|  |
| --- |
|  |
| Development Spec Document |
| Data Loader |
| rev 1.0  Reltio Technical Services |
| Joel Snipes |
| 6/20/2017 |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Author Name** | **Date** | **Description** |
| 1.0 | Joel Snipes | 6/20/2017 | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

The Data Loader is a tool for bulk loading entities, relations, and interactions into a reltio tenant in JSON form. The tool allows for the process tracking with email alerts and output redirection to a csv file when records fail to load. The tool is commonly used in sequence following the JSON Generator tool to prepare data for loading.

## Use Case(s)

You have tens of thousands of records that need to be loaded for a customer in JSON format, but you know some of the records may have bad data. You use the Data Loader tool to load the data and the bad records are automatically filtered to CSV file for human review, along with the reason the record was rejected. When the load is complete you receive an email notification alerting you that the job is done so that you can begin reviewing the bad records and making corrections.

## Constraints & Assumptions

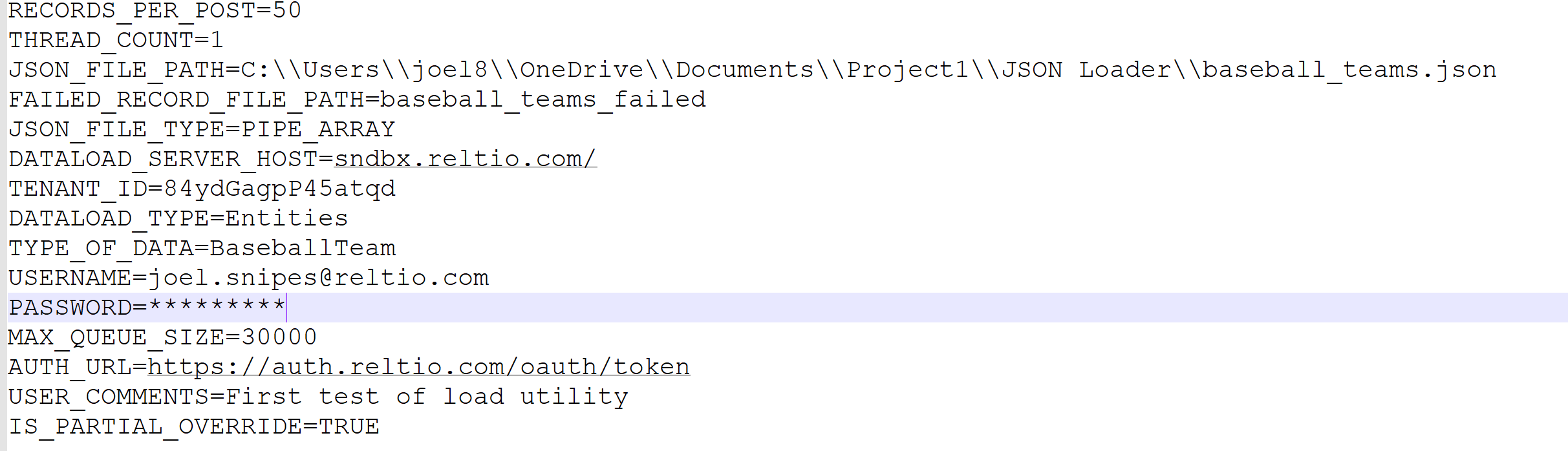
* Data must come in one of the three formats that the loader accepts, PIPE\_ARRAY, ARRAY, or OBJECT
* Entity, Relation, and Interaction data must be loaded separately and specified in the config file
* If the process tracker is to be used, the user must have access to the L3 config of the tenant
* User must know ahead of time whether data to be loaded will be a partial override of existing data

# Configuration Requirements

In order to execute the Data Loader tool a configuration file must be present. Options and a sample are listed below.

|  |  |  |
| --- | --- | --- |
| **Data Loader Configuration Options** | | |
| **Setting** | **Optional** | **Options** |
| RECORDS\_PER\_POST | N | Number of Records per Post. Recommended Value: 50 |
| THREAD\_COUNT | N | Number of Threads used in data load. Recommended Value: 10 |
| JSON\_FILE\_PATH | N | Path of the JSON File to be loaded |
| FAILED\_RECORD\_FILE\_PATH | ? | Output path for any records that fail to load |
| JSON\_FILE\_TYPE | N | PIPE\_ARRAY, ARRAY, or OBJECT |
| DATALOAD\_SERVER\_HOST | N | Host name of the data load server |
| TENANT\_ID | N | ID of target Tenant |
| DATALOAD\_TYPE | N | Entities/Relations (What is being loaded) |
| TYPE\_OF\_DATA | N | Entity Type or Relationship Type |
| USERNAME | N | Reltio Username |
| PASSWORD | N | Reltio Password |
| MAX\_QUEUE\_SIZE | Y | Maximum size of record queue. Recommended Value: 300000 |
| AUTH\_URL | N | Complete Auth URL |
| USER\_COMMENTS | Y | Helpful Comments Describing the job |
| IS\_PARTIAL\_OVERRIDE | N | TRUE/FALSE |

## Config Example



## Classes and Methods

Below the most relevant classes and methods are outlined

|  |  |  |
| --- | --- | --- |
| **LoadJsonToTenant** | | |
| **Method** | **Arguments** | **Description** |
| *main* | String[] **args** | This class is just a shell for the main method |

|  |  |  |
| --- | --- | --- |
| **DataloadFunctions** | | |
| **Method** | **Arguments** | **Description** |
| *invalidJSonError* | String **json**  DataloaderInput **dataloaderInput**  ReltioFileWriter **reltioFileWriter** | Writes bad JSON to rejected record file, writes the bad record to the consol, and increments the Failure Count |
| *getAttribute* | List<Attribute**> attributes**  Object **obj** | Takes in an object and list of attributes, adds the attributes from the object to the list, and returns the list |
| *getAttribute* | List<Attribute**> attributes**  String **value** | Takes in an a string and list of attributes, adds the attributes from the string to the list, and returns the list |
| *getAttribute* | List<Attribute**> attributes**  String **value**  String **type** | Takes in an a value string, type string, and list of attributes. Creates an attribute from the value and sets it type then adds the attributes to the list, and returns the list |
| *getAttribute* | List<Attribute**> attributes**  String **value**  String **type**  Boolean **dataProvider** | Takes in an a value string, type string, and list of attributes. Creates an attribute from the value and sets it type and dataProvider then adds the attributes to the list, and returns the list |
| *getAttribute* | List<Attribute**> attributes**  String **value**  String **type**  String **updateDate** | Takes in an a value string, type string, and list of attributes. Creates an attribute from the value and sets it type and updateDate then adds the attributes to the list, and returns the list |
| *getAttribute* | List<Attribute**> attributes**  String **value**  String **type**  String **createDate**  String **updateDate** | Takes in an a value string, type string, and list of attributes. Creates an attribute from the value and sets it type, createDate and updateDate then adds the attributes to the list, and returns the list |
| *getAttribute* | List<Attribute**> attributes**  String **value**  String **type**  String **createDate**  String **updateDate**  String **deleteDate** | Takes in an a value string, type string, and list of attributes. Creates an attribute from the value and sets it type, createDate, deleteDate and updateDate then adds the attributes to the list, and returns the list |
| *checkNull* | String **value** | Checks value for possible variations of a null value returns true if not null |
| *getStringValue* | Object  **obj** | Returns value of object as a string |
| *printDataloadPerformance* | long **totalTasksExecuted**  long **totalTasksExecutionTime**  long **totalQueueWaitTime**  long **programStartTime**  long **numberOfThreads** | Prints performance statistics to the console |
| *waitForTasksReady* | Collection<Future<Long>> **futures**  int **maxNumberInList** | Waits for futures (load tasks list put to executor) are partially ready.maxNumberInList parameters specifies how much tasks could be uncompleted. |
| *waitForQueue* | String **srcUrl**  int **numberOfEvents**  ThreadPoolExecutor **ThreadPoolExecutor**  ReltioAPIService **reltioAPIService** | Waits for the queue to drop below a particular size (numberOfEvents) |
| *sendHcps* | String **srcUrl**  String **stringToSend**  ReltioAPIService **reltioAPIService** | Sends an entity(stringToSend) to the srcURL and returns a response |
| *getQueuesSize* | String **srcUrl**  ReltioAPIService **reltioAPIService** | Returns the queue size as an int array |

## Test Cases

## Link to test cases and results are below

## <https://docs.google.com/spreadsheets/d/1n-gQCTscDx8DmWtqZuxyQD3ftfAt1mVgqDxF7kqMGR0/pubhtml>

## 

# Deployment & Instructions

In order to use the process tracker portion of the data loader the linked [entity type](https://help.reltio.com/sites/default/files/processtracker_1.json) must be added to your L3 config. From there follow the build and execution instructions below.

## Building

The build path of the **dataload-processor** must include the **reltio-cst-core**. The main method of the application is at the following path.

dataload-processor / src / main / java / com / reltio / cst / dataload / impl / LoadJsonToTentant.java

## Executing

When you are finished creating your configuration file and have added the process tracker entity type to your L3 you will be ready to execute the Data Loader. The command format is as follows.

|  |  |
| --- | --- |
| **Operating System** | **Command Format** |
| Windows | java -jar reltio-dataload-processor-2.0.2.jar propertiesFile.txt > $logfilepath$ |
| Unix | nohup java -jar reltio-dataload-processor-2.0.2.jar propertiesFile.txt > $logfilepath$ 2>&1 & |

# Other Relevant Links

|  |  |
| --- | --- |
| **Link** | **Description** |
| [Bitbucket Repository](https://bitbucket.org/reltio-ondemand/util-dataload) | Link to the Bitbucket Repository where the source is maintained |
| [Knowledge Base Writeup](https://help.reltio.com/kb/865) | Link to the write up about the tool on the Reltio Knowledge base |