GT BATCH FORM PROCESSING

Generating eforms for you

**Document Owner:** Jay Jorgensen, Technical Consultant

**Last Updated:** Gusty Toluta’u, Lead Tech Consultant, 05/27/2020





Table of Contents

[Table of Contents 2](#_Toc49759869)

[Introduction 2](#_Toc49759870)

[CSV File Formatting 3](#_Toc49759871)

[General Formatting Rules 3](#_Toc49759872)

[Guidelines: 3](#_Toc49759873)

[Batch Processing Setup 4](#_Toc49759874)

[Setting Up Batch Processing for a Form Type 4](#_Toc49759875)

[Setting up a batch configuration for a form type: 4](#_Toc49759876)

[Special Cases 5](#_Toc49759877)

[Special Fields 5](#_Toc49759878)

[Grid Segments 5](#_Toc49759879)

[Add Task Search Keys 6](#_Toc49759880)

[Uploading and Processing a Batch 7](#_Toc49759881)

[Uploading a Batch 7](#_Toc49759882)

[PeopleCode Hooks 8](#_Toc49759883)

[General Information 9](#_Toc49759885)

[PeopleCode Hook Definitions 9](#_Toc49759886)

[PreProcessBatch 9](#_Toc49759887)

[PreProcessLine 9](#_Toc49759888)

[PostFormCreate 9](#_Toc49759889)

[PopulateField 10](#_Toc49759890)

[PopulateGrid 10](#_Toc49759891)

[FinalizeForm 10](#_Toc49759892)

[FinalizeBatch 11](#_Toc49759893)

Introduction

GT Batch Form Processing allows users to use a CSV file to create forms for each line in the file. This document outlines the following:

* Formatting CSV files to be compatible with GT’s batch tool
* How to use GT’s batch tool in general
* PPC hooks that can be used to add custom client functionality to GT’s batch tool id

CSV File Formatting

General Formatting Rules

In order to be compatible with the GT Batch Tool, CSV files need to be formatted according to the following guidelines:

Guidelines:

The first row contains the Input Field IDs – At what level are the forms being initiated? The level at which the forms are being initiated will identify what percent of transactions are started by each role. Knowing which forms are being initiated and where they are being initiated, will assist you in identifying how distributed your eForm implementation is. (ie. Departments are initiating vs core office initiating.)



The second row contains the Field Labels. These do not have to match Input Field IDs or Field Names used during Batch Setup. This row is not used by the Batch Processing tool. It is simply an arbitrary label created for convenience by the creator(s)/viewer(s) of the CSV file. Despite not serving any batch processing purpose, this row still must exist even if you leave the cells blank. The Batch Processing tool assumes that the first form in a batch will always be on row 3.



Each subsequent row contains data for one form. The data in these rows will populate form fields when the tool begins processing and creating forms.



Batch Processing Setup

Setting Up Batch Processing for a Form Type

Setting up a batch configuration for a form type:

1. Navigate to [Root] > Manage GT eForms 3.x > Batch > Batch Load
2. Click the ‘Add a New Value’ tab
3. Enter the ‘Form Type’ and ‘Batch Configuration’
4. Click ‘Add’

You will see a screen that looks like this:



Explanation of the fields on the grid are as follows (from left to right):

* Input Field ID – This is a required field. It corresponds to the top row of your CSV file. The batch tool will match the text in the top row of your CSV with this field. Using the CSV file shown in the General Formatting Rules section, the Input Field IDs in this setup table would be EMPLID, EMPL\_RCD, FIRST\_NAME, LAST\_NAME, and EFFDT.
* Input Field Number – This is a required field. This tells the tool which order the input field IDs will appear in the CSV file.
* Segment Type – This is a required field. This specifies the type of segment that the form field is on, i.e. Column or Grid.
* Segment Record – This is the record tag that the form field is on. It is used to match the CSV field value to the form’s desired Field Value when the form is created and fields are populated. This is not a required field because there are special cases that may require this to be left blank. For more information on when you might leave this field blank see the Special Cases section below.
* Field Name – This corresponds to the form field name that the CSV file field will match to when creating the form and populating field values. It is not required because you can have fields in your CSV that will not populate actual form fields.
* Default Field Value – You can use this field to default a value when your CSV has a blank value. i.e. If the SUPERVISOR\_ID row in a CSV is blank, a Default Field Value can be set to a certain supervisor’s emplid so that rows with blank supervisors will still populate a value when the form is created.

The setup for the example used in the General Formatting Rules section might look like this:



Special Cases

Special Fields

* **INITIATOR\_ID** – If you would like to control the initiator id for each form separately, you must add a row in the setup (and a column in your CSV) with an Input Field ID of INITIATOR\_ID. If you do not specify INITIATOR\_ID in your CSV, the batch tool will assume the initiator is the user that is processing the batch.
* **G3THIS\_COMMENT** – If you would like to add comments for forms in a batch, you must add a row in the setup (and a column in your CSV) with an Input Field ID of G3THIS\_COMMENT.

Grid Segments

Grid segments must be handled differently because the batch tool has to convert flattened data from a single CSV row to multiple rows in a grid on a form. An example of how to set up the configuration for a grid segment is as follows:

1. **Add a row for each field in the form grid.** In the screenshot below, the assumption is that there are only two fields in the form grid.



1. **Set the configuration as you normally would.**
2. **Append a ‘1’ to the ends of each Input Field ID.**
3. **Set Segment Type to ‘Grid’ and ensure that you use the same Segment Record tag for each row of the same grid.**

****

1. **Repeat steps 1-4 according to the number of rows you want to be able to add to your form grid.** i.e. If I want to be able to add data for 4 rows in my form grid, I’ll repeat those steps 3 more times.
	1. Instead of appending a ‘1’ to Input Field ID each time, append the number of reoccurrence of each field.

When you are done, it will look something like this:



Notice that even though the appended number for each Input Field ID increases, the Field Names and Segment Record tag will stay the same.

It is important to know that form grid values will be inserted according to the number appended to the Input Field ID, and not in the order they appear on the spreadsheet. This means that when ACTION4 and REASON4 appear before ACTION1 and REASON1 in the CSV, the values in ACTION1/REASON1 will be inserted into the form grid before ACTION4/REASON4.

Also, just like column segment fields, it is not required to populate every Input Field ID. If for a particular form it's necessary to add fewer rows than the configuration’s capacity, that is perfectly fine. However, best practice would be to leave the higher-numbered fields blank first.

Add Task Search Keys

Task Search Keys are a prominent example of a time it is necessary to add fields to the batch setup that are not on a form. When adding Task Search Keys it is not necessary to specify a segment record or field name on a row in the setup grid.

Additionally, there are some search keys that receive built-in special handling:

* **OPRID** – If OPRID is on the form’s search record, it will be populated by the INITIATOR\_ID field from the CSV. If INITIATOR\_ID is not specified, OPRID will be populated with the OPRID of the user that is processing the batch.
* **ROWSECCLASS** – If ROWSECCLASS is on the form’s search record, it will automatically be filled based on the value assigned for OPRID. There is no need to have ROWSECCLASS be a field in your CSV.
* **OPRCLASS** – If OPRCLASS is on the form’s search record, it will automatically be filled based on the value assigned for OPRID. There is no need to have OPRCLASS be a field in your CSV.

Uploading and Processing a Batch

Once the batch setup is complete, you will be able to upload and process a batch which will create a form each row in your csv.

Uploading a Batch

The following section details how to upload a batch csv file.

1. **Navigate** to [Root] > eForm Solutions Setup > Batch > Process Batch
2. **Click the Add a New Value tab**
3. **Click Add**
4. **Click Upload File**



1. **Click Choose File** and browse to the location where your csv is saved.
2. **Click Upload**
3. **Click Prepare**

****

1. **Click Process**

This may take some time depending on the number of forms in your batch file. When the tool finishes processing the batch, it will change the status of each row to Complete or Error depending on whether or not the tool successfully created a form. If an error did occur, the error will be output to the Line Error Text field to aid in debugging.



PeopleCode Hooks

In order to facilitate client customization, there are several PeopleCode hooks that can be employed. This section details each hook and offers examples of what you could potentially do with each. The hooks are explained in the order that they fire in the code, meaning that PreProcessBatch is the first hook to fire, whereas FinalizeBatch is the last.

General Information

All of these hooks must reside in the BatchEvents class which needs to be located at the base level of the form package. For example, if my form app package were named ‘G\_FORM\_HIRE’, each of my peoplecode hooks would reside in the G\_FORM\_HIRE:BatchEvents app class. Also, none of these hooks are required; if a certain hook has not been defined, the delivered tool functionality will continue.

PeopleCode Hook Definitions

PreProcessBatch

PreProcessBatch fires immediately after clicking the Process button.

Parameters:

* BatchEvent Object

Returns:

* Nothing

Examples of when to use this hook:

* Any adjustments that need to be made to the entire batch file
* Validation that needs to be done to the entire batch file

PreProcessLine

PreProcessLine fires right before each row in the csv is processed.

Parameters:

* LineRecord

Returns:

* Nothing

Examples of when to use this hook:

* Validation to make sure the line is valid
* Adjusting any values on the line

PostFormCreate

While a csv row is being processed, a form is created. PostFormCreate fires immediately after the form object has been instantiated.

Parameters:

* LineRecord
* Form object

Returns:

* Nothing

PopulateField

PopulateField fires after PostFormCreate.

Parameters:

* BatchRecord
* SetupRecord
* Form object

Returns:

* Nothing

Examples of when to use this hook:

* Applying special formatting to text values from csv
* Translating csv values to acceptable form values
* Field-specific logic

PopulateGrid

PopulateGrid fires when a form contains a grid segment and population of said grid begins.

Parameters:

* BatchRecord
* SetupRecord
* Form Object

Returns:

* Nothing

Examples of when to use this hook:

* Overriding delivered grid population functionality

FinalizeForm

FinalizeForm fires when the processing for each row in the csv is complete.

Parameters:

* Form object

Returns:

* Boolean (true = this method overrides delivered FinalizeForm code, false = this method is additive to the delivered FinalizeForm code

Examples of when to use this hook:

* Email notifications on form error/success
* Final form adjustments

FinalizeBatch

FinalizeBatch fires at the very end of processing the batch. It is the last thing that happens before the tool’s grid is populated with the results from processing the batch.

Parameters:

* BatchEvent object

Returns:

* Nothing

Examples of when to use this hook:

* Email notifications on batch completion