Data Extract, Transform and Load (ETL) Design

Version 1.1

**Contributors**:

Gaurav Gada

Shipra Gupta

# Version Info:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Updates | Contributor | Date |
| V 1.0 | Initial version | Gaurav Gada | 01/06/2017 |
| V 1.1 | Restructured document. Added Design and Specification Details | Shipra Gupta | 01/08/2017 |
|  |  |  |  |
|  |  |  |  |

# Objective

The purpose of this document is to capture design and specifications to extract, transform and load data from multiple systems (like Jobvite and NuView) into BigQuery.

# Assumptions

1. Data has to be extracted from two system of records, Jobvite and NuView.
2. Data will be extracted and consumed in CSV or JSON format.

# Scope

## In-Scope

* Analysis of current and future size of data
* Design and specifications to extract data from Jobvite and NuView
* Design and specifications to load data into BigQuery
* Data security and availability

## Out-of-Scope

* Procurement of servers and licenses
* Upgrade and maintenance of applications

# Overview

This design considers how data from internal systems (Jobvite and NuView) can be exported to a Data Warehouse i.e., BigQuery.

# Extraction

This section will include the design and specifications to extract data from different systems of records.

## Extraction from Jobvite:

Jobvite’s APIs are REST based web services that exchange JSON.

REST (Representational State Transfer) exploits technology and protocols of the Web, using HTTP methods and JSON data representation to create, read, update, and delete resources. Jobvite uses REST APIs to deliver stateless, scalable interfaces that allow customers or integration partners to create and access data from Jobvite near real time.

### Design

#### Approach

Following types of information will be extracted from Jobvite using different APIs. All APIs will be accessed using HTTP GET method as information needs to be only retrieved. All APIs will return information in JSON format.

1. Get Candidate

The Get Candidate API call contains job seeker information and the requisition level data that the job seeker applied to.

1. Get Requisition/Job Feed

The Job Feed API allows one to obtain all requisitions for a given company.

1. Get Location

The location API allows users to obtain all configured locations. The location value typically designates where the requisition is hiring candidates for.

1. Get Category

The GET category API allows users to retrieve all configured categories. Categories designate the requisition grouping within a customer’s career website. Ex. Engineering, Sales, Customer Success

1. Get Department

The GET department API allows users to retrieve all configured departments. Departments designate which group the requisition belongs to. Ex. Finance, Sales, Engineering. Departments are usually tied to one’s HRIS system. The difference between departments and categories is that departments are typically used for payroll purposes whereas categories are used to group requisitions within a customer’s career website.

#### Specifications

Detail specifications (Request and Response) for all above mentioned APIs are available in the document below:

161102 Jobvite API v2.pdf

1. Get Candidate - Page Number: 9
2. Get Requisition/Job Feed - Page Number: 35
3. Get Location - Page Number: 75
4. Get Category - Page Number: 84
5. Get Department- Page Number: 91

#### Prerequisites

* API Credentials – Can be requested via <http://www.jobvite.com/support/submit-a-case/>
  + API Names - To be provided in the request
  + API Key
  + Secret Key
* API URLs must use the ASCII character set.
* JSON Headers:
  + **Content-Type:** application/json; charset=utf-8
  + **Accept:** application/json
* Encoded Headers:
  + **Hash Algorithm:** HMAC-SHA256
  + **API Key:** api
  + **API Secret:** secret **String to Hash:** <API key>|<epoch> (Ex. api|1476218468)
  + **Hashed Value (X-JVI-SIGN)**: HMAC-SHA256(API Secret, String to Hash) (Ex. HMAC-SHA256(secret, api|1476218468)

#### Important Notes:

* These APIs also provide an option of getting the entire dataset or changed data. Based on the data volume, we can strategize whether we need to do a full refresh or partial refresh.
* We have 20,000 calls/day for this API, so rate limiting should not be a problem.

## Extraction from NuView:

NuView, a core Human Resource Management System (HRMS) is a web-native single sign-on system that simplifies the collection and management of the people data and standardizes key processes of the organization. This system aggregates all of the employee data into a single system of record, giving maximum visibility of information (human resource metrics).

There are various ways using which data from NuView can be extracted:

1. A report, set it to run on a specific schedule and email CSV file to recipients.
2. Export data in CSV format and then have it uploaded to any FTP/SFTP/SCP server using a workflow.
3. Customized forms in the application to display and save data in any format to your local disk. These forms can also be accessed via web URL.

### Design

#### Approach

NuView will be configured to generate data extract in CSV format at regular intervals and send this file to a remote FTP/SFTP location.

Benefits of this approach:

* This would be the most secure way to get data from NuView due to the security properties of secure shell transmission.
* This is an automated way to extract information. On the other hand, the other two methods will need manual intervention to generate the extract.

#### Specifications

* **Scheduler:** Everyday 5AM PST (*Exact Value to be discussed and determined*)
* **CSV File Name:** nuview\_{date}.csv
* **FTP Location Details:**
  + IP Address: {TBD}
  + Username: {TBD}
  + Password: {TBD}
  + Path: {TBD}

# Transformation

This section will include how extracted information from various systems will be transformed to make it structured and ready for persistence in the data warehouse.

## Design

### Approach

Once we have the CSV from NuView and JSON from Jobvite, we need to determine our infrastructure strategy i.e., do we load directly to BigQuery or do we perform transformations, simplify the data, remove unneeded attributes and then load the data into BigQuery. Based on this strategic decision, we can determine our staging and transformation infrastructure. For this, we can use PostgreSQL/MySQL or other open source database alternatives.

### Specifications

**Server (Where transformation will take place):**

Getting the raw data from JobVite and NuView can be run quite reliably if we have a reliable server end-point to receive data from these systems. This system would play a crucial role in getting our data on-boarded to BigQuery. This would be placed along with your existing IT infrastructure. We will receive the data from the two systems on this server and we will run the BigQuery load process here. This will provide the control needed to schedule incoming data requests, run any needed transformations and perhaps also serve as a contingency location.

# Load

This section will include how processed or transformed data from different sources will be loaded into the data warehouse i.e. BigQuery.

## Loading data into BigQuery:

Google BigQuery is an [enterprise data warehouse](https://cloud.google.com/solutions/bigquery-data-warehouse) that enables super-fast SQL queries using the processing power of Google's infrastructure. We can access BigQuery by using a [web UI](https://bigquery.cloud.google.com/) or a [command-line tool](https://cloud.google.com/bigquery/docs/cli_tool), or by making calls to the [BigQuery REST API](https://cloud.google.com/bigquery/docs/reference/v2) using a variety of [client libraries](https://cloud.google.com/bigquery/docs/reference/libraries) such as [Java](https://developers.google.com/api-client-library/java/apis/bigquery/v2), [.NET](https://developers.google.com/api-client-library/dotnet/get_started) or [Python](https://developers.google.com/api-client-library/python/). There are also a variety of [third-party tools](https://cloud.google.com/bigquery/third-party-tools) that we can use to interact with BigQuery, such as visualizing the data or loading the data.

BigQuery provides the following ways to load data into its systems –

1. BigQuery CLI\*
2. Web UI
3. API

## Design

### Approach

For our use case, the BigQuery CLI would be the best way to do the data load, primarily because it would be an automated process that we can trigger once we have our data transformed on our server.

Since we are working with JSON and CSV, we shouldn’t face any challenges loading the data into BigQuery.

As per Google documentation, in BigQuery, we typically should denormalize the data structure to enable super-fast querying. [https://cloud.google.com/bigquery/preparing-data-for-loading]

### Specifications

This section will include information about setting up the BigQuery for our project. Such as:

* Load configurations
* Storage
* Datasets
* Tables
* Jobs