JiraWLUploader

users guide

The application was primarily created to export user’s work-log from excel to JIRA. The main reason for creating this application was to provide logs for the agency and the client without additional manual work.

**Major changes:**

* 2016/12 Supporting JIRA with basic authentication
* 2017/02 UI to delete worklogs from JIRA
* 2019/04 Added support for JIRA’s token based authentication
* 2020/05 Added support for Azure DevOps 7pace Timetracker

1. Contents

[1 The application 2](#_Toc40093982)

[1.1 Setting up an excel file 2](#_Toc40093983)

[1.1.1 Codes inside special JIRA columns 2](#_Toc40093984)

[1.2 Configuration of worklog API endpoints 2](#_Toc40093985)

[1.3 Uploading worklogs 3](#_Toc40093986)

[1.3.1 Rules 4](#_Toc40093987)

[1.3.2 Fail-safe 4](#_Toc40093988)

[2 Setting up JIRA account 4](#_Toc40093989)

[2.1 Generate ApiToken 4](#_Toc40093990)

[3 Setting up Azure DevOps 7Pace 5](#_Toc40093991)

# The application

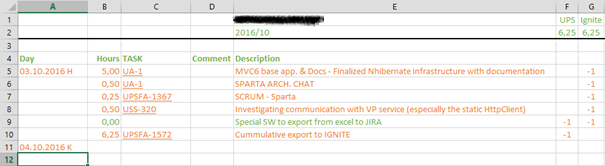
## Setting up an excel file

**The excel file must have a static structure, as provided in the sample below.**

* A 1st column – date (if no date is written, the 1st date above that row is used)
* B 2nd column – hours (numeric value)
* C 3rd column – the JIRA task ID (or Azure work item ID)
* E 5th column – the text for the work log

There are also **special columns** for each JIRA server (contains HTTP status codes or -1 if no upload is wanted) – see col. F & G in the screenshot.

We might add extra header rows, as currently the application will search for the string “day” in column “A” to detect from which point to parse data.



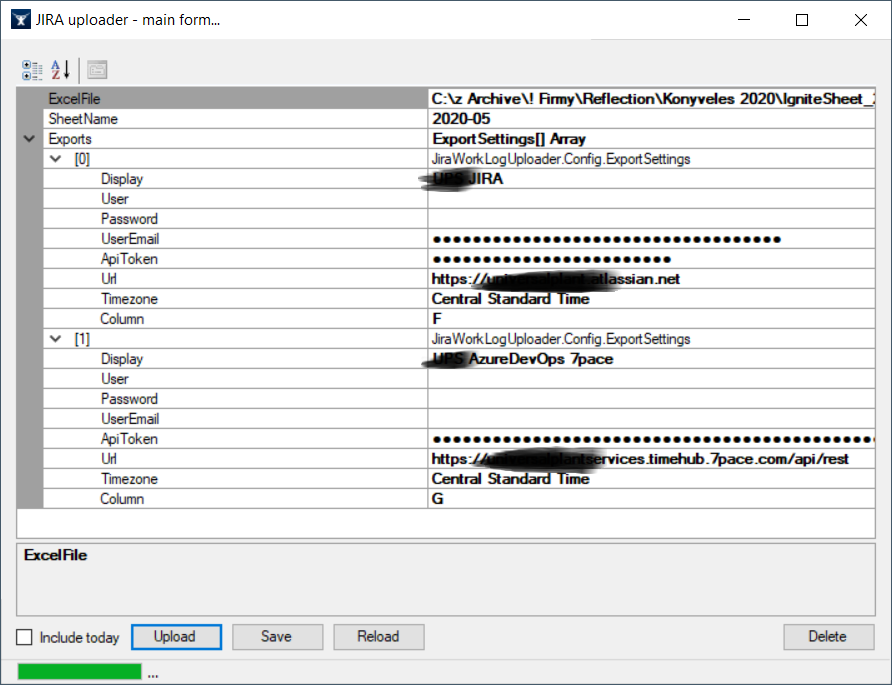
### Codes inside special JIRA columns

* 0 or empty = write log
* -1 = do not write log on the corresponding server
* greater than 0 = HTTP result code – whereas (200 to 300 = OK) = 201 is actually the “work log added” result on JIRA, other systems might use plain HTTP 200 instead

## Configuration of worklog API endpoints

**Minimal configuration:**

* **ExcelFile** – File must not be locked (opened elsewhere) while working with this app – also note: the app will create a copy of the sheets before writing upload results, so best to keep XLSX in a specific folder (also that folder should be accessible by the account running this application).
* **SheetName** – Once the file is selected and not blocked, the app will fetch sheet names. We can split logs on a monthly basis, each month – separate sheet.
* **User, Password, UserEmail, ApiToken – are containing user related security details and are protected.**
* **Url** – which is the URL to the JIRA
* **TimeZone** – this is the JIRA’s time-zone, must be set as we must consider time-shift when upload to get the entries with the desired date/time.
* **Column** – is the column’s name in excel where the upload status is stored (-1, 0, 404, 201)



The application uses a slightly customized **PropertyGrid** to display settings. Settings are stored in **App.json**

**It’s important to have JIRA server time-zone set correctly – otherwise the logged time will be shifted!**

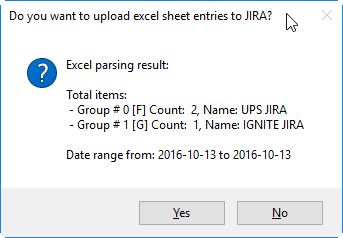
Save = save JSON

Reload = reload JSON

The COLUMN is an excel column name, and it should point to the column where the status codes for the given JIRA server will be stored. In most cases user does not need to log work to multiple JIRA’s so there will be only 1 JIRA setting and 1 status-code column – so when using the sample EXCEL as a template, it’ll be the column: F

## Uploading worklogs

The application will do a backup of excel first (into the same directory) – the backup will be created with a timestamp in the name.

The upload process first will parse the selected sheet, then display the result.  


After confirming the result, upload will start. It’s not super-fast, it just works…

### Rules

**Future logs are never exported.**

**Today logs can be exported by checking the “include today” checkbox.**

Row are ignored when:

* No hours are filled
* No task ID is present
* No description is filled
* The corresponding JIRA column has other than 0 or empty value

### Fail-safe

The in-memory excel is modified while uploading (result codes are written)…

Before writing out modified excel (only when change is present) a backup file will be created… Backup is done with failsafe, so unlimited retries are possible on error.

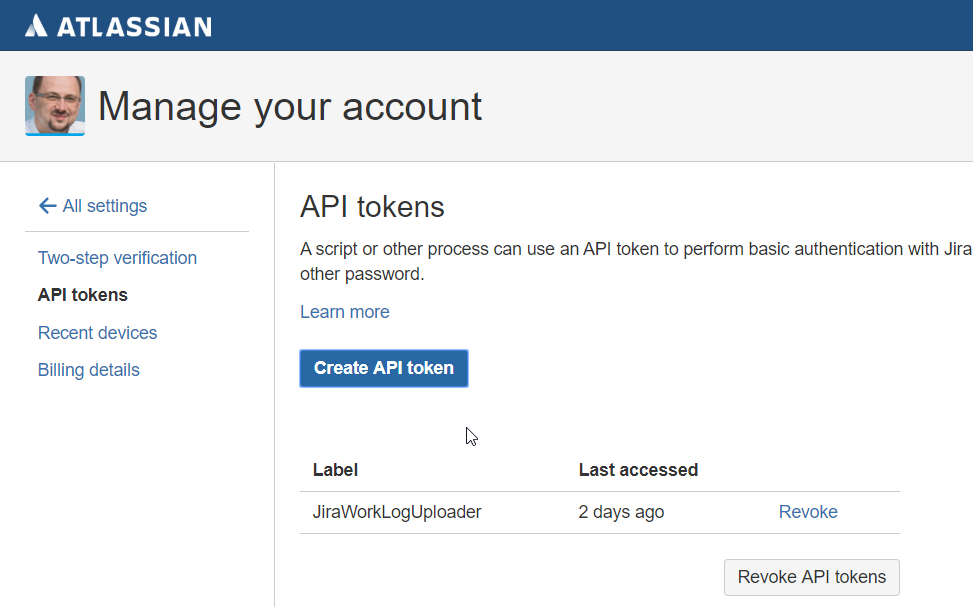
The same failsafe is applied for the new excel file. This was needed for the case that if someone accidentally re-opened the original excel while upload was in progress…so he now has the possibility to close it and allow the application to write the results into it (anyhow changes done by the user will be replaced)…

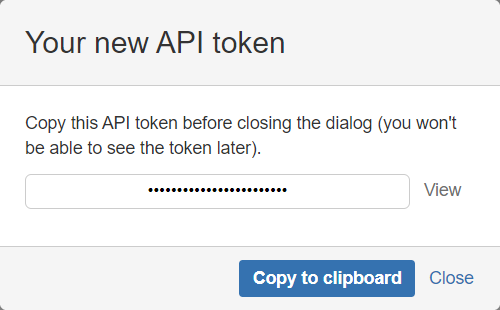
# Setting up JIRA account

The way of authentication has changed. Authorization headers are used, where the **UserEmail** and the **ApiToken** is necessary. The old User / Password authentication is not supported anymore.

## Generate ApiToken

1. Log in to JIRA.
2. Visit <https://id.atlassian.com/manage/api-tokens>.
3. Press **Create API token** and on the pop-up **enter a name** for the token – see picture below.
4. When the token is created, a dialog will open. **Make sure you copy over the token from the dialog, as it will not be shown any more.** Use this token in the application settings. Make sure you save the settings (credentials will be encoded via a machine key, so the config is not usable when copied over to a different computer or when windows is reinstalled).





# Setting up Azure DevOps 7Pace

Make sure you generate an API key in the 7pace Timetracker – settings – reporting API. This will be used layer inside the application settings.

By clicking the “Reporting API Reference” you will be redirected to the REST **documentation**.

