Managing monthly deliverables reports

31 July 2023

# Contents

[Contents 2](#_Toc141697033)

[Background to this document 3](#_Toc141697034)

[End-to-end process 3](#_Toc141697035)

[Managing data in JIRA 3](#_Toc141697036)

[JIRA filters 3](#_Toc141697037)

[Labels are empty 3](#_Toc141697038)

[Parent / Epic is empty 4](#_Toc141697039)

[Sample monthly report 4](#_Toc141697040)

[Generating the report 4](#_Toc141697041)

[Pre-requisites 4](#_Toc141697042)

[Downloading the code 5](#_Toc141697043)

[Using git 5](#_Toc141697044)

[Downloading direct 5](#_Toc141697045)

[Extracting the necessary data from JIRA 6](#_Toc141697046)

[Running the Python application 6](#_Toc141697047)

## 

# Background to this document

The purpose of this document is to provide an overview of the process that we follow to generate monthly deliverable reports for the HMRC Online Trade Tariff.

Monthly deliverable reports constitute a list of all of the stories that have been delivered in the prior month, based on what has been set to “Done” during the most recent billing month.

The reason for doing all of this is that HMRC need to see evidence of what we have produced, and we need to be able to back up what we have done with documentary evidence.

These monthly deliverable reports also feed in to the monthly governance reports that are needed by the HMRC Commercial teams.

# End-to-end process

## Managing data in JIRA

There are possibly more automated ways in which to do this, but the process we follow is pretty quick and it works.

It uses two features of stories to generate and organise the report:

* Epics
* Labels

Epics are used to group together stories in the report, in the same way in which they are used to group Epics together in JIRA

Labels are used to identify the month in which the story was completed. These labels are defined by us, and use the format:

ott\_report\_20xx\_mmm

where xx is the year (e.g. 23) and mmm is the month name (e.g. june, july). At the start of each month, we create a new label.

### JIRA filters

There are three JIRA filters that are used to help to generate this report, as follows:

#### Labels are empty

This identifies where there is a story in the JIRA project that:

* is in a **DONE** state
* is set to **DONE** in recent months, since ticket number HOTT-2100, which is when this process was first set up.
* has **no label** against it. As per the description above, the label (month and year) is used to identify a story to be pulled into a report.

#### Parent / Epic is empty

* this identifies when a story exists but has no epic set against it.
* An epic is needed to produce the deliverables report, as it groups the stories together according to that epic.

#### Sample monthly report

And finally, this is the filter which actually produces the raw data that is needed to go into the monthly deliverables report.

project = HOTT AND key > 'HOTT-2100' AND status in (Done) AND labels = ott\_report\_2023\_july ORDER BY Parent ASC, updated DESC, created DESC

The JQL (Jira query language) above identifies:

* all stories later than HOTT-2100
* which are DONE
* and have a label that matches the current month, e.g. in this case ott\_report\_2023\_july (July 2023)

These filters are all shared with those who need them, and are owned by the DM.

## Generating the report

### Pre-requisites

You need to have the following tools installed on your local machine to be able to run these reports.

| What | Why |
| --- | --- |
| Git | This is needed to clone the latest version of the code onto your local machine. Alternatively, you may just download the ZIP version on the data into your local machine, unzip it and take it from there, but git is the easier and preferred way. |
| Python 3.x | You need Python 3.xx (latest version is 3.11.4 and is preferred, but not needed) – must be Python3 or above though.  Python3 will need to be on your path – guidance on how to ensure that Python is on your path is best found on the web; IT support may be needed to help to implement this. |
| Terminal or Command Prompt | Needed for running the application, unless you have Visual Studio Code, which is excellent at being a combine code editor and executor, with embedded terminal. |

### Downloading the code

There are two ways to download the code:

#### Using git

This is the preferred usage, as it allows you to continue to download updates to the application, or make changes and update the application, which is not possible unless you have local access to the repo.

* Go to the parent folder for the folder into which you want to install the application.
* Type the following:  
    
  git clone <https://github.com/trade-tariff/deliverables-report-writer>  
    
  which is the command to take a copy of the repo from where it is located on GitHub.

#### Downloading direct

If you do not have GitHub, then you can just take a copy of the entire repository, without using git, by downloading a copy of the repo from GitHub.

* Go to <https://github.com/trade-tariff/deliverables-report-writer> in a web browser.
* Click on the green “Code” button.
* Select “Download ZIP” from the dropdown menu underneath.

A screenshot of a computer

Description automatically generated

* Save the ZIP file in the parent of the folder where you want to install the application.
* UnZIP the file.

The rest of the instructions are the same, whether you have used git or not, with the only caveat that if you download and do not use git, then you are not connected to the repo if there are any changes.

### Extracting the necessary data from JIRA

* Load up the sample monthly report in JIRA by selecting the filter.
* From the “Export” button in the top right of the screen, select “Export CSV (all fields)”
* Save the report with the filename 2306 Sample monthly report (JIRA).csv in the /resources/jira folder of the project that you have accessed from GitHub, where the 2306 refers to the year and month to which the report refers.

A screenshot of a computer

Description automatically generated

That’s all that is needed in JIRA for this process.

## Running the Python application

* Go to the folder in which the application has been downloaded.
* Open up a command prompt, or open the application in Visual Studio Code[[1]](#footnote-1).
* As a **one-off** you will need to create a local virtual environment that has all of the necessary Python packages in it.
* Python uses packages to shortcut common tasks that would otherwise take aeons to write, e.g. the generation of a Word document, which is the purpose of this application.
* To create the virtual environment (**a one-off task**), use the following command, at the command prompt, either in Terminal or iTerm2 (Mac), or the Windows Command Prompt, or within Visual Studio Code.  
    
  python3 -m venv venv/
* This will create a subfolder called venv inside your project folder – you will only need to do this once, not multiple times. The purpose of this is to generate a local environment to this project that has access to all of the right packages, which will be installed in a subsequent step.
* Next, activate the virtual environment that you have created – you will need to do this every time you run the application, but it is only one line of code. You need to do this before you install the packages, as the packages will then be installed into the venv folder, and accessible by the application.  
    
  On a Mac, type source venv/bin/activate  
    
  On Windows using the Command Prompt: path\to\venv\Scripts\activate.bat  
    
  On Windows using PowerShell: path\to\venv\Scripts\Activate.ps1
* Then install the required packages – this is also a **one-off task**. The packages have been broken into two groups:
* those needed for development
* those needed for ‘production’, i.e. running the application.
* Strictly speaking you only really need those that are required for production, not the development equivalents, but there is no harm in having both, and if you are using VS Code, it will make the experience better.
* To install the **development** packages, do the following at the command prompt:  
    
  pip3 install -r requirements-dev.txt
* To install the **production** packages, do the following at the command prompt:  
    
  pip3 install -r requirements.txt

pip is the package installer for Python by the way.

### Environment variable setting

Copy the `env.sample` file to `.env` and then populate with required environment variables.

* `governance\_folder` - set to the folder to which reports are to be copied / stored permanently. If this is not set, then the file will not be copied.
* `write\_story\_points` - Set to `1` to write a story points column, or `0` (or omit) to not write the story points column.

### JIRA fields of interest

|  |  |
| --- | --- |
| Column | About |
| 0 | Summary, i.e. the title of the story |
| 1 | Issue key (unique ID)| |
| 81 | Parent summary, i.e. the name of the epic to which the story belongs. |

## Usage

Execute the application via `python3 write.py`

This:

* looks for the latest CSV file in the `/resources/jira` folder.
* writes an equivalent `.docx` Word document to the `report` folder, using the template stored in the `/resources/template` folder.
* copies the file to the destination folder specified in the .env file (`governance\_folder` key)

1. Visual Studio Code is brilliant at running and managing Python applications all in one place, and is recommended. [↑](#footnote-ref-1)