EWM Command Line Tool

Table of Contents

Overview

ewm.py and ewm_wrapper.py are command line scripts to interact with an EWM repository. An example repository can be found here: https://jazz07.rchland.ibm.com:13443/jazz/web. For a quick intro to the tool, see the Quick Start section.
These tools ultimately call a version of the underlying rtcwi.py tool developed by the Systems DevOps team (found here: https://github.ibm.com/SystemsDevOpsSolutions/rtcwi-cli) but abstract away some of the details of working with this tool like login, project specific settings, installing custom modules, and working with Python3.

Which tool should I use?

1. ewm.py – use this if you have python3 setup as the default python installation in your environment.

```
$ python --version
Python 3.6.12
```

2. ewm_wrapper.py – **use this if you do NOT have python3 setup** by default in your environment and don't want to install/setup a python3 installation.

Quick Start

The tools reside in /esw/bin/bld or can be clone locally via a Git repository see Installation.

Test if you can authenticate to the tool. The tool should print your w3 id. If authentication to EWM fails, there are a couple different ways to provide login details to the tool see Setup Credentials.

```
$ ewm.py whoami
Joshua.Andersen1@ibm.com
```

View an STG Defect

```
$ ewm.py view --id 276281
{
    "Activation": "Unassigned",
    ...
    "Found In": "Unspecified",
    "HW Platform": "n/a",
    "Id": "276281",
    ...
}
```

Run a Query.

```
$ ewm.py runquery "Active STG Defects I Am Subscribed To"
{
    "headers": [
        "Type",
        "Universal ID",
        "Id",
        "Summary",
        ...
    ],
    "results": [
        {
            "Id": "282739",
            ...
            "Summary": "test - feature type",
            "Type": "STG Defect",
            "Universal ID": "PE009H6M"
        },
        {
            "Id": "276281",
            ...
            "Summary": "Code Update Signing from Signing Server",
            "Type": "STG Defect",
            "Universal ID": "SW522645"
        }
```

```
    ]
}
```

Add a note/comment to a defect.

```
$ ewm.py addnote --id 282739 --message "addnote test1"
```

View rtcwi.py help message

```
$ ewm.py help
Usage: rtcwi.py [options] subcommand

Options:
      ...

Available Subcommands:
        login:  Login to the repository
        help:   Display help. Optionally specify the subcommand to display help f
or.
        create: Create a work item
        display:        Displays a work item
        modify: Modify a work item
        whoami: Shows ID of user signed in to RTC
        CLEAR:  Removes configuration files.
        setcwe: Set the default values to use for repository and project.
        listqueries:    List queries in the project area.
        runquery:       Run query.
        subscribe:      Subscribe yourself or other users to a work item.
        unsubscribe:    Unsubscribe yourself from a work item.
        link:   Add a link from one work item to another.
        unlink: Remove a link from a work item.
        addcomment:     Add a comment to a work item.
        list:   List information about categories, actions, etc... See help for m
ore details.
        browser:        Open the specified work item in a web browser.
        search: Search RTC Project for strings contained in work items.
        logout: Log out of a RTC Repository.
        swat-
catowners: Use the SWAT Servicelayer REST APIs to retrieve an RTC Project categor
y owners.
```

Installation

The tools can be found in:

1. /esw/bin/bld
2. https://github.ibm.com/power-devops/ewm

If cloning a version of the tool, use the following command:

```
git clone --recurse-submodules git@github.ibm.com:power-devops/ewm.git
```

When trying to run the tool and you get the following error:

```
ModuleNotFoundError: No module named 'requests'
```

You will need to install this commonly used Python module for your Python distribution. If you do not have authority to install the requests module with your distribution, you could look into Python's virtual environment feature https://docs.python.org/3.6/tutorial/venv.html.

General Information

This tool provides a generic interface which can be used by various issue tracking tools. This way, users can get used to calling the tools with a common syntax and improve code reuse in the future. The action is the first positional argument when calling the tool.

Issue tracking tools will support the following generic functions or "actions".

1. Create Issue = create
2. Modify Issue = modify
3. View Issue = view
4. Add note/comment = addnote

Additional actions can be added to expose additional functionality for the issue tracking tool. For example, rtcwi.py uses subcommands to interact with EWM in various ways. These subcommands can be used directly in ewm.py, substituting the subcommand for the action value. For ewm.py, the generic actions will do a translation which ultimately calls the rtcwi.py specific subcommand.

For contents of wifile, see Work Item JSON File Details.

Create Issue Syntax:

```
ewm.py create --type "STG Defect" --wifile workitem_template.json --
attributes KEY1: VALUE1, KEY2: VALUE2
```

Modify Issue Syntax:

```
ewm.py modify --id 123456 --attributes Description : "details about defect"
```

View Issue Syntax:

```
ewm.py view --id 123456
```

Add Note/Comment Syntax:

```
ewm.py addnote --id 123456 --message "built into b0505a2119.1010"
```

Setup Credentials

To authenticate to the EWM Server, four details need to be provided.

1. W3 ID aka your email address.
2. W3 ID password
3. Repository URL
4. Project Name

ewm.py attempts to automatically determine these details in the following order (below). If a detail is duplicated between multiple methods like netrc and environmental variable, the higher precedence (larger number in the following list) will be used.

The recommended approach is to use the tool's default project and repository URL and then add an entry to your netrc file.

1. Parse tool config file, defaults_ewm.ini. This establishes a default Repository URL, Project Name, and authentication file (path to a file containing the clear text password).
2. Default username, USER@us.ibm.com, based on your USER environmental variable.
3. Parse your $HOME/.netrc file for an entry with machine name EWM.

```
# An example line in your $HOME/.netrc file
machine EWM login joshua.andersen1@ibm.com password YOURW3PASSWORD
```

4. Parse custom config INI file via command line. Make sure appropriate file permissions are setup on the config file if your password is present in it.

```
# Custom config contents
[ewm]
repository=https://jazz07.rchland.ibm.com:13443/jazz
project=Cognitive Systems Software Development
authentication_file=$HOME/private/password
password=YOURPASSWORD
id=joshua.andersen1@ibm.com
```

5. Set environmental variables:
    a. EWM_ID
    b. EWM_PASSWORD
    c. EWM_PROJECT
    d. EWM_REPOSITORY

Work Item JSON File Details

When trying to create or modify a work item, there are a variety of different attributes that can be changed. Each work item has a specific type (aka STG Defect, Change Record), and within that type, there are different attributes that can be changed. Certain attributes only support specific contents like STG Defect attribute Severity.

To get a list of supported Work Item types:

```
ewm.py list types
```

To get a list of attributes for a specific Work Item type:

```
ewm.py list attributes "STG Defect"
```

To get supported values for a specific attribute:

```
ewm.py list values Severity
```

Example contents of a work item JSON file:

```
{
    "Filed Against" : "fips_build",
    "Found In" : "Unspecified",
    "Phase Found" : "Unspecified",
    "Severity" : "3 - Moderate",
    "HW Platform" : "n/a",
    "System Name" : "n/a",
    "STG Defect Type" : "Feature",
    "Owned By": "MYEMAIL@ibm.com",
    "Summary": "test - work item template for creating a defect",
    "Description": "Testing to create an STG Defect from the ewm.py commandline t
ool"
}
```

When trying to create a new work item like STG Defect, certain attributes are required to properly create the new STG Defect, see *Figure 1: Example STG Defect*, the required fields are marked with an asterisk. The JSON file has a lower precedence and can be overridden by attributes explicitly stated on the command line. Note: the attribute KEY or VALUE that contains spaces (eg: “System Name”) need to be quoted on the command line to be properly parsed.

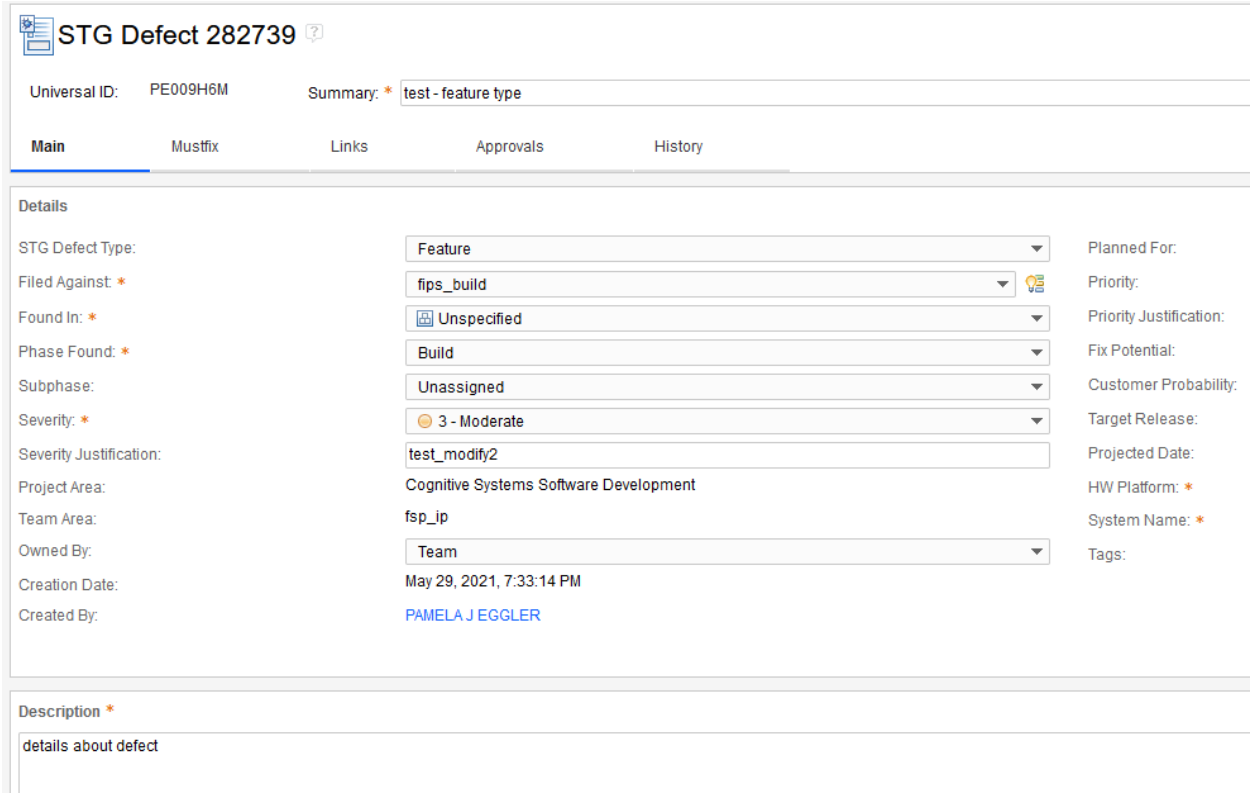

Figure 1: Example STG Defect

Command Line Interface (CLI) Usage

To simplify tool usage, most (except for create and modify) of the underlying rtcwi.py subcommands can be called with the same syntax from ewm.py. The command consists of 3 parts: EWM tool specific options EWM-OPTIONS, the desired ACTION, and the action specific ACTION-OPTIONS.

```
ewm.py [EWM-OPTIONS] ACTION [ACTION-OPTIONS]
```

To print the top-level help:

```
ewm.py --help
```

To print the ACTION help usage (documentation created by ewm.py) for how to run the specific action CLI:

```
ewm.py display --help
```

To learn more about the rtcwi.py subcommand use the "help" ACTION (documentation created by rtcwi.py):

```
ewm.py help runquery
```

Note: the EWM-OPTIONS need to be listed BEFORE the ACTION positional argument.

An example can be found here:
https://github.ibm.com/power-devops/ewm/blob/master/example_code/testEwm.pl

EWM-OPTIONS:

1. --verbose. Print extra information about the script.
2. --outfile OUTFILE. File to store rtcwi.py output.
3. --ewm-config EWM_CONFIG. Config file for program. Explicit override for default vaules.

ACTION:

Generic Action

create: create a new Work Item
Example:

```
ewm.py create --type "STG Defect" --wifile
./config/workitem_template.json --attributes "Owned By":
joshua.andersen1@ibm.com
```

modify: update a detail in a Work Item
Examples:

```
ewm.py modify --id 282739 --attributes "Found In": Unspecified,
Tags: "fwexec bob", "System Name": "new value", "Description":
"using a new description2"

ewm.py modify --id 282739 --attributes Tags: fwexec

ewm.py modify --id 282739 --wifile ./my_new_attributes.json
```

addnote: add a note to a Work Item (synonym for addcomment, syntax is a little different)
Example:

```
ewm.py addnote --id 282739 --message "addnote test 06/28/21"
```

view: get the contents of a Work Item
Example:

```
ewm.py view --id 282739
# Get specific attribute of Work Item
ewm.py view --id 282739 --attribute "Severity Justification"
ewm.py view --id 282739 --attribute Type
```

rtcwi.py Specific Action/Subcommand

login: authenticate to the tool from the established credentials (this is done automatically by ewm.py)

Example:

```
ewm.py login
```

help: print help information for the various rtcwi.py commands

Examples:

```
ewm.py help

ewm.py help runquery
```

display: get the contents of a Work Item

Example:

```
ewm.py display 282739
```

whoami: print the currently authenticated user

Example:

```
ewm.py whoami
```

CLEAR: remove rtcwi.py configuration files

Example:

```
ewm.py CLEAR
```

setcwe: store credentials for the current working environment

Example:

```
ewm.py setcwe
```

listqueries: list the queries available in the current project area

Example:

```
ewm.py listqueries
```

runquery: run a specific query

Example:

```
ewm.py runquery "Active STG Defects I Am Subscribed To"
```

```
ewm.py runquery "Active STG Defects I Am Subscribed To" --text-mode
```

subscribe: allow a user to be notified of Work Item updates

Example:

```
ewm.py subscribe 282739 joshua.andersen1@ibm.com
```

unsubscribe: remove a user from Work Item updates (can only remove self aka authenticated user)

Example:

```
ewm.py unsubscribe 282739
```

link: add a link to a Work Item. For supported link_types like “Related Artifacts”, run ewm.py list attributes “STG Defect”

Example:

```
ewm.py link 282739 "Related Artifacts" http://www.google.com
```

unlink: remove a link to a Work Item

Example:

```
ewm.py unlink 282739 "Related Artifacts" http://www.google.com
```

addcomment: add a note/comment to a Work Item (synonym for addnote)

Example:

```
ewm.py addcomment 282739 "Test addcomment 06/28/21"
```

list: List information about the RTC CLI environment or project area.

Examples:

```
ewm.py list cwe
ewm.py list attributes "STG Defect"
ewm.py list types
ewm.py list attributes "STG Defect"
ewm.py list values "Found In"
```

browser: launches a web browser (unsupported)

Example:

```
TBD
```

search: run a full text search for work items in an EWM Project.

Examples:

```
ewm.py search "Code Update Signing"

# limit to 4 results

ewm.py search "Code Update Signing" -m 4
```

logout: logs the user out of a repository.

Example:

```
ewm.py logout
```

swat-catowners: Use SWAT Servicelayer REST APIs to display an RTC project's categories and those category's owners (TBD).

Example:

```
TBD
```

Object Oriented Interface

ewm.py supports an Ewm class that can be used in Python3 scripts. An .ini configuration file can be passed during object creation to override the default settings. During object initialization, defaults are setup, and the user logins into the EWM repository.

An example can be found here:
https://github.ibm.com/power-devops/ewm/blob/master/example_code/testEwm_oo.py

Constructor:

```
import ewm
instance = ewm.Ewm()

# override default settings
instance = ewm.Ewm(config_file="/path/to/config/file.ini")
```

Methods:

create: create a new Work Item
Example:

```
instance.create(witype="STG Defect", wifile=attributes.json)
```

modify: update a detail in a Work Item
Example:

```
instance.modify(id="282739", wifile=attributes.json)
```

addnote: add a note to a Work Item (synonym for addcomment, syntax is a little different)
Example:

```
instance.addnote(id="282739", message="addnote test 06/28/21")
```

view: get the contents of a Work Item
Example:

```
data = instance.view(id="282739")
data = instance.view(id="282739", attribute="Found In")
```

login: authenticate to the tool from the established credentials (this is done automatically during object creation). The user is logged in for a set about of time; therefore, for long running jobs (example over a day), the user might need to relogin.

Example:

```
instance.login()
```

help: print help information for the various rtcwi.py commands

Examples:

```
data = instance.help()
data = instance.help(subcommand="runquery")
```

display: get the contents of a Work Item

Example:

```
data = instance.display(id="282739")
```

whoami: print the currently authenticated user

Example:

```
user = instance.whoami()
```

CLEAR: remove rtcwi.py configuration files

Example:

```
instance.CLEAR()
```

setcwe: store credentials for the current working environment. Settings are initially determined during object creation.

Example:

```
output = instance.setcwe()
output = instance.setcwe(repository="", project="", user="")
```

listqueries: list the queries available in the current project area

Example:

```
queries = instance.listqueries()
```

runquery: run a specific query

Example:

```
data = instance.runquery(query_name="Active STG Defects I Am
Subscribed To")
```

subscribe: allow a user to be notified of Work Item updates

Example:

```
current_subscribers = instance.subscribe(id="282739",
[joshua.andersen1@ibm.com])
```

unsubscribe: remove a user from Work Item updates (can only remove self aka authenticated user)

Example:

```
current_subscribers = instance.unsubscribe(id="282739")
```

link: add a link to a Work Item. For supported link_types like "Related Artifacts", run ewm.py list attributes "STG Defect"

Example:

```
instance.link(id="282739", link_type="Related Artifacts",
work_item_ids=["http://www.google.com"])
```

unlink: remove a link to a Work Item

Example:

```
instance.unlink(id="282739", link_type="Related Artifacts",
work_item_ids=["http://www.google.com"])
```

addcomment: add a note/comment to a Work Item (synonym for addnote)

Example:

```
instance.addcomment(id="282739", comment="addcomment test
06/28/21")
```

list: List information about the RTC CLI environment or project area.

Examples:

```
instance.list(topic="cwe")

instance.list(topic="attributes", qualifiers="STG Defect")

instance.list("types")

instance.list(topic="attributes", qualifiers="STG Defect")

instance.list(topic="values", qualifiers="Found In")
```

browser: launches a web browers (unsupported)

Example:

```
TBD
```

search: run a full text search for work items in an EWM Project.

Examples:

```
results = instance.search(search_text="Code Update Signing")

results = instance.search(search_text="Code Update Signing",
max_num=4)
```

logout: logs the user out of a repository.

Example:

```
instance.logout()
```

swat-catowners: Use SWAT Servicelayer REST APIs to display an RTC project's categories and those category's owners (TBD).

Example:

TBD