TEKsystems
Own change

TECHNICAL WHITE PAPER

Data Integration With REST API

Real-time or near real-time accurate and fast retrieval of key metrics is a critical need for an organization. Many times, valuable data are stored in siloed systems with zero or incomplete integration with other critical systems, thereby leading to redundancy, inconsistency and confusion when reporting on the data. This also requires allocating expensive person-hours to manually collect, clean, aggregate and report the key metrics, resulting in valuable time lost accessing critical data for key decisions. Maintenance and reporting costs and complexity increase as data keep growing if no archival strategy is in place.

Our use case details an implementation that was successfully performed to resolve all of these challenges.

Benefits achieved

- By creating an interactive dashboard for the CIO and executives of the organization, both flexibility and standardization were achieved—a key factor in driving sound business decisions.
- The integration and transformation processing automation has increased productivity by eliminating human intervention and decreasing the risk of human error.
- By initially landing data in the data lake, the organization has lowered the cost of storing historical data and created a flexible area to conduct expiration for analytics and data science workloads.

Architecture

The below diagram depicts the logical architecture of the implementation.

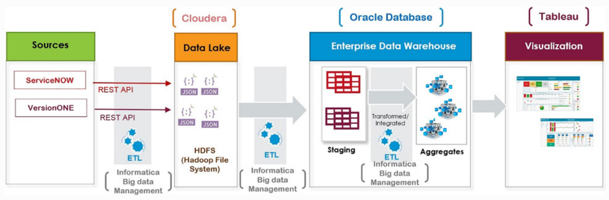


Figure 1: REST data integration architecture

Data resides in multiple applications, both cloud-based and local (file systems/databases). The strategy used for cloud-based sources was Representational State Transfer (REST) API. For local source systems, direct connections were used. This article focuses on the cloud-based source data extraction.

Data flow

1. Data is pulled from sources in a predefined frequency by invoking respective REST API routines. Informatica Java transformation in Informatica is used for this purpose.
2. The data are then stored in Cloudera data lake in native format.
3. The relevant data are then pulled into the enterprise warehouse and transformed/ aggregated for analytical purposes.
4. Tableau connects to the analytics warehouse and uses the aggregated data in the dashboards.

An overview of REST API

REST API, also known as RESTful APIs, is a popular type of API. An advantage of REST API is that it uses existing protocols like HTTP; therefore, no additional software is needed for its creation.

Another advantage is its flexibility. It can handle multiple types of calls and return data in different formats.

Roy Thomas Fielding originated REST architecture and communicated six constraints that form the basis of the RESTful style:

1. **Client-Server.** The client and the server should be separate from each other and allowed to evolve independently.
2. **Stateless.** REST APIs are stateless; requests can be made independently of one another. Each request

© 2019 TEKsystems, Inc. ALL RIGHTS RESERVED.

from the client will contain all the required data for the server to process the request.

3. **Cache.** Because a stateless API can increase request overhead by handling large loads of incoming and outbound calls, a REST API should be designed for cacheable data. This improves network efficiency, as a client cache can reuse the response data for future similar requests if needed.

4. **Uniform interface.** The key to decoupling the client from the server is having a uniform interface that allows independent evolution of the application without having the application's services, or models and actions, tightly coupled to the API layer itself.

5. **Layered system.** REST APIs have different layers of their architecture working together to build a hierarchy, which helps create a more scalable and modular application.

6. **Code on Demand.** Code on Demand allows for code or applets to be transmitted via the API for use within the application.

ETL implementation

The sequence (four steps) of ETL implementation using Informatica BDM is explained. The first two steps are relevant to this article; therefore, only those are covered in detail.

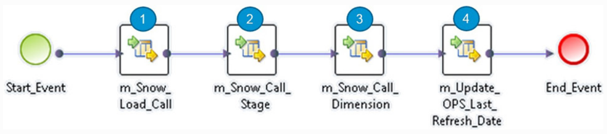


Figure 2: ServiceNow dataset workflow

Step 1: Place the API request to ServiceNOW

Step 2: Load the response data from step 1, which will be in JSON format, into Cloudera (data lake) in the form of a flat file, and parse the JSON contents for loading to a database staging table

Step 3: Load the data from staging table to dimension/fact tables

Step 4: Refresh the ETL configuration data for incremental load

The maps for a ServiceNOW dataset

1. **The Java_Snow_Call map with relevant properties**

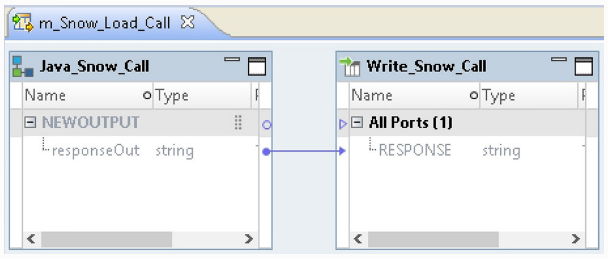
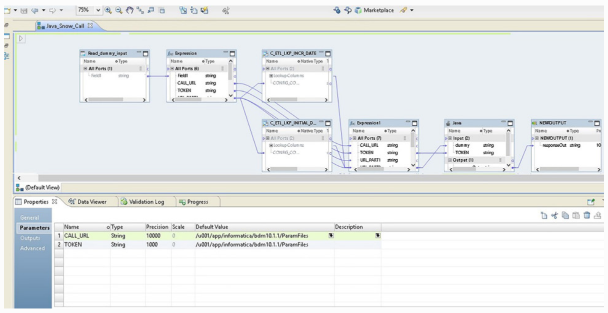
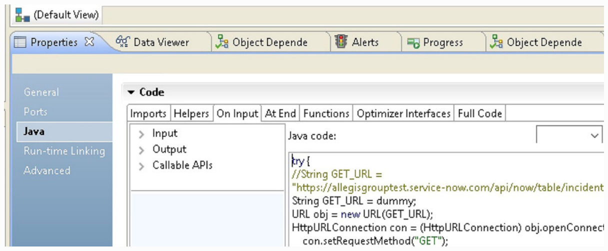


Figure 3: Informatica Developer Java_Snow_Call to Write_Snow_Call



	Name	Type	Precision	Scale	Default Value	Description
1	CALL_URL	String	10000	0	/u001/app/informatica/bdm10.1.1/ParamFiles	
2	TOKEN	String	1000	0	/u001/app/informatica/bdm10.1.1/ParamFiles	

Figure 4: Java_Snow_Call map with Properties



```
try {
//String GET_URL =
"https://allegisgrouptest.service-now.com/api/now/table/incident
String GET_URL = dummy;
URL obj = new URL(GET_URL);
HttpURLConnection con = (HttpURLConnection) obj.openConnect
   con.setRequestMethod("GET");
```

Figure 5: Java code

2. **The map that loads the response from the REST API to the data lake and staging table of the data warehouse**

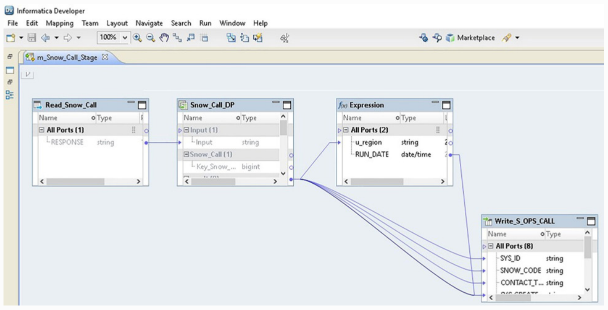


Figure 6: m_Snow_Call_State

© 2019 TEKsystems, Inc. ALL RIGHTS RESERVED.

Java code snippet

Code that invokes the REST API

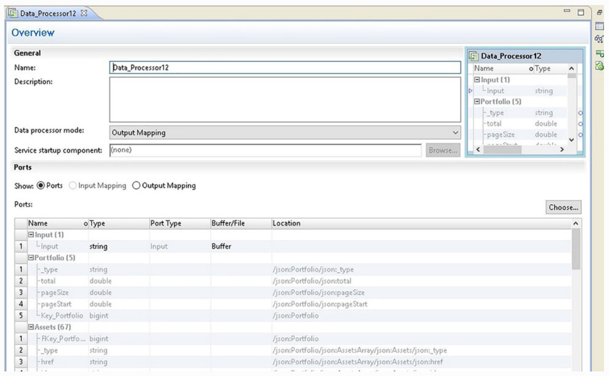
```
// Start of 'On Input Row' code snippet.

try {
String GET_URL = "https://xxxxx.service-now.com/api/now/table/incident?sysparm_limit=2";
URL obj = new URL(GET_URL);
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
con.setRequestProperty("Authorization", TOKEN);
con.setRequestProperty("Accept", "application/json");
int responseCode = con.getResponseCode();
System.out.println("GET Response Code :: " + responseCode);
if (responseCode == HttpURLConnection.HTTP_OK)
    {//success
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())        );
    String inputLine;
    StringBuffer response = new StringBuffer();
    while ((inputLine = in.readLine()) != null)
    {
    response.append(inputLine);
    }
    in.close();
    // print result
    System.out.println(response.toString());
    responseOut = response.toString();
    }
    else
    {
    System.out.println("GET request not worked");
    }
    System.out.println("GET DONE");
 }
catch (Exception e)
{
logInfo("Error");
}

// End of 'On Input Row' code snippet.
```

JSON parsing

BDM has a built-in data processor transformation that can be used to parse JSON data into key value pairs, after which data can be loaded to any relational target easily.

Steps involved:

- Obtain a sample of the JSON data that needs to be parsed. Do this for all the distinct endpoints. cURL query can be used for obtaining the JSON dataset that is the response of a REST API request.
- Create a reusable data processor transformation using the wizard.
- Select JSON as the source type, and use the sample data fetched to infer the schema for data processor transformation.



Data_Processor12

Overview

General

Name: Data_Processor12

Description:

Data processor mode: Output Mapping

Service startup component: (none)

Ports

Show: (● Ports) ○ Input Mapping ○ Output Mapping

Ports:

	Name	Type	Port Type	Buffer/File	Location
	Input (1)				
1	Input	string	Input	Buffer	
	Portfolio (5)				
1	_type	string			/json:Portfolio/json:_type
2	total	double			/json:Portfolio/json:total
3	pageSize	double			/json:Portfolio/json:pageSize
4	pageStart	double			/json:Portfolio/json:pageStart
5	Key_Portfolio	bigint			/json:Portfolio
	Assets (67)				
1	FKey_Portfo...	bigint			/json:Portfolio
2	_type	string			/json:Portfolio/json:AssetsArray/json:Assets/json:_type
3	href	string			/json:Portfolio/json:AssetsArray/json:Assets/json:href

Name	Type
Input (1)	
Input	string
Portfolio (5)	
_type	string
total	double
pageSize	double

Figure 7: Data_Processor12

- Data processor transformation has one input port that takes in the JSON string to be parsed as input. Connect the output ports from the data processor as needed to the correct downstream write transformation.

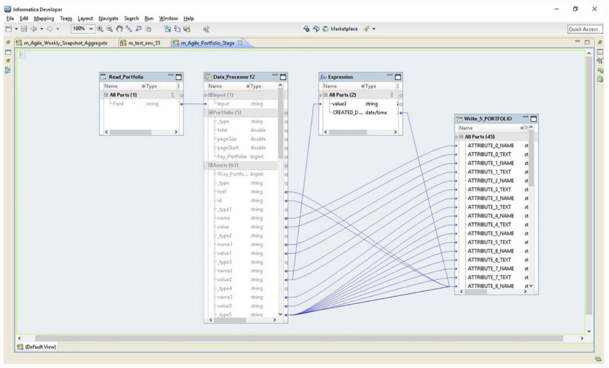
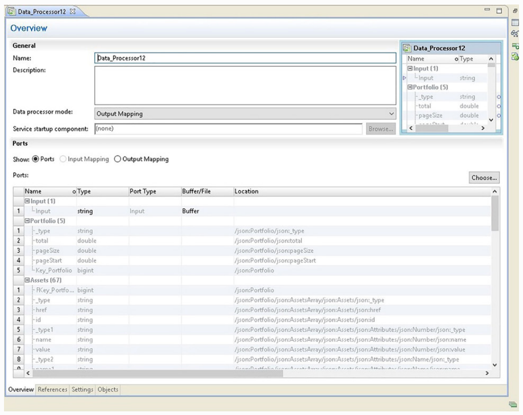


Figure 7: Data_Processor12


Data_Processor12

Overview

General

Name: Data_Processor12

Description:

Data processor mode: Output Mapping

Service startup component: (none)

Ports

Show: (● Ports) Input Mapping Output Mapping

Ports:

	Name	Type	Port Type	Buffer/File	Location
	⊟Input (1)				
1	Input	string	Input	Buffer	
	⊟Portfolio (5)				
1	_type	string			/json:Portfolio/json:_type
2	total	double			/json:Portfolio/json:total
3	pageSize	double			/json:Portfolio/json:pageSize
4	pageStart	double			/json:Portfolio/json:pageStart
5	Key_Portfolio	bigint			/json:Portfolio
	⊟Assets (67)				
1	FKey_Portfo...	bigint			/json:Portfolio
2	_type	string			/json:Portfolio/json:AssetsArray/json:Assets/json:_type
3	href	string			/json:Portfolio/json:AssetsArray/json:Assets/json:href
4	id	string			/json:Portfolio/json:AssetsArray/json:Assets/json:id
5	_type1	string			/json:Portfolio/json:AssetsArray/json:Assets/json:Attributes/json:Number/json:_type
6	name	string			/json:Portfolio/json:AssetsArray/json:Assets/json:Attributes/json:Number/json:name
7	value	string			/json:Portfolio/json:AssetsArray/json:Assets/json:Attributes/json:Number/json:value
8	_type2	string			/json:Portfolio/json:AssetsArray/json:Assets/json:Attributes/json:Name/json:_type

Figure 8: m_Agile_Portfolio_Stage

Tips, tricks and troubleshooting

During the initial stage of exploring the REST APIs of the respective applications, there can be a few potential hurdles:

1. **The URL/URI format.** Framing the correct base URL. The respective application's API documentation will provide this information.

2. **Authentication method.** Not all applications support single sign-on (SSO) when it comes to authentication. REST API calls may need a local user (application user) or an authentication token tagged to a local user.

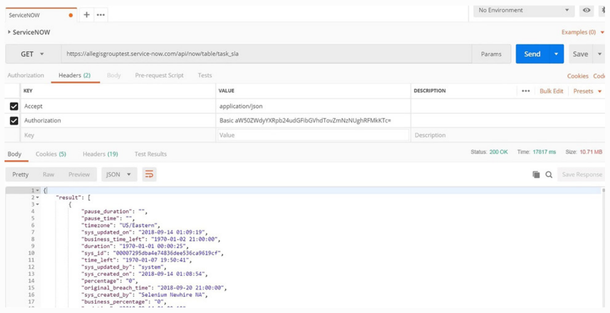
3. **Passing parameters and filters in API request.** The type of parameters or filters that can be passed will be critical to reduce the volume of data that gets pulled. Less data volume means better performance. In a few applications, there is a maximum limit of the data volume that cannot be exceeded (e.g., ServiceNOW can restrict the number of characters that can be in the API response to ensure there is no performance bottleneck).

4. **List of available attributes/calls.** API documentation of respective applications may have all the attributes or will have details of the code the metadata API invoked, which will give the needed details.

Postman

Postman (getpostman.com/) is a good API tool that we can install in our local machine for testing API calls before those are used in the actual code.

Different formats of the responses (e.g., XML, JSON, HTML, TXT) can be viewed, and different authorization methods (e.g., Oauth, Basic, NTLM) are supported.


ServiceNOW

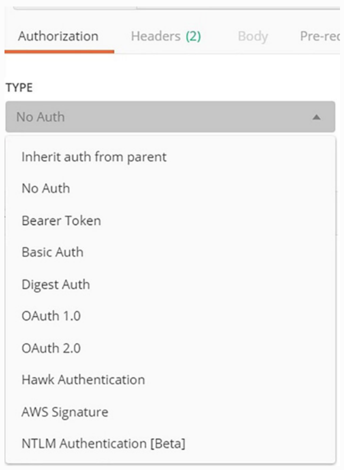
GET https://allegisgrouptest.service-now.com/api/now/table/task_sla

	KEY	VALUE	DESCRIPTION
✓	Accept	application/json	
✓	Authorization	Basic aW50ZWdyYXRpb24udGFibGVhdTovZmNzNUghRFMkKTc=	
	Key	Value	Description

Status: 200 OK Time: 17817 ms Size: 10.71 MB

```
{
    "result": [
        {
            "pause_duration": "",
            "pause_time": "",
            "timezone": "US/Eastern",
            "sys_updated_on": "2018-09-14 01:09:19",
            "business_time_left": "1970-01-02 21:00:00",
            "duration": "1970-01-01 00:00:25",
            "sys_id": "00007295dba4e74836dee536ca9619cf",
            "time_left": "1970-01-07 19:50:41",
            "sys_updated_by": "system",
            "sys_created_on": "2018-09-14 01:08:54",
            "percentage": "0",
            "original_breach_time": "2018-09-20 21:00:00",
            "sys_created_by": "Selenium Newhire NA",
            "business_percentage": "0",
```

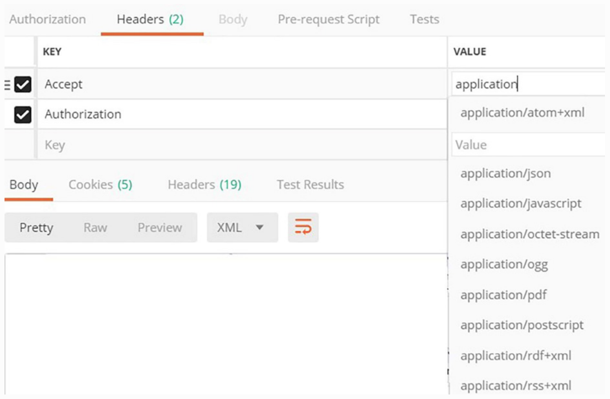
Figure 9: Postman ServiceNOW page


TYPE

No Auth

Inherit auth from parent
No Auth
Bearer Token
Basic Auth
Digest Auth
OAuth 1.0
OAuth 2.0
Hawk Authentication
AWS Signature
NTLM Authentication [Beta]

Figure 10: Authorization


	KEY	VALUE
✓	Accept	application
✓	Authorization	
	Key	Value

application/atom+xml
application/json
application/javascript
application/octet-stream
application/ogg
application/pdf
application/postscript
application/rdf+xml
application/rss+xml

Figure 11: Value: application

Authors:

Harshith Venkatesh at hvenkate@TEKsystems.com
Latha Jayaram at ljayaram@TEKsystems.com

© 2019 TEKsystems, Inc. ALL RIGHTS RESERVED.

ABOUT THE AUTHORS

Latha Jayaram
ljayaram@TEKsystems.com

Latha Jayaram is a senior practice architect for TEKsystems Global Services. With more than 20 years' experience in analytics, she has worked with a wide variety of tools and technologies in architecting, solutioning, data modeling, reporting and data warehouse applications.

Harshith Venkatesh
hvenkate@TEKsystems.com

Harshith is a Big Data engineer. He has worked on different Hadoop distributions like Cloudera, Hortonworks and MapR, using multiple tools including HDFS, Hive, Impala, NiFi, Sqoop, Spark and Pig. He also has significant experience with the Informatica Big Data Management suite.

About TEKsystems

We're partners in transformation. We help clients activate ideas and solutions to take advantage of a new world of opportunity. We are a team of 80,000 strong, working with over 6,000 clients, including 80% of the Fortune 500, across North America, Europe and Asia. As an industry leader in Full-Stack Technology Services, Talent Services and real-world application, we work with progressive leaders to drive change. That's the power of true partnership. TEKsystems is an Allegis Group company.

Experience the power of real partnership. **TEKsystems.com**

© 2019 TEKsystems, Inc. ALL RIGHTS RESERVED.