**End User Guide**

**Excel to Json Utility**

Table of Contents

[1 Revision History 3](#_Toc520458846)

[2 Document Objective 4](#_Toc520458847)

[3 Prerequisites 5](#_Toc520458848)

[4 How To Populate the Excel 7](#_Toc520458849)

[5 Steps To Execute 9](#_Toc520458850)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Changes | Author |
| 1.0 | 25/10/2018 | Version 1 | Manik Batra |
| 2.0 | 05/02/2019 | Version 2 | Manik Batra |
| 3.0 | 21/05/2019 | Version 3 | Manik Batra |
| 4.0 | 19/06/2019 | Version 4 | Manik Batra |

# Document Objective

The purpose of this document is to act as a guide to convert excel data to JSON which can be directly imported to Gigya. The document is intented to be used by the market as part of their data migration activity. The main aspects included in this document are as below:

* **Format of the Input Excel file**
* **Steps to execute the utility**
* **Success and Error Message scenarios**
* **Output JSON creation**

# Prerequisites

Following are the prerequisites to use the Utility:

* [**Install Java**](https://www.wikihow.com/Install-Java)on your computer if it isn't installed. You can't run JAR files without Java installed on your computer. If you don't already have Java installed, go to Java's website at <https://www.java.com/en/download/> and click the Free Java Download button to get the latest version of Java, then install Java once it downloads. You may have to restart your computer for Java to be fully installed.

**Mac:**

When installing a non-Apple file on Mac, you may have to first click OK on the warning, click the Apple menu, click System Preferences, click Security & Privacy, unlock the menu, click Open Anyway next to the file's name, and then click Open when prompted before you can install the file.

* ***The input Excel format files:***

***Full registration Excel Lite Registration Excel***

The input files are provided for both Full Registration and Lite Registration Accounts. Both the files are categorized into two sheets:

**1.** **User Data** – The sheet contains the fields for all the account data captured for a user which is covered in detail below.

The files are structured in a specific way with headers as the Gigya schema fields. The schema comprises of mainly 4 categories:

**(i) Profile**

**(ii) Data**

**(iii) Subscriptions**

**(iv) Preferences**

The field names are stored in the Excel in a similar way it is stored in Gigya.

***Example 1:***

A field firstName is mapped in Excel as **profile.firstName** since it is a part of **Profile** schema.

Similarly, all the other fields are mapped based on the schema that they fall into i.e. Profile, Data or Subscriptions.

***Example 2:***

The field **interestCode** is a part of **areaOfInterest** which in turn is part of **Data** schema. So, the excel header value is mapped as **data.areaOfInterest.interestCode**.

There are some fields which are not a part of any of the three schemas mentioned above as they fall under the accounts schema. Those fields are mentioned as there exact names in the excel.

For example: created, isActive, isVerified etc.

**2.** **Market Data** – The sheet contains market specific values that are common and mapped for all the user accounts. Example – marketCode, locale etc.

**\*\* *Before using the utility, the data mapping should have already been completed for the market and the excel column name for both User and Market Data with the corresponding Gigya field value should have been finalised*. Once, the market sends us the data mapping, we will send a customized version of the Input Excel files to be used.\*\***

* ***Excel Field Validations***

There are few validations in place to ensure that the mandatory data is captured in the JSON.

**Market Data Sheet** : In the Market Data sheet, all the fields mentioned are mandatory. If any or all the fields are not provided, then an error will be thrown and the utility won’t run unless all the values are provided.

**User Data Sheet** : In the User Data sheet, the only mandatory value is profile.email. Email has been taken as a unique identifier for all the user account data.

**(i)**  If email is not provided for any user, the JSON will be created for all the other accounts but that account will not be included in the JSON created and an error will be thrown highlighting the row where email field is empty.

**(ii)** If the email provided for any user is invalid (i.e not of format **local-part@domain**), then in the output JSON, that user wont be included and an error message will be thrown highlighting the row where the email is of invalid format. For example – [John.Smith@example.com](mailto:John.Smith@example.com) is a valid email format.

**(iii)** The various date fields like data.birthDate, data.child.birthDate etc. have to be of Date Type in the Excel. The format of the date could be anything but the type of the Excel cell has to be of type DATE.

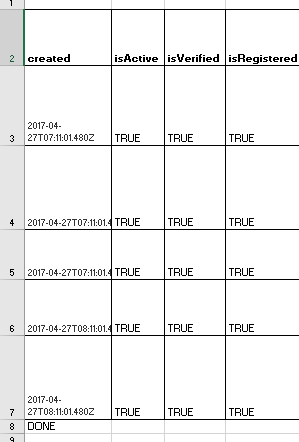
**(iv)** The various timestamp fields like created, data.areaOfInterest.creationDate, data.areaOfInterest.lastUpdateDate, subscriptions.email.lastUpdatedSubscriptionState etc. have to be in gigya recommended format **"yyyy-MM-ddTHH:mm:ssZ"** in the excel where yyyy is year, MM is month, dd is date, HH is hours, mm is minutes and ss is seconds.

**(v)** There are few schema fields like **isActive, isVerified, is Registered** etc that are of Boolean type i.e. they can either contain true or false value. If any other value is provided for these fields, then that complete user record will be skipped in the JSON and an error message will be shown in the status with the error field name and its corresponding row number.

**(vi)** There are few implicit fields that needs to have a value when the data is imported in Gigya. For full user registration, the fields **isActive, isVerified and isRegistered** should be mandatorily populated in the Excel for every user record. In case of Lite users, the field **isLite** should be populated.

**(vii)** In the case, where the field subscriptions.email.tags has multiple values, then the values has to be separated by a comma. For example in the excel the data could be populated like **sourceApplication:TRNINWEB\_MIG,details: Google.**

**\*\*Note** – The excel file contains a keyword **DONE** which implies the end of user records in the file. Please do not remove or modify the keyword. All the records should be incorporated above this word. PFB the example for your reference.



# How To Populate the Excel

The data needs to be populated in the excel in the respective header fields. A sample data has been provided in the embedded excel files.

A user data could be a single user record or a user record with multiple values i.e multiple childs, or subscriptions or other array data for various fields.

**Array Data Fields** – Any Gigya schema fields that can contain multiple values are termed as Array Data Fields. For reference – identities, areaOfInterest, externalApplication, child, subscriptions.

**Array Data Format** – If the user record has multiple values for an array data field, for example multiple subscriptions, then the complete row is replicated for that user in the excel with the same values for all the other fields, just the array field values of the subscription would be different.

Please refer to Scenario 2 or 3 below for more detail.

Below are the templates with few fields that will help in populating the excel with the data in a proper way.

***Scenario 1:***

***A user having one Child and one subscription***



***Scenario 2:***

***A user having two child details, one area of Interest details, and two subscriptions***

******

*The user*[***zkhan@yopmail.com***](mailto:zkhan@yopmail.com) has two child and subscriptions details which are the array data fields. So, two rows have been created for this user to capture the information of second array data record for Child and Subscriptions. The second row contains the same value for all the other fields. If, for any reason, the second row for this user record contains any different value for any field other than the array data fields, then that value will be the final value and the previous value will be overwritten.

For example- in the field profile.city, the first row has value Sydney but the second row contains the value Melbourne. It should be noted that both these rows are of the same user, so the value present in the last row would be the final value for any field and the same will be reflected in the JSON.

***Scenario 3:***

***A user having two child details, two area of Interest details, two external application details and three subscriptions***



*The user*[***zkhan@yopmail.com***](mailto:zkhan@yopmail.com) contains multiple array data field values. Subscriptions has the maximum array value i.e. 3, therefore, three rows have been created for this user. It should be noted that all the single field values are same in all the three rows. Only the values in the array data fields would be different.

***\*\* It should be noted that the rows created for a user having an array data record, should be all together in the Excel. For example, User A has 3 subscriptions, so 3 rows will be created in the Excel for this user one beneath the other. There should not be a row for any other user in between the 3 rows for User A.\*\****

**Password Fields –** There are two ways in which the password is stored in Gigya. In some cases, **compoundHashedPassword** is used. In others, a combination of **hashedPassword** and **algorithm** and other fields are necessary.

You are required to include either compoundHashedPassword or some combination of hashedPassword and password related parameters.

* **hash**: If **compoundHashedPassword** is not passed, then **hashedPassword** parameter is required.
* **hashSettings** : this includes:
  + **algorithm**: The algorithm used for the password, i.e. “md5”, “sha1”, or “sha256” among others.
  + **salt**: the BASE64 encoded value of the salt. If the **format is specified** and it contains **$salt**, then salt is required and should be clear text, **not BASE64 encoded.**
  + **format** : a template describing how to merge clear-text password that is entered by the user with the salt. The string must contain **$password**, which will be replaced by clear-text password. It may also contain **$salt** which will be replaced with the value passed to the salt parameter ( or you can pass the salt directly in the template and omit the **salt** parameter). For example:

***“format” : “$password::$salt”***

Example values for both the ways mentioned above:

// Example A

"password": {

    "compoundHashedPassword": "$S$D27r.aObIYkw3I8tO9VDYs.FfuF/4ZjBKDDMx0hxVvB3Kt2iCPEY"

}

// Example B

"password": {

    "hashedPassword": "Y2FlOTkzNThhYjRlYzE2YmZhYTMyYWI2MzFiMGFhOTc=",

    "hashSettings": {

        "algorithm": "md5",

        "salt": "lYzE2YmZhYTMy",

        "format": "$password::$salt"

    }

}

**Excel template if hashedPassword is being used:**



**Preferences:**

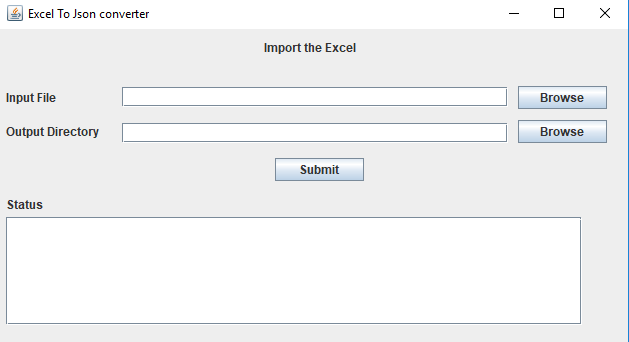
Every user must have one mandatory preference. Please refer to the attached Full registration excel on how to populate the preference fields.

# Steps To Execute

After the excel file has been correctly populated with the user data, the following steps could be followed to execute the utility.

**Step 1**: Double click on the Jar file in the below zip to run it. You should be able to see the screen as below:





**Step 2:** Click on browse against the Input file and select the excel file that you want to convert to JSON.

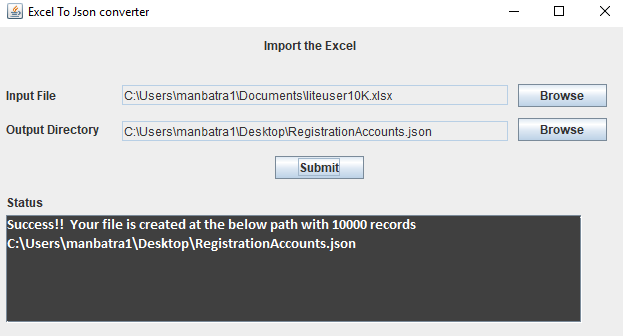
**Step 3:** Click on browse against the Output Directory and select the path where you would like to save the converted JSON file. A file with the name RegistrationAccounts.json will be created in the directory selected. If the file with this name already exists, then it will be overwritten.

**Step 4:** Click on Submit and your file will be created. You could also see the Status message in case of success or any validation errors as shown below:

***Messages:***

The messages displayed in the UI has been categorized into two:

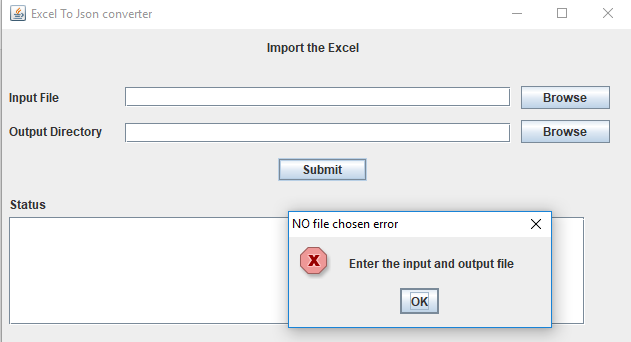
* **Success Message:** If the file has been created successfully without any errors, then the following screen will be displayed.



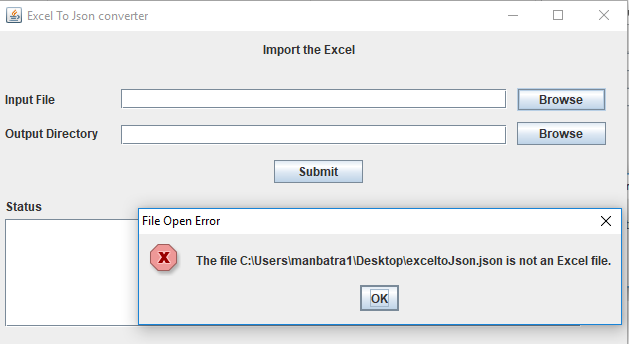
* **Error Messages:** The error messages has been further categorized into two:

**(i) File validation Errors:**

***If both the input and output file values are empty:***

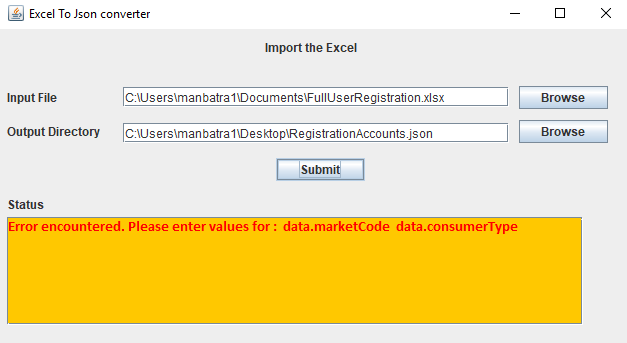


***If the input file provided is not an excel file ( xls, xlsx):***

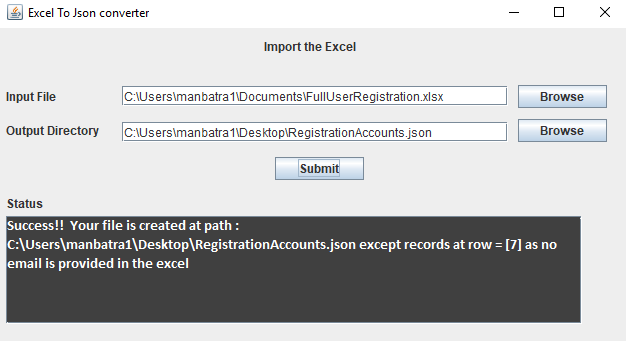


**(ii) Data Validation Errors:**

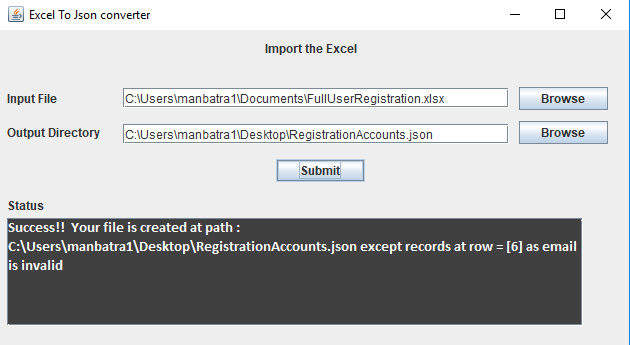
***If any/all the Market Data values are empty:***



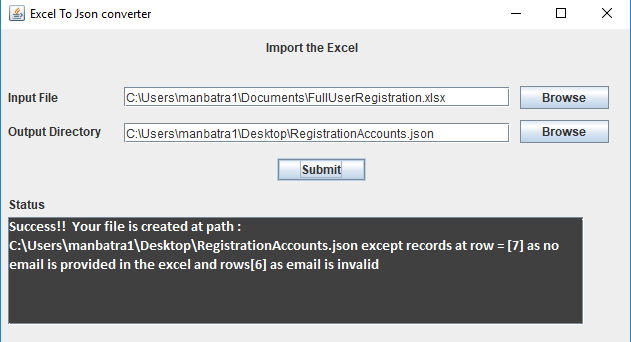
***If profile.email field value is empty i.e email is not provided for any user:***



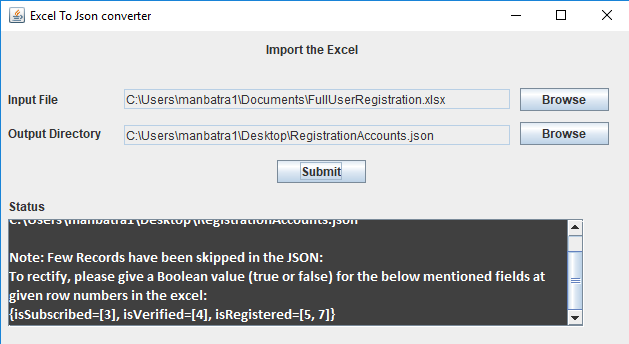
***If profile.email field value is invalid for any user record:***



***If profile.email field value is invalid and profile.email field value is empty for any user record:***



***If a string or text value is provided in any of the Boolean fields for any user record:***



***After Execution:***

After the above steps have been executed, the output Json file will be created. Below is the sample of the JSON.

***Output JSON Sample***

*** ***

***It is recommended that the user reviews the output JSON manually and randomly checks for few records to ensure that they have been incorporated as expected before sending it to Gigya.***