**GRIT Midtier Developer’s Doc**

**Generating JSONs**

Version 1

December 7, 2018

Table of Contents

[Static GRIT Mongo Collections 3](#_Toc531970669)

[Generate the surveyQuestions and surveyQuestionSets JSONs 5](#_Toc531970670)

[Generate the Badge and BadgeBase64 JSONs 10](#_Toc531970671)

[Generating the BadgeBase64 JSON 11](#_Toc531970672)

[Generating the Badge JSON 12](#_Toc531970673)

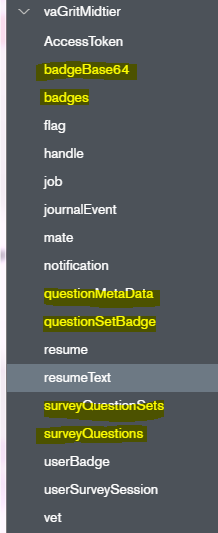
[Generate the QuestionMetaData 14](#_Toc531970674)

[Generate QuestionSetBadge JSON 14](#_Toc531970675)

# Static GRIT Mongo Collections

The GRIT Mongo collections that are static are populated using JSON files. Highlighted in yellow are the static mongo collections.

# 



|  |  |  |
| --- | --- | --- |
| Collection | Description | JSON File |
| badgeBase64 | Contains the Base 64 data for the badge image. Each badge has a small awarded, small unawarded, large awarded, and large unawarded Base 64 document. | badgesBase64\_data\_minified.json |
| badges | Contains the badge information: title, unawarded text, awarded text, and the four badgeBase64 ids | badges\_data\_minified.json |
| questionMetaData | The question ids for the meta data surveys of mood faces, mood text, and life changes | questionMetaData\_minified.json |
| questionSetBadge | The badge id that will be awarded when a user completes a baseline survey | questionSetBadge\_data\_minified.json |
| surveyQuestionSets | The different question sets for the baseline surveys | surveyQuestionSets\_data\_minified.json |
| surveyQuestions | The survey questions | surveyQuestions\_data\_minified.json |

The JSON files are in the git “va-grit-midtier” repository “midtier/assets/json/Minified\_JSON\_Files\_For\_Import” folder .

The following sections describe how to generate the different JSONs used to populate the static mongo collections:

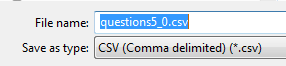
1. [Generate the surveyQuestions and surveyQuestionSets JSONs](#_Generate_the_surveyQuestions)
2. [Generate the Badge and BadgeBase64 JSONs](#_Generate_the_Badge)
3. [Generate QuestionSetBadge JSON](#_Generate_QuestionSetBadge_JSON)
4. [Generate QuestionSetBadge JSON](#_Generate_QuestionSetBadge_JSON)

## Generate the surveyQuestions and surveyQuestionSets JSONs

The survey questions and survey question sets are generated from the GRIT question scoring & actions spreadsheet.

1. **Open the latest GRIT questions scoring & action\_vx.y.xls spreadsheet**. Where” x.y” is the version number. For example 5.0. The spreadsheet is in the box GRIT/Content/Assessment folder, https://ibm.ent.box.com/folder/51746871296
2. **Create the questions csv file**.
3. Click on the QuestionPool tab in the spreadsheet
4. Selec t “Save As “ -> “Other Formats”
5. Change the “File name field” to something like “questionsx\_y.csv”. Select from the “Save as type” field to “CSV (Comma delimited) (\*.csv)”

For example:



1. **Create the questionSets csv file.**
2. Click on the QuestionSet tab in the spreadsheet
3. Selec t “Save As “ -> “Other Formats”
4. Change the “File name field” to something like “questionSetsx\_y.csv”. Select from the “Save as type” field to “CSV (Comma delimited) (\*.csv)”

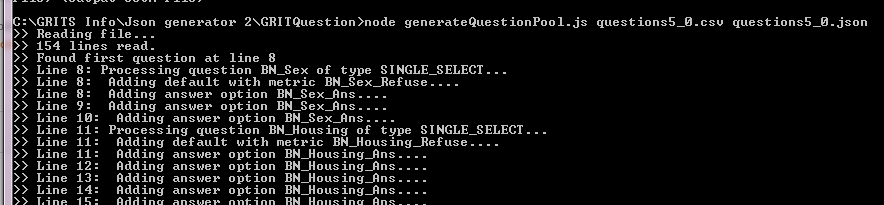
For example:



1. **Generate the questions json**.
2. Use the generateQuestionPool tool that is in the git /midtier/tools/GRITQuestion folder
3. node generateQuestionPool.js <csv file> <output json file>

Example :

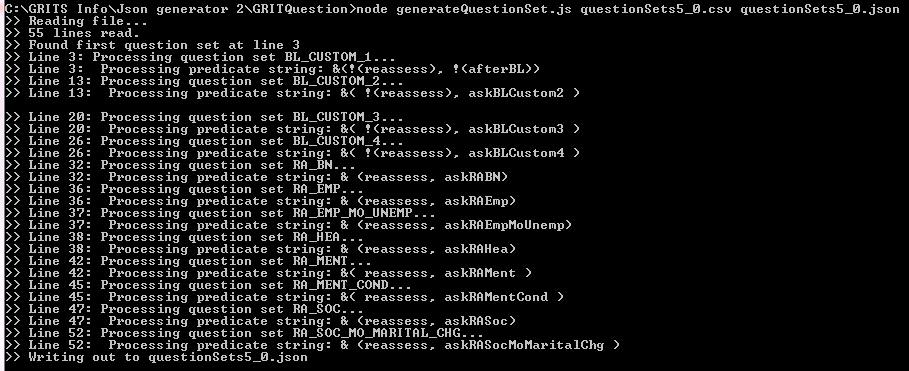
node generateQuestionPool.js questions5\_0.csv questions5\_0.json



1. **Generate the questionSets json**.
2. Use the generateQuestionSet tool that is in the git /midtier/tools/GRITQuestion folder
3. node generateQuestionPool.js <csv file> <output json file>

Example:

node generateQuestionSet.js questionSets5\_0.csv questionSets5\_0.json

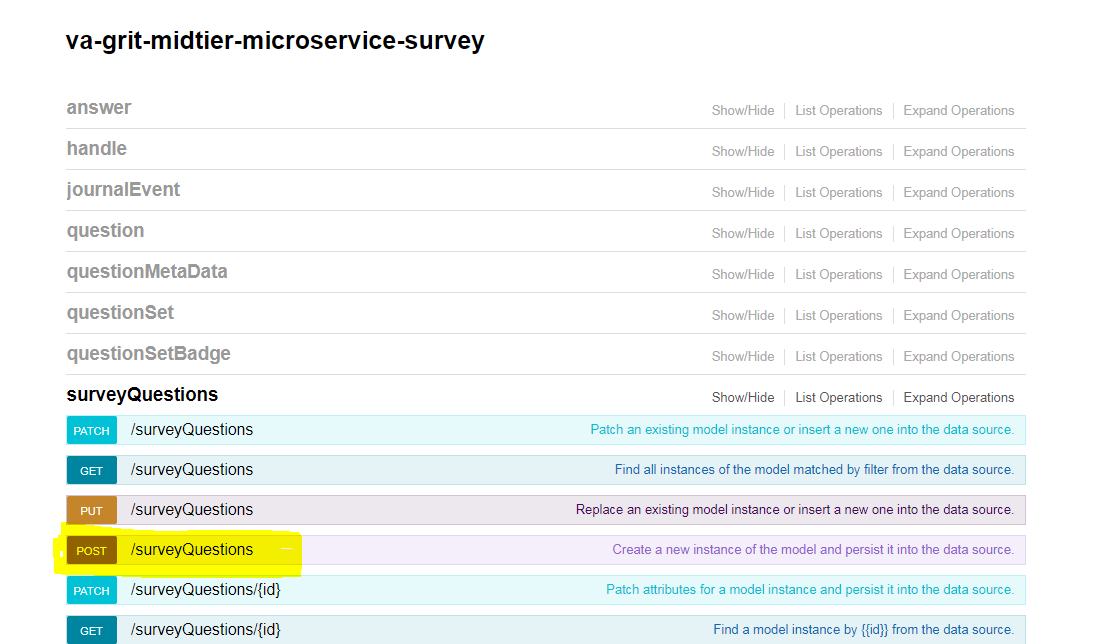


1. **Generate a “surveyQuestions” JSON Mongo Import**. Create an import that can be used to populate the mongo “surveyQuestions” collection.

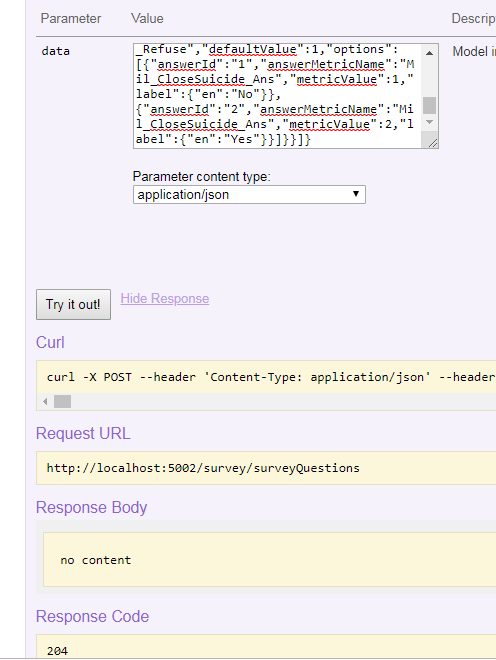
For some reason the jsons that were created using the node tool above does not import using the mongo client. A solution is to use loopback api explorer to do the post of the json mongo and then export the data into a minified data file. There might be a better solution using a mongo command line client. But this is a solution that works

1. Start your local survey microservice
2. Delete the mongo documents in the “surveyQuestions” collection in your **local** mongo database.
3. Post the question json using the LoopbackAPI explorer into the surveyQuestions mongo collection.

* Go to the Loopback API explorer: <http://localhost:5002/explorer/>
* Click on the surveyQuestions POST. Highlighted in yellow below



* Copy the survey question json into the “data” field.
* Press “Try it Out” . You should see a 204 response, and the surveyQuestions is now populated in the mongo database.

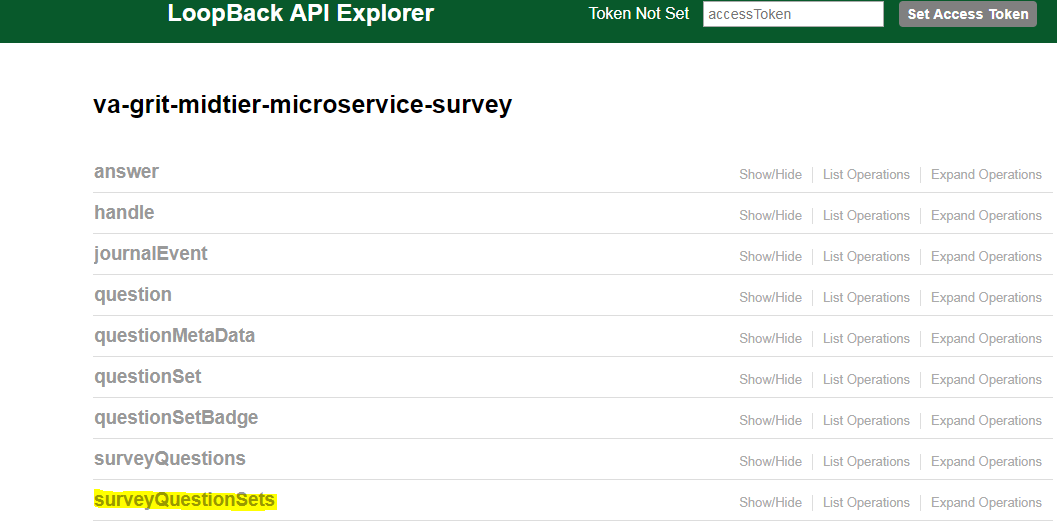


1. Verify the data is in your local mongo “surveyQuestions” collection. Verify words such as “don’t” looks good and do not contain a special character.
2. Export a minified “surveyQuestions” json from mongo.



1. Save the file into git. The folder is va-grit-midtier/midtier/assets/json. Give it a name of “surveyQuestions\_data\_minified.json”.
2. **Generate a “surveyQuestionSets” JSON Mongo Import**. Create an import that can be used to populate the mongo “surveyQuestionSets” collection.
3. Start your local survey microservice
4. Delete the mongo documents in the “surveyQuestionSets” collection in your **local** mongo database.
5. Post the questionSets json using the LoopbackAPI explorer into the surveyQuestionSets mongo collection.

* Go to the Loopback API explorer: <http://localhost:5002/explorer/>
* Click on the surveyQuestionSets POST. Highlighted in yellow below



* Copy the survey questionSets json into the “data” field
* Press “Try it Out” . You should see a 204 response, and the surveyQuestionSets is now populated in the mongo database

1. Verify the “surveyQuestionSets” collection in the mongo database looks correct
2. Export a minified “surveyQuestionSets” json from mongo



1. Save the file into git. The folder is va-grit-midtier/midtier/assets/json. Give it a name of “surveyQuestionSets\_data\_minified.json”.
2. **Test the Survey using Postman**. Refer to the “Surveys.doc”, Survey Testing section for details.

## Generate the Badge and BadgeBase64 JSONs

The badges for GRIT are stored in Mongo in a Base64 format. Each badge has four different Base64 states: Small unawarded state, small awarded state, large unawarded state, and large awarded state.

There are two collections in mongo for badges.

* badges. Contains the following data : addOrder, category, title, description, unawardedSmallBase64, unwardedLargeBase64, awardedSmallBase64, and awardedLargedBase64.
* badgeBase64. Contains the Base64 Images.

The following are the steps to add a new badge to GRITS.

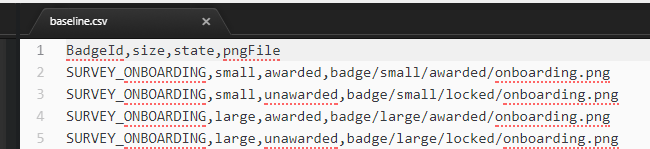
1. Retrieve the badge png and references from the UX Team. For release 1 badges, the data is in the badge\_export.zip file stored in the box folder.
2. Create the badge json file. Refer to section, [Generating the BadgeBase64 JSON](#_Generating_the_BadgeBase64) for details.
3. Create the badgeBase64 json. Refer to section, [Generating the Badge JSON](#_Generating_the_Badge) for details.
4. Add the badge to the badgeConstant.js file. . The file is in the midtier/common/npm-packages/va-grit-microservices/constant directory. Using gradle to copy this file to the different microservices.
5. Test Using the POSTMAN VA-GRIT Collection, Badge, getBadgeTypes.

### Generating the BadgeBase64 JSON

Each badge has four different states: small unawarded, small awarded, large unawarded, large awarded.

1. Use the badgeBase64JsonGenerator . This tool is in git, midtier/assets/tools directory.
2. In this directory are cvs files that are used to generate the base 64 json. Each badge should have 4 lines in the cvs file. The first row is the header that shows the format.

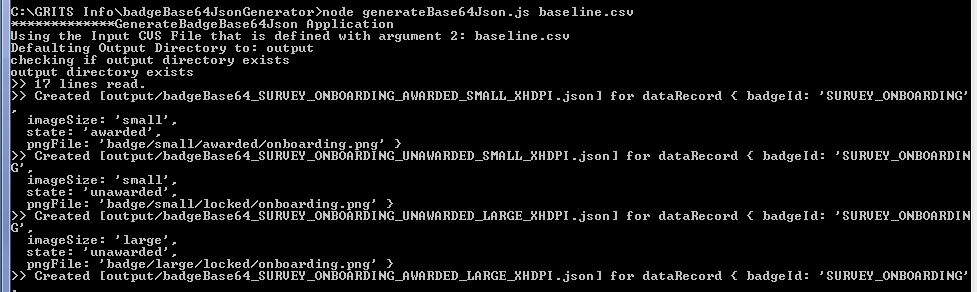
* BadgeId : The badge id in the badgeConstants.js file. This file is in the common/npm-package/va-grit-microservices/constants folder
* size: The size value can be “small” or “large”
* state : The state can be “awarded” or “unawarded”
* pngFile: The location where the png is. Note: we are using xhdpi



1. node badgeBase64JsonGenerator <optional csv files> <optional output location>

* If csv file is not specified, it will default to inputFile.csv
* If the output location is not specified, it will default to the output folder

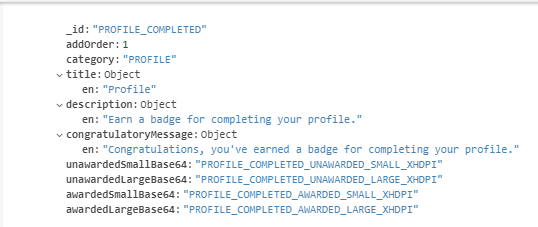
Example:



1. Start your local Loopback API Explorer for the vet microservices . <http://localhost:5001/explorer/>
2. Do the POST /badgeBase64. Do this for all 4 JSON Badge IDs .
3. Verify the 4 badgeBase64 documents are in mongo
4. Using Mongo, Export the badgeBase64 collection into a minified JSON file. The name should be “badgeBase64\_data\_minified.json”.
5. Store the minified file into git. The location is midtier/assets/json/ Minified\_JSON\_Files\_For\_Import.

### Generating the Badge JSON

The Badge JSON are manually created . The following shows of an example badge document.



1. Copy a badges\_xxxx.json file as a template. The badges files is in git. “midtier/json/Manual\_Generated\_Badge\_JSON\_Files” folder.
2. Fill in the json with the following data:

Currently, for Release 1, all the badges have an addOrder of 1. The idea of the addOrder is that the mobile app, will first invoke the user/badges/getBadgeTypes api with a highestAddOrder of 0. All the badges will be returned with an addOrder of 1. Later if we need to add different badges, then we will have a higher addOrder of 2. And the mobile app, will have a highestAddOrder of 1.

1. Category : Currently Category values are : “PROFILE”, “EMPLOYMENT”, “BASELINE”, “REASSESSMENT”, “SKILLS”, “SQUAD”. If a new category is added, it should match the mobile app category.
2. Title: Title of the badge. The title can be in “en” or “es”. The value is given by the UX team. Currently, it is the export\_export/reference. Just look for the badge for the different states, and the title, description, and congratulatory message it in it.
3. Description: Description of the badge. Same as the title where the text is in the reference directory.
4. CongratulatoryMessage: Congratulation message. Same as the title where the text is in the reference directory.
5. unawardedSmallBase64: The id of the small unawarded badge that was generated.
6. unawardedLargeBase64: The id of the large unawarded badge that was generated.
7. awardedSmallBase64: The id of the small awarded badge that was generated.
8. awardedLargeBase64: The id of the large awarded badge that was generated.
9. Start your local Loopback API Explorer for the vet microservices . <http://localhost:5001/explorer/>
10. Do the POST /badges for the JSON that was manually created.
11. Check that the badge document was inserted correctly in the “badges” mongo collection.
12. Export the badges collection from mongo. The file should have a name of “badges\_data\_minified.json”
13. Store the minified file into git. The location is midtier/assets/json/ Minified\_JSON\_Files\_For\_Import.

## Generate the QuestionMetaData

The QuestionMetaData Collection has one document that indicates the questionSetId for Life Changes, Mood Text, and Mood Faces.

If for some reason the questionSet Ids need to be changed, perform the following:

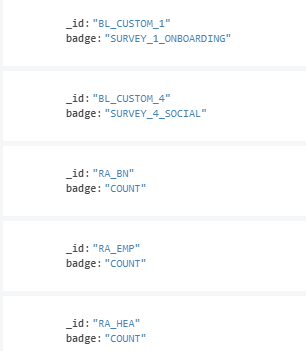
1. Go to your local mongo and update the document.
2. Export the questionMetaData from mongo. The file should have a name of “questionMetaData\_minified.json”
3. Store the minified file into git. The location is midtier/assets/json/ Minified\_JSON\_Files\_For\_Import.

## Generate QuestionSetBadge JSON

When a Baseline Survey is completed, the mid-tier will determine what badge should be awarded using the QuestionSetBadge collection.

The QuestionSetBadge document, has two fields.

* Id : The id of the questionSet
* Badge: A value of “COUNT” or the Badge ID. A “COUNT” indicates a count for a particular badge. For example: Reassessent has a count of 1x, 5x, 10x, 20x, 30x, and 50x. Mood has a count of 1x, 5x,10x, 20x, 50x, and 100x. The count for Reassessment and Mood is stored in the userSurveySession for the vetId.



The following are the steps to add QuestionSetBadge document

1. Most likely the reason you need to do this is because there is a new Baseline Question Set. All the Question Set Ids are document in the **GRIT questions scoring & action** spreadsheet.
2. If that is the case, then the easiest way to add a new questionSetBadge document, is through the loopback api explorer for survey. Start your local Loopback API Explorer for the vet microservices . <http://localhost:5002/explorer/>
3. Do a POST for the QuestionSetBadge with the id and badgeId
4. Verify the document was created correctly in the mongo “questionSetBadge” collection
5. Export the “questionSetBadge” collection into a minified json file with the name of “questionSetBadge\_data\_minified.json”.
6. Store the minified file into git. The location is midtier/assets/json/ Minified\_JSON\_Files\_For\_Import