IssueSyncTool

v. 0.0.1

Tran Duy Ngoan

01.11.2024

Contents

Chapter 1

Introduction

The **IssueSyncTool** is a command-line utility designed to streamline issue synchronization across various tracking systems, including **GitHub**, **Jira**, **GitLab** and **IBM RTC**.

Its primary objective is to automate and simplify the integration and synchronization of issues between these platforms, enabling efficient tracking and planning for teams that work with multiple tools.

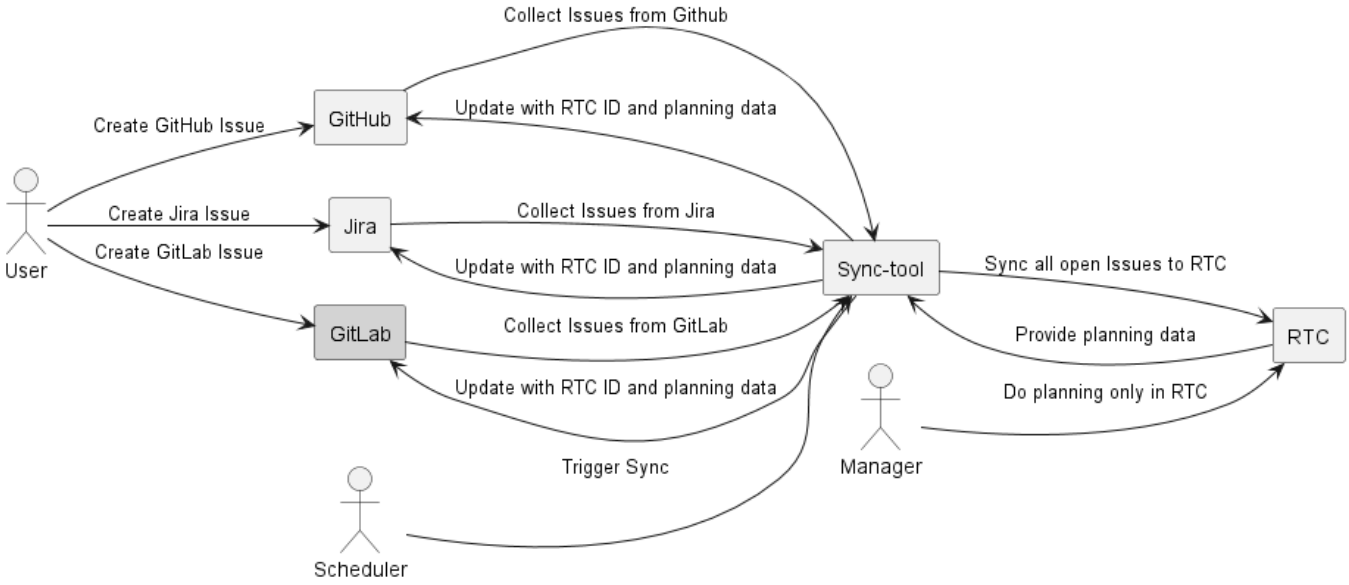

Figure 1.1: Tool's use case

1.1 Use Cases

- **Multi-Tool Teams:** For teams using a combination of GitHub, Jira, and RTC for issue management, this tool acts as a bridge to consolidate data.
- **Planning and Reporting:** Synchronization ensures managers and stakeholders have a centralized view of issues for effective planning.
- **Automated Workflows:** With scheduled triggers, the tool eliminates manual synchronization efforts, saving time and reducing errors.

1.2 Benefits

- **Automation:** Reduces manual synchronization overhead.
- **Consistency:** Ensures data integrity across platforms.
- **Customizable:** Flexible configurations to suit various project needs.

- **Centralized Planning:** Aligns all issues with RTC, the central planning tool.

Chapter 2

Description

2.1 Tool features

The **IssueSyncTool** facilitates seamless integration and synchronization between multiple issue tracking platforms. The main operations include:

1. Configuration Parsing
   - Reads the JSON configuration file to understand the synchronization scope and behavior.
2. Issue Collection
   - Fetches issues from GitHub, Gitlab and Jira based on the specified conditions.
   - Uses user mappings to ensure issues are associated correctly across platforms.
3. Issue Update
   - Updates the source issues with RTC IDs and planning data after synchronization.
4. Synchronization to RTC
   - Creates or updates work items in RTC with the collected issues.
   - Includes planning data provided by RTC.

2.2 Tool usage

Use below command to get tools's usage:

```
IssueSyncTool -h
```

The tool's usage should be showed as below:

```
usage: IssueSyncTool (Tickets Sync Tool) [-h] --config CONFIG [--dryrun] [--csv] [-v]

IssueSyncTool sync ticket|issue|workitem between tracking systems such as Github Issue, ↩
    ↪ JIRA and IBM RTC

optional arguments:
  -h, --help       show this help message and exit
  --config CONFIG  path to configuration json file
  --dryrun         if set, then just dump the tickets without syncing
  --csv            if set, then store the sync status to csv file sync_status.csv
  -v, --version    version of the IssueSyncTool
```

Sample command to run **IssueSyncTool** with the configuration JSON file and save sync status as csv file:

```
IssueSyncTool --config your-config-file --csv
```

2.3 JSON Configuration File

The tool uses a JSON configuration file to define synchronization behavior. Below is an explanation of the sample configuration:

2.3.1 Source and Destination Platforms

```
{
   "source": ["github", "gitlab", jira"],
   "destination": ["rtc"]
   ...
}
```

This configuration specifies GitHub and Jira as sources and RTC as the destination for synchronization.

2.3.2 Tracker Configurations

```
{
   ...
   "tracker": {
      "github": {
         "project" : "test-fullautomation",
         "token": "your-PAT",
         "repository": [
            "python-jsonpreprocessor"
         ],
         "condition": {
            "state": "open",
            "exclude": {
               "assignee": "empty",
               "labels": "0.13.1"
            }
         }
      },
      ...
   }
   ...
}
```

Above code is sample configuration of Github tracker which contain the information about:

- **Project:** Github project `"test-fullautomation"`
- **Token:** A personal access token for authentication.
- **Repositories:** List of repositories to collect issues from.
- **Condition:** Define the condition (query) to collect the issues.
  - `"state"` : Syncs only issues in the specified state, e.g., `"open"` .
  - `"exclude"` : Specifies negative conditions. For example:
    - `"assignee": "empty"` : Excludes issues with no assignee.
    - `"labels": "0.13.1"` : Excludes issues labeled `"0.13.1"` .

The other tracker can be configured as the same way.

2.3.3 User Mapping

User mapping ensures that the correct user is assigned in the synchronization process across different platforms. In the configuration file, each user is mapped to their corresponding accounts across GitHub, Jira, Gitlab and RTC. This mapping helps to ensure that the right assignee is applied to issues in the appropriate tracker system.

- The `user` section of the configuration file specifies the mapping between the users' names in GitHub, Gitlab, Jira and RTC.
- This ensures that the correct user is set as the assignee in each platform when syncing issues.
- For instance, when syncing an issue from Jira to RTC, the tool will automatically assign the same user (as per the mapping) to the issue in RTC.
- If the user has different usernames across platforms (e.g., "githubUser" in GitHub, "jiraUser" in Jira, and "rtcUser" in RTC), the tool ensures the correct mapping is applied so that all systems reflect the same assignee.

Example configuration:

```
{
   ...
   "user": [
      {
         "name": "Tran Duy Ngoan",
         "github": "ngoan1608",
         "jira": "ntd1hc",
         "gitlab": "ntd1hc",
         "rtc": "ntd1hc"
      },
      ...
   ]
}
```

In this example:

- The user `Tran Duy Ngoan` is mapped to `ngoan1608` in GitHub, `ntd1hc` in Jira, and `ntd1hc` in RTC.
- When syncing issues between GitHub, Jira, and RTC, the tool ensures that issues assigned to `ngoan1608` in GitHub and `ntd1hc` in Jira will be assigned to `ntd1hc` in RTC, ensuring consistent user data across all platforms.

Chapter 3

rtc_client.py

3.1 Function: get_xml_tree

Parse xml object from file. issuesynctool-rtc-client-rtcclient ==========

Imported by:

```
from IssueSyncTool.rtc_client import RTCClient
```

3.1.1 Method: get_filedAgainst

3.1.2 Method: get_info_from_url

3.1.3 Method: login

Authenticate and establish a session with RTC. issuesynctool-rtc-client-rtcclient-get-workitem ----------

3.1.4 Method: update_workitem

3.1.5 Method: update_workitem_state

3.1.6 Method: update_workitem_action

3.1.7 Method: create_workitem

Chapter 4

sync_issue.py

4.1 Function: write_csv_files

4.2 Function: process_cli_argument

4.3 Function: process_configuration

4.4 Function: process_new_issue

Process to create new issue on destination tracker and update original issue's title with destination issue's id.

- New issue's description is consist of original issue url and its description.

- Assignee is get from issuesynctool-sync-issue-process-sync-issues ==========
Update source (original) issue due to information from appropriate destination one.

Defined sync attributes:

- 'Title': add issue ID as prefix e.g '[123] Ticket title'when creating on destination tracker
- 'Story point': when planning existing issue on destination tracker
- 'Version': when planning existing issue on destination tracker

Update destination issue due to information from source.

Defined sync attributes:

- 'Status': status is synced from original ticket, not allow to update directly on destination tracker

4.5 Function: SyncIssue

4.6 Class: Logger

Imported by:

```
from IssueSyncTool.sync_issue import Logger
```

Logger class for logging message.

4.6.1 Method: config

Configure Logger class.

Arguments:

- `output_console`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  Write message to console output.
- `output_logfile`
  / *Condition*: optional / *Type*: str / *Default*: None /
  Path to log file output.
- `dryrun`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  If set, a prefix as 'dryrun' is added for all messages.

Returns:
(*no returns*)

4.6.2 Method: log

Write log message to console/file output.
Arguments:

- `msg`
  / *Condition*: optional / *Type*: str / *Default*: '' /
  Message which is written to output.
- `color`
  / *Condition*: optional / *Type*: str / *Default*: None /
  Color style for the message.
- `indent`
  / *Condition*: optional / *Type*: int / *Default*: 0 /
  Offset indent.

Returns:
(*no returns*)

4.6.3 Method: log_warning

Write warning message to console/file output.
Arguments:

- `msg`
  / *Condition*: required / *Type*: str /
  Warning message which is written to output.
- `indent`
  / *Condition*: optional / *Type*: int / *Default*: 0 /
  Offset indent.

Returns:
(*no returns*)

4.6.4 Method: log_error

Write error message to console/file output.

- `msg`
  / *Condition*: required / *Type*: str /
  Error message which is written to output.
- `fatal_error`
  / *Condition*: optional / *Type*: bool / *Default*: False /
  If set, tool will terminate after logging error message.
- `indent`
  / *Condition*: optional / *Type*: int / *Default*: 0 /
  Offset indent.

Returns:
(*no returns*)

Chapter 5

tracker.py

5.1 Class: Status

Imported by:

```
from IssueSyncTool.tracker import Status
```

5.1.1 Method: normalize_issue_status

5.1.2 Method: get_native_status

5.2 Class: Ticket

Imported by:

```
from IssueSyncTool.tracker import Ticket
```

Normalized Ticket with required information for syncing between trackers issuesynctool-tracker-ticket-update ----------

Update issue on tracker with following supported attributes:

- title
- assignee
- labels

5.2.1 Method: is_synced_issue

Verify whether the ticket is already synced or not.

It bases on the title of issue, it should contain destination ID information. E.g [1234] Title of already synced ticket issuesynctool-tracker-trackerservice ==========

Imported by:

```
from IssueSyncTool.tracker import TrackerService
```

Abstraction class of Tracker Service issuesynctool-tracker-trackerservice-connect ----------

Method to connect to tracker issuesynctool-tracker-trackerservice-get-ticket ----------

Method to get single ticket by id from tracker issuesynctool-tracker-trackerservice-get-tickets ----------

Method to get all tickets which satisfied given condition|query issuesynctool-tracker-trackerservice-create-ticket ----------

Method to create a new ticket on tracker system issuesynctool-tracker-trackerservice-exclude-issue-by-condition ----------

Process to verify whether the given issue is satisfied the exclude conditions issuesynctool-tracker-jiratracker ==========

Imported by:

```
from IssueSyncTool.tracker import JiraTracker
```

5.2.2 Method: connect

5.2.3 Method: get_ticket

5.2.4 Method: get_tickets

5.2.5 Method: create_ticket

5.2.6 Method: update_ticket

5.3 Class: GithubTracker

Imported by:

```
from IssueSyncTool.tracker import GithubTracker
```

5.3.1 Method: connect

5.3.2 Method: get_tickets

5.3.3 Method: get_ticket

5.3.4 Method: create_ticket

5.3.5 Method: update_ticket

5.4 Class: GitlabTracker

Imported by:

```
from IssueSyncTool.tracker import GitlabTracker
```

Tracker client to integrate with issues on Gitlab.

Except, get_tickets which allow to get issues from gitlab, group and project level, other method requires project information to interact properly with inside issues. issuesynctool-tracker-gitlabtracker-connect ----------

5.4.1 Method: get_ticket

5.4.2 Method: get_tickets

5.4.3 Method: create_ticket

5.4.4 Method: update_ticket

5.5 Class: RTCTracker

Imported by:

```
from IssueSyncTool.tracker import RTCTracker
```

5.5.1 Method: connect

5.5.2 Method: get_ticket

5.5.3 Method: get_tickets

5.5.4 Method: get_plannedFor

5.5.5 Method: update_ticket_state

5.5.6 Method: create_ticket

5.5.7 Method: update_ticket

5.6 Class: Tracker

Imported by:

```
from IssueSyncTool.tracker import Tracker
```

5.6.1 Method: create

5.6.2 Method: get_support_trackers

Chapter 6

user.py

6.1 Class: User

Imported by:

```
from IssueSyncTool.user import User
```

6.2 Class: UserManagement

Imported by:

```
from IssueSyncTool.user import UserManagement
```

6.2.1 Method: get_user

Chapter 7

Appendix

About this package:

Table 7.1: Package setup

Setup parameter	Value
Name	IssueSyncTool
Version	0.0.1
Date	01.11.2024
Description	Tool to synchronize issues between trackers (Github Issue, Gitlab Issue, RTC, JIRA)
Package URL	python-issue-sync-tool
Author	Tran Duy Ngoan
Email	Ngoan.TranDuy@vn.bosch.com
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 8

History

0.0.1	11/2024
Initial version	

IssueSyncTool.pdf

Created at 13.12.2024 - 15:29:21

by GenPackageDoc v. 0.41.1