO\_URI>'**,  
 ssl=**True**, # If SSL certificate is required  
 ssl\_ca\_certs=**'<SSL\_CERT\_PATH>'**)  
 db = client[**'<MONGO\_DATABASE\_NAME>'**]  
 collection = db.<COLLECTION\_NAME>  
 result = collection.insert\_one(r\_json)  
 print(**f'Notes document with UNID {**notes\_unid**} is migrated to MongoDB'**)