Migrate to Zephyr Scale on Jira Data Center via APIs

In this document we will walk through the necessary process to migrate to Zephyr Scale. Below will also be attached an example script and a video walking through the process. The example script and video will use Zephyr Squad to Zephyr Scale on Jira Data Center as the example.

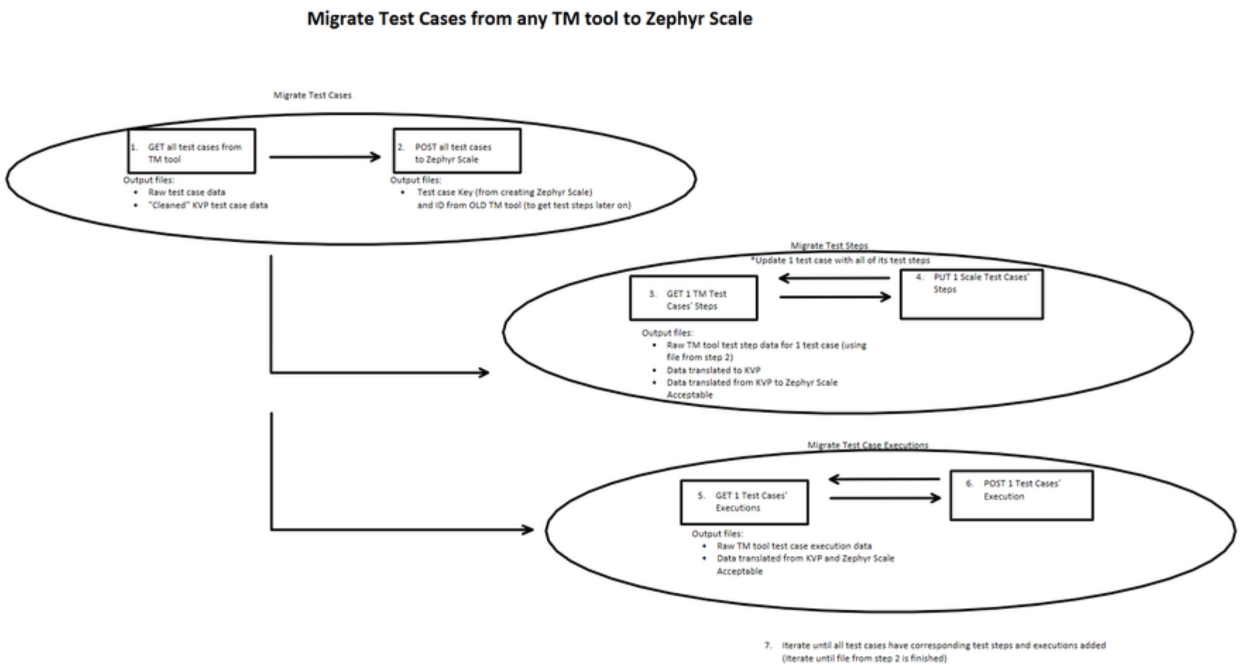
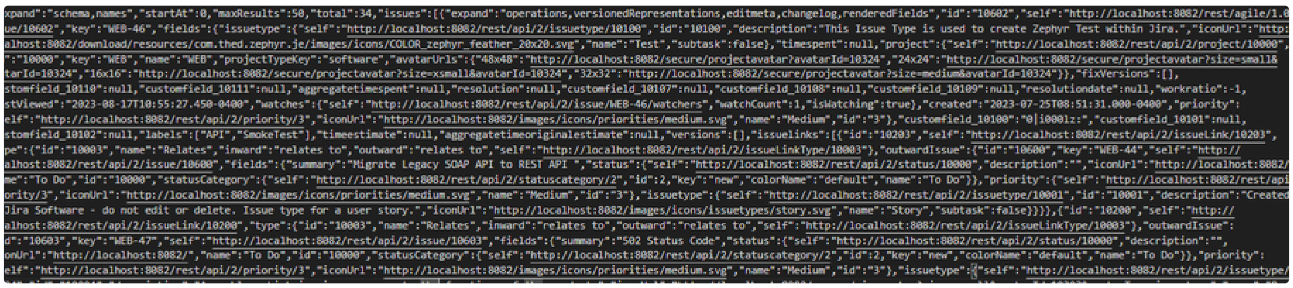

Migrate Test Cases from any TM tool to Zephyr Scale
Migrate Test Cases
1. GET all test cases from TM tool
2. POST all test cases to Zephyr Scale
Output files:
• Raw test case data
• "Cleaned" KVP test case data
Output files:
• Test case Key (from creating Zephyr Scale) and ID from OLD TM tool (to get test steps later on)
Migrate Test Steps
*Update 1 test case with all of its test steps
3. GET 1 TM Test Cases' Steps
4. PUT 1 Scale Test Cases' Steps
Output files:
• Raw TM tool test step data for 1 test case (using file from step 2)
• Data translated to KVP
• Data translated from KVP to Zephyr Scale Acceptable
Migrate Test Case Executions
5. GET 1 Test Cases' Executions
6. POST 1 Test Cases' Execution
Output files:
• Raw TM tool test case execution data
• Data translated from KVP and Zephyr Scale Acceptable
7. Iterate until all test cases have corresponding test steps and executions added (iterate until file from step 2 is finished)

Figure 1: High-level Diagram of Migration Process

Step 1: Get All Test Cases From the Test Management Tool

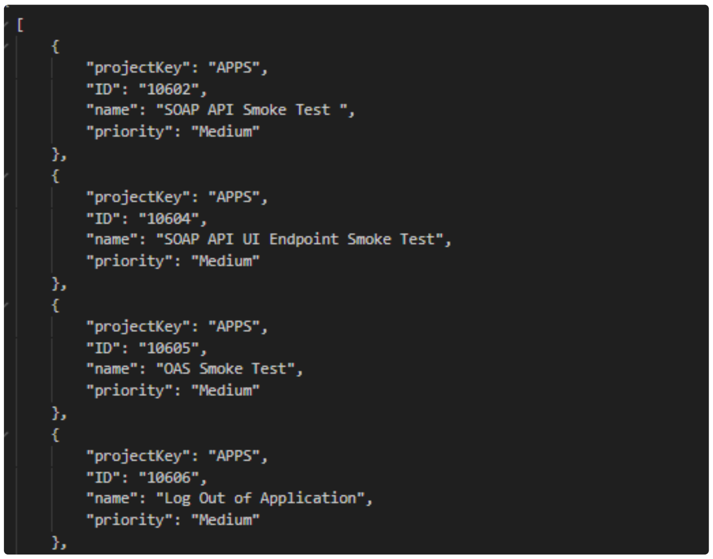
First we must get the available test cases from the test management tool. Generally the ability exists to query all available test cases, or to export a subset of test cases. Store this information in JSON so we can parse it later.
API Used: `"http://localhost:8082/rest/agile/1.0/epic/none/issue?jql=project = WEB AND issuetype = Test ORDER BY createdDate ASC"`



```
xpand":"schema,names","startAt":0,"maxResults":50,"total":34,"issues":[{"expand":"operations,versionedRepresentations,editmeta,changelog,renderedFields","id":"10602","self":"http://localhost:8082/rest/agile/1.0
ue/10602","key":"WEB-46","fields":{"issuetype":{"self":"http://localhost:8082/rest/api/2/issuetype/10100","id":"10100","description":"This Issue Type is used to create Zephyr Test within Jira.","iconUrl":"http:
alhost:8082/download/resources/com.thed.zephyr.je/images/icons/COLOR_zephyr_feather_20x20.svg","name":"Test","subtask":false},"timespent":null,"project":{"self":"http://localhost:8082/rest/api/2/project/10000",
":"10000","key":"WEB","name":"WEB","projectTypeKey":"software","avatarUrls":{"48x48":"http://localhost:8082/secure/projectavatar?avatarId=10324","24x24":"http://localhost:8082/secure/projectavatar?size=small&
tarId=10324","16x16":"http://localhost:8082/secure/projectavatar?size=xsmall&avatarId=10324","32x32":"http://localhost:8082/secure/projectavatar?size=medium&avatarId=10324"}},"fixVersions":[],
stomfield_10110":null,"customfield_10111":null,"aggregatetimespent":null,"resolution":null,"customfield_10107":null,"customfield_10108":null,"customfield_10109":null,"resolutiondate":null,"workratio":-1,
stViewed":"2023-08-17T10:55:27.450-0400","watches":{"self":"http://localhost:8082/rest/api/2/issue/WEB-46/watchers","watchCount":1,"isWatching":true},"created":"2023-07-25T08:51:31.000-0400","priority":
elf":"http://localhost:8082/rest/api/2/priority/3","iconUrl":"http://localhost:8082/images/icons/priorities/medium.svg","name":"Medium","id":"3"},"customfield_10100":"0|i0001z:","customfield_10101":null,
stomfield_10102":null,"labels":["API","SmokeTest"],"timeestimate":null,"aggregatetimeoriginalestimate":null,"versions":[],"issuelinks":[{"id":"10203","self":"http://localhost:8082/rest/api/2/issueLink/10203",
pe":{"id":"10003","name":"Relates","inward":"relates to","outward":"relates to","self":"http://localhost:8082/rest/api/2/issueLinkType/10003"},"outwardIssue":{"id":"10600","key":"WEB-44","self":"http://
alhost:8082/rest/api/2/issue/10600","fields":{"summary":"Migrate Legacy SOAP API to REST API ","status":{"self":"http://localhost:8082/rest/api/2/status/10000","description":"","iconUrl":"http://localhost:8082/
me":"To Do","id":"10000","statusCategory":{"self":"http://localhost:8082/rest/api/2/statuscategory/2","id":2,"key":"new","colorName":"default","name":"To Do"}},"priority":{"self":"http://localhost:8082/rest/api
ority/3","iconUrl":"http://localhost:8082/images/icons/priorities/medium.svg","name":"Medium","id":"3"},"issuetype":{"self":"http://localhost:8082/rest/api/2/issuetype/10001","id":"10001","description":"Created
Jira Software - do not edit or delete. Issue type for a user story.","iconUrl":"http://localhost:8082/images/icons/issuetypes/story.svg","name":"Story","subtask":false}}}},{"id":"10200","self":"http://
alhost:8082/rest/api/2/issueLink/10200","type":{"id":"10003","name":"Relates","inward":"relates to","outward":"relates to","self":"http://localhost:8082/rest/api/2/issueLinkType/10003"},"outwardIssue":
d":"10603","key":"WEB-47","self":"http://localhost:8082/rest/api/2/issue/10603","fields":{"summary":"502 Status Code","status":{"self":"http://localhost:8082/rest/api/2/status/10000","description":"",
onUrl":"http://localhost:8082/","name":"To Do","id":"10000","statusCategory":{"self":"http://localhost:8082/rest/api/2/statuscategory/2","id":2,"key":"new","colorName":"default","name":"To Do"}},"priority":
elf":"http://localhost:8082/rest/api/2/priority/3","iconUrl":"http://localhost:8082/images/icons/priorities/medium.svg","name":"Medium","id":"3"},"issuetype":{"self":"http://localhost:8082/rest/api/2/issuetype/
```

Figure 2: Output File of Raw Test Case Data, Inclusive of All Test Cases

The next part is to use a parsing engine to parse the raw test case data, and output the fields we want to transfer over to key value pairs. We can add more key value pairs here if necessary. Be sure to save the reference ID to each test case so we can update the correct test case, with the correct test steps.



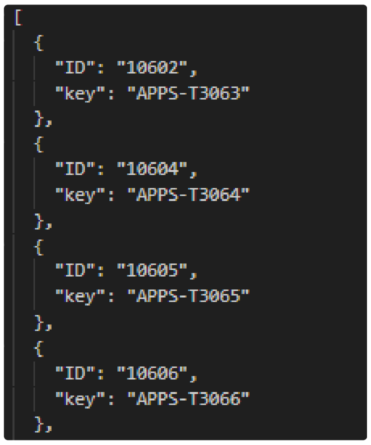
```
[
    {
        "projectKey": "APPS",
        "ID": "10602",
        "name": "SOAP API Smoke Test ",
        "priority": "Medium"
    },
    {
        "projectKey": "APPS",
        "ID": "10604",
        "name": "SOAP API UI Endpoint Smoke Test",
        "priority": "Medium"
    },
    {
        "projectKey": "APPS",
        "ID": "10605",
        "name": "OAS Smoke Test",
        "priority": "Medium"
    },
    {
        "projectKey": "APPS",
        "ID": "10606",
        "name": "Log Out of Application",
        "priority": "Medium"
    },
```

Figure 3: Output File of Test Case Data Converted to Key Value Pairs

Step 2: Post All Test Cases to Zephyr Scale

Now we have the necessary data, lets create test cases in Zephyr Scale. You can publish the whole array of test cases from Figure 3. If there was any rework to the naming conventions we can adjust that here. While we are publishing the test cases to Zephyr Scale, we note the test case key that is created from Zephyr. We then output a new file which contains the Zephyr Scale key and the reference ID to the test steps from the test management tool in key value pair, for each test case.

API Used: `"http://localhost:8082/rest/atm/1.0/testcase"`



```
[
  {
    "ID": "10602",
    "key": "APPS-T3063"
  },
  {
    "ID": "10604",
    "key": "APPS-T3064"
  },
  {
    "ID": "10605",
    "key": "APPS-T3065"
  },
  {
    "ID": "10606",
    "key": "APPS-T3066"
  },
```

Figure 4: Output File of ID and Zephyr Scale Key

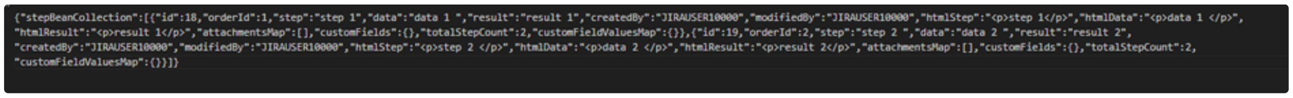
The file from Figure 4 becomes our iteration file. We will update a single Zephyr Scale test case with the corresponding test steps, using the Key and ID respectively.

Note: Iteration starts here

Step 3: Get a Single Test Case’s Test Steps from the Test Management tool

Using Figure 4’s ID, query the test steps for 1 test case. This data will be raw, so we likely will need to clean it up.

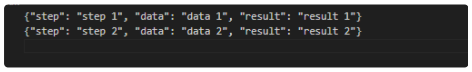
API Used: `"http://localhost:8082/rest/zapi/latest/teststep/{ID}"`



```
{"stepBeanCollection":[{"id":18,"orderId":1,"step":"step 1","data":"data 1 ","result":"result 1","createdBy":"JIRAUSER10000","modifiedBy":"JIRAUSER10000","htmlStep":"<p>step 1</p>","htmlData":"<p>data 1 </p>",
"htmlResult":"<p>result 1</p>","attachmentsMap":[],"customFields":{},"totalStepCount":2,"customFieldValuesMap":{}},{"id":19,"orderId":2,"step":"step 2 ","data":"data 2 ","result":"result 2",
"createdBy":"JIRAUSER10000","modifiedBy":"JIRAUSER10000","htmlStep":"<p>step 2 </p>","htmlData":"<p>data 2 </p>","htmlResult":"<p>result 2</p>","attachmentsMap":[],"customFields":{},"totalStepCount":2,
"customFieldValuesMap":{}}]}
```

Figure 5: Output File of Test Steps for 1 Test Case from Legacy Test Management Tool

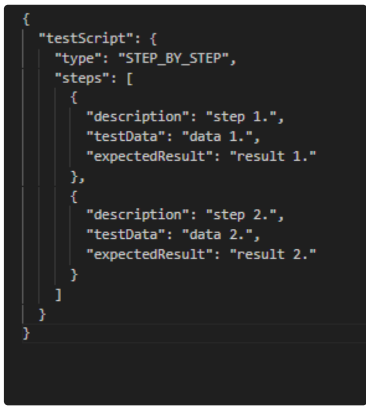
We then transform the data to key value pairs. If there was more data (like custom fields) or naming convention adjustment, we can add that here.



```
{"step": "step 1", "data": "data 1", "result": "result 1"}
{"step": "step 2", "data": "data 2", "result": "result 2"}
```

Figure 5: Output File of Transformed Step Data

Before we publish to Zephyr Scale we need to transform the key value pairs to a format Zephyr Scale will accept via API. Again if there was more data in Figure 5, you can add that in to the script here.



```
{
  "testScript": {
    "type": "STEP_BY_STEP",
    "steps": [
      {
        "description": "step 1.",
        "testData": "data 1.",
        "expectedResult": "result 1."
      },
      {
        "description": "step 2.",
        "testData": "data 2.",
        "expectedResult": "result 2."
      }
    ]
  }
}
```

Figure 6: Output File That Becomes
Payload of Step 4

Step 4: Update a Single Test Case’s Test Steps in Zephyr Scale

Using the output file as a payload, example shown in Figure 6, update the test case with the corresponding test steps in Zephyr Scale.

API Used: `"http://localhost:8082/rest/atm/1.0/testcase/{key}"`

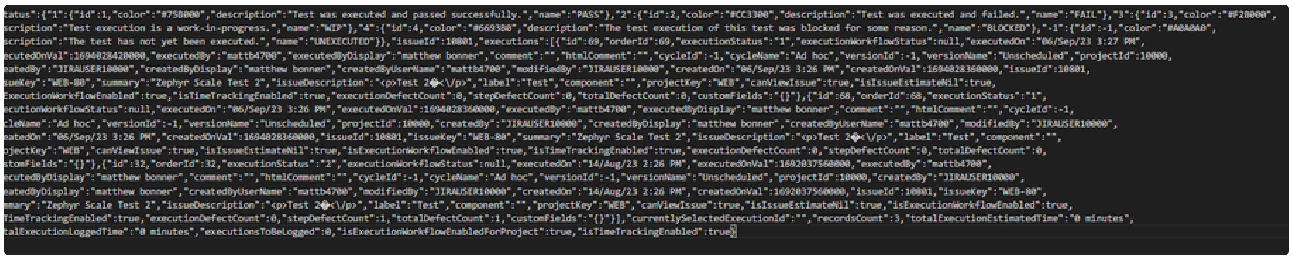
Step 5: Get All Test Executions for 1 Test Case

Now we use the ID that is stored in Figure 4 to query the correct executions per the right test case.

In order to create multiple executions at once we need to create a test run, which will give us a test cycle where we can POST multiple results to.

API Used: `"http://localhost:8082/rest/zapi/latest/execution?issueId={ID}"`

API Used: `"http://localhost:8082/rest/atm/1.0/testrun"`

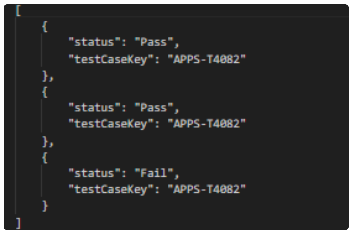


```
tatus":{"1":{"id":1,"color":"#75B000","description":"Test was executed and passed successfully.","name":"PASS"},"2":{"id":2,"color":"#CC3300","description":"Test was executed and failed.","name":"FAIL"},"3":{"id":3,"color":"#F2B000",
scription":"Test execution is a work-in-progress.","name":"WIP"},"4":{"id":4,"color":"#669380","description":"The test execution of this test was blocked for some reason.","name":"BLOCKED"},"-1":{"id":-1,"color":"#A0A0A0",
scription":"The test has not yet been executed.","name":"UNEXECUTED"}},"issueId":10801,"executions":[{"id":69,"orderId":69,"executionStatus":"1","executionWorkflowStatus":null,"executedOn":"06/Sep/23 3:27 PM",
ecutedOnVal":1694028420000,"executedBy":"mattb4700","executedByDisplay":"matthew bonner","comment":"","htmlComment":"","cycleId":-1,"cycleName":"Ad hoc","versionId":-1,"versionName":"Unscheduled","projectId":10000,
eatedBy":"JIRAUSER10000","createdByDisplay":"matthew bonner","createdByUserName":"mattb4700","modifiedBy":"JIRAUSER10000","createdOn":"06/Sep/23 3:26 PM","createdOnVal":1694028360000,"issueId":10801,
sueKey":"WEB-80","summary":"Zephyr Scale Test 2","issueDescription":"<p>Test 2�<\/p>","label":"Test","component":"","projectKey":"WEB","canViewIssue":true,"isIssueEstimateNil":true,
ExecutionWorkflowEnabled":true,"isTimeTrackingEnabled":true,"executionDefectCount":0,"stepDefectCount":0,"totalDefectCount":0,"customFields":"{}"},{"id":68,"orderId":68,"executionStatus":"1",
ecutionWorkflowStatus":null,"executedOn":"06/Sep/23 3:26 PM","executedOnVal":1694028360000,"executedBy":"mattb4700","executedByDisplay":"matthew bonner","comment":"","htmlComment":"","cycleId":-1,
cleName":"Ad hoc","versionId":-1,"versionName":"Unscheduled","projectId":10000,"createdBy":"JIRAUSER10000","createdByDisplay":"matthew bonner","createdByUserName":"mattb4700","modifiedBy":"JIRAUSER10000",
eatedOn":"06/Sep/23 3:26 PM","createdOnVal":1694028360000,"issueId":10801,"issueKey":"WEB-80","summary":"Zephyr Scale Test 2","issueDescription":"<p>Test 2�<\/p>","label":"Test","component":"",
ojectKey":"WEB","canViewIssue":true,"isIssueEstimateNil":true,"isExecutionWorkflowEnabled":true,"isTimeTrackingEnabled":true,"executionDefectCount":0,"stepDefectCount":0,"totalDefectCount":0,
stomFields":"{}"},{"id":32,"orderId":32,"executionStatus":"2","executionWorkflowStatus":null,"executedOn":"14/Aug/23 2:26 PM","executedOnVal":1692037560000,"executedBy":"mattb4700",
ecutedByDisplay":"matthew bonner","comment":"","htmlComment":"","cycleId":-1,"cycleName":"Ad hoc","versionId":-1,"versionName":"Unscheduled","projectId":10000,"createdBy":"JIRAUSER10000",
eatedByDisplay":"matthew bonner","createdByUserName":"mattb4700","modifiedBy":"JIRAUSER10000","createdOn":"14/Aug/23 2:26 PM","createdOnVal":1692037560000,"issueId":10801,"issueKey":"WEB-80",
mmary":"Zephyr Scale Test 2","issueDescription":"<p>Test 2�<\/p>","label":"Test","component":"","projectKey":"WEB","canViewIssue":true,"isIssueEstimateNil":true,"isExecutionWorkflowEnabled":true,
TimeTrackingEnabled":true,"executionDefectCount":0,"stepDefectCount":1,"totalDefectCount":1,"customFields":"{}"}],"currentlySelectedExecutionId":"","recordsCount":3,"totalExecutionEstimatedTime":"0 minutes",
talExecutionLoggedTime":"0 minutes","executionsToBeLogged":0,"isExecutionWorkflowEnabledForProject":true,"isTimeTrackingEnabled":true}
```

Figure 7: Raw Test Case Execution Data

Step 6: Post All Test Executions for 1 Test Case

We then translate the test case execution data to key value pairs, and make it Zephyr Scale acceptable. The output file, highlighted in Figure 8, is used as the payload when posting test case executions.
API Used: `'http://localhost:8082/rest/atm/1.0/testrun/{cycleKey}/testresults'`



```
[
    {
        "status": "Pass",
        "testCaseKey": "APPS-T4082"
    },
    {
        "status": "Pass",
        "testCaseKey": "APPS-T4082"
    },
    {
        "status": "Fail",
        "testCaseKey": "APPS-T4082"
    }
]
```

Figure 8: Cleaned Test Case Execution Data

Video Example using Zephyr Squad to Zephyr Scale on Jira Data Center

Video Conferencing, Web Conferencing, Webinars, Screen Sharing
Passcode: g^09=z#3

Script Example using Zephyr Squad to Zephyr Scale on Jira Data Center

```python
import requests
from requests.auth import HTTPBasicAuth
import json


################## Step 1: Get Zephyr Squad Test Cases via Jira API ###################################
### This creates 1 files containing the raw Zephyr Squad test case response, call it file1.

## Get all Zephyr Squad test cases, and output to a file

#This will query all tests per epic, per JQL expression
url = "http://localhost:8082/rest/agile/1.0/epic/none/issue?jql=project = WEB AND issuetype = Test ORDER BY cre
##Replace with Jira username
username = ""
##Replace with Jira Password
password = ""

##Output file path of the Zephyr Squad Test Cases
output_file_path = "1.txt"

#Use auth when creating session
session = requests.Session()
```

```python
session.auth = (username, password)

try:
    # Send GET request
    response = session.get(url)

    # Check if the request was successful
    if response.status_code == 200:
        # Parse the response content (you might need to adjust this depending on the response format)
        response_content = response.text

        # Save the response content to the output file
        with open(output_file_path, "w") as output_file:
            output_file.write(response_content)

        print("Response saved to", output_file_path)
    else:
        print("Request failed with status code:", response.status_code)
except requests.RequestException as e:
    print("An error occurred:", e)

################## Parse Zephyr Squad Test Case Data to Post to Zephyr Scale ##################################
### This creates 1 files containing the parsed Zephyr Squad test case data, call it file2.

## Transform raw Zephyr Squad GET test cases' response into key-value pairs of required test case information.
## We can add as many fields here as we want.

##File path of Zephyr Squad Test Cases
file_path = r"1.txt"
##New File that will output specific Key Value Pairs from file path
output_file_path = r"2.txt"
# Read JSON data from file
with open(file_path, 'r') as file:
    json_data = json.load(file)

# Extract projectKey, ID, Name, and priority for each issue
#Here is where we would specify any and all fields we want from Zephyr Squad to be pushed to Zephyr Scale
parsed_data = []
for issue in json_data["issues"]:
    projectKey = issue["fields"]["project"]["key"]
    ID = issue["id"]
    name = issue["fields"]["summary"]
    priority_info = issue["fields"].get("priority")
    priority = priority_info["name"] if priority_info else "No priority"


    parsed_data.append({
        #"projectKey": projectKey,
        "projectKey": "APPS",
        "ID": ID,
        "name": name,
        "priority": priority,


    })

# Write parsed data to output file
with open(output_file_path, 'w') as output_file:
    json.dump(parsed_data, output_file, indent=4)

print(f"Parsed data has been written to {output_file_path}")

################## Step 2: POST Zephyr Squad Test Cases to Zephyr Scale and Create Key-Value Pairs  ###########
### This creates multiple test cases in Zephyr Scale, however many are contained with the output of file2. Whil
### it is saving in a new file, the key value pairs of  Zephyr Squads test case IssueID and the newly created Z
### callit file2.5.


# POST Zephyr Scale Test Case's from file2

##File path that 2.txt was written to. You could use outputfile path variable above.
with open("C:\\ZScale\\ConverterReady\\ReallyConvertReady\\SimplifiedByClass\\2.txt", 'r')  as file: #load outp
    data = json.load(file)

##Zephyr Scale POST test cases API
url = "http://localhost:8082/rest/atm/1.0/testcase"


# Initialize a list to store the responses
responses = []

# Iterate through each object in the JSON array
for item in data:
    # Extract desired fields
    project_key = item["projectKey"]
    name = item["name"]

    # Create payload using extracted fields
    payload = {
        "projectKey": project_key,
        "name": name
    }

    # Send POST request with the payload and Basic Authentication
    response = requests.post(url, json=payload, auth=HTTPBasicAuth(username, password))

    # Check the response status and content
    if response.status_code == 201:
        response_data = response.json()
        key = response_data.get("key")
        responses.append({"ID": item["ID"], "key": key})
        print(f"POST request for {item['ID']} successful!")
        print("Response content:", response_data)
    else:
        print(f"POST request for {item['ID']} failed with status code:", response.status_code)
        print("Response content:", response.text)

## Save Key-Value pairs of Zephyr Squad Test Case IssueID (in order to get Zephry Squad Test Steps)
## and Newly Created Zephyr Scale Test Case Key(in order to update the proper test cases with the proper script
## into file2.5.

## Write Key and ID to an output file to use later
with open('2.5.txt', 'w') as output_file:
    json.dump(responses, output_file, indent=2)

print("Responses written to 2.5.txt")
```

```python

########################################################## Iteration Starts Here ##################################
################### Step 3: GET 1 Zephyr Squad Test Case's Test Steps and Step 4: PUT (update) Zephyr Scales Te
### This will GET 1 Zephyr Squad test case's test steps from file2.5, and save its test steps in file3. Then tr
### file4. Transform file 4 to the test steps format Zephyr Scale will accept, save that to file5. Update the Z
### and using the data saved as payload in file5. Iterate until file 2.5 is finished.

## Use file2.5 to iterate down Test case keys
##File path that 2.5txt was written to. You could use outputfile path variable above.
with open("C:\\ZScale\\ConverterReady\\ReallyConvertReady\\SimplifiedByClass\\2.5.txt", "r") as file:
    data = json.load(file)

## Step 3: Get a Single Test Case’s Test Steps from the Test Management tool
## Get Zephyr Squad Test Case Steps
# Create a session with basic authentication
session = requests.Session()
session.auth = (username, password)
for item in data:
        key = item["key"]
        ID = item["ID"]
        url = f"http://localhost:8082/rest/zapi/latest/teststep/{ID}"
        try:
            # Send GET request
            response = session.get(url)

            # Check if the request was successful
            if response.status_code == 200:
                # Parse the response content (you might need to adjust this depending on the response format)
## Write Zephyr Squad test step data to a file, file3
                response_content = response.text
                output_file_path = "3.txt"
                # Save the response content to the output file
                with open(output_file_path, "w") as output_file:
                    output_file.write(response_content)

                print("Response saved to", output_file_path)
            else:
                print("Request failed with status code:", response.status_code)
        except requests.RequestException as e:
            print("An error occurred:", e)

## Transform Raw Zephyr Squad Test Step Data to Key-Valye Pairs
## Save in file, file4
        # Replace this with the path to your JSON file
        json_file_path = "3.txt"
        output_file_path = "4.txt"  # Replace with your desired output file path

        # Read the JSON data from the file
        with open(json_file_path, "r") as json_file:
            json_data = json_file.read()

        # Parse the JSON data
        parsed_data = json.loads(json_data)

        # Extract and store "step", "data", and "result" values in key-value pairs
        step_data_pairs = []
        for step_entry in parsed_data["stepBeanCollection"]:
            step_data_pairs.append({
                "step": step_entry["step"].strip(),
                "data": step_entry["data"].strip(),
                "result": step_entry["result"].strip()
            })

        # Write the extracted key-value pairs to the output file
        with open(output_file_path, "w") as output_file:
            for pair in step_data_pairs:
                output_file.write(json.dumps(pair) + "\n")

        print("Extracted data saved to", output_file_path)
        # Replace this with the path to your JSON file
        input_file_path = "4.txt"
        output_file_path = "5.txt"  \

        # Read the JSON data from the file
        with open(input_file_path, "r") as input_file:
            extracted_data = input_file.readlines()

## Transform Key-Value Pairs to Zephyr Scale Acceptable
## Save trannsformed data in file5

        # Transform the extracted data
        transformed_data = {
            "testScript": {
                "type": "STEP_BY_STEP",
                "steps": []
            }
        }

        for entry in extracted_data:
            kvp = json.loads(entry)

            step_description = kvp["step"]
            data_description = kvp["data"]
            result_description = kvp["result"]

            step = {
                "description": step_description.strip(".") + ".",
                "testData": data_description.strip(".") + ".",
                "expectedResult": result_description.strip(".") + "."
            }

            transformed_data["testScript"]["steps"].append(step)

        # Write the transformed data to the output file
        with open(output_file_path, "w") as output_file:
            json.dump(transformed_data, output_file, indent=2)

        print("Transformed data saved to", output_file_path)

## Send PUT Request to Update Corresponding Zephyr Scale Test Case with Correct Test Steps
## Use file2.5 to get the correct Zephyr Scale test case key, and file5 for the payload of test steps.

#Need filepath declared for file 2.5.txt or variable
        with open("C:\\ZScale\\ConverterReady\\ReallyConvertReady\\SimplifiedByClass\\2.5.txt", "r") as file:
            data = json.load(file)
        key = item["key"]
        ID = item["ID"]

        # Replace these with your actual values
#Step 4: Update a Single Test Case’s Test Steps in Zephyr Scale
#PUT Request to update Zephyr Scale Test Case
        url = f"http://localhost:8082/rest/atm/1.0/testcase/{key}"
        file_path = "5.txt"  # Replace with the actual file path

        # Read the payload from the file
        with open(file_path, "r") as file:
            payload_data = file.read()

        # Set up Basic Authentication
        auth = HTTPBasicAuth(username, password)

        # Send the PUT request with the payload
        response = requests.put(url, data=payload_data, auth=auth, headers={"Content-Type": "application/json"}

        if response.status_code == 200:
            print("PUT request successful")
        else:
            print("PUT request failed with status code:", response.status_code)


## Step 5: Get Zephyr Squad Test Case Executions
## Use file2.5 to get the correct Zephyr Squad executions POST the right test Executions .

        with open("C:\\ZScale\\ConverterReady\\ReallyConvertReady\\SimplifiedByClass\\2.5.txt", "r") as file:
            data = json.load(file)
        key = item["key"]
        ID = item["ID"]

        url = f"http://localhost:8082/rest/zapi/latest/execution?issueId={ID}"
        ##Replace with Jira username
        username = ""
        ##Replace with Jira Password
        password = ""

        ##Output file path of the Zephyr Squad Test Cases
        output_file_path = "6.txt"

        #Use auth when creating session
        session = requests.Session()
        session.auth = (username, password)

        try:
            # Send GET request
            response = session.get(url)

            # Check if the request was successful
            if response.status_code == 200:
                # Parse the response content (you might need to adjust this depending on the response format)
                response_content = response.text

                # Save the response content to the output file
                with open(output_file_path, "w") as output_file:
                    output_file.write(response_content)

                print("Response saved to", output_file_path)
            else:
                print("Request failed with status code:", response.status_code)
        except requests.RequestException as e:
            print("An error occurred:", e)


        with open("6.txt", "r") as file:
            data = file.read()

        # Load the JSON data
        data_dict = json.loads(data)

        key_value_pairs = []

        # Define a mapping of execution status values to their descriptions
        status_mapping = {
            "1": "Pass",
            "2": "Fail",
            "3": "WIP",
            "4": "Blocked",
            "-1": "Unexecuted"
        }

        # Iterate through the executions
        for execution in data_dict["executions"]:
            execution_status_id = execution["executionStatus"]
            execution_status = status_mapping.get(execution_status_id, "Unknown")

            # Create a key-value pair for each execution
            key_value_pair = {
                "status": execution_status,
                "testCaseKey": key
            }

            # Append the key-value pair to the list
            key_value_pairs.append(key_value_pair)

        # Write the list of key-value pairs to an output file as a JSON array
        with open("7.txt", "w") as output_file:
            json.dump(key_value_pairs, output_file, indent=4)  # indent for pretty formatting

        print("Data extracted and saved to '7.txt'.")

        payload = {
            "name": "Migrating Executions From Legacy Tool",
            "projectKey": "APPS"

        }
        url = 'http://localhost:8082/rest/atm/1.0/testrun'

        username = ""
        ##Replace with Jira Password
        password = ""
        auth = HTTPBasicAuth(username, password)

        # Send the POST request with the payload data
        response = requests.post(url, auth=auth, json=payload)

        # Check the response
        if response.status_code == 201:
            data = response.json()  # Parse the JSON response
            cycleKey = data["key"]  # Extract the "key" value and store it in cycleKey
        else:
            print(f"Request failed with status code {response.status_code}")

    ##Step 6: POST executions to Zephyr Scale

        url = f'http://localhost:8082/rest/atm/1.0/testrun/{cycleKey}/testresults'

        file_path = "7.txt"  # Replace with the actual file path

            # Read the payload from the file
        with open(file_path, "r") as file:
            payload_data = file.read()

        # Set up Basic Authentication
        auth = HTTPBasicAuth(username, password)

        # Send the PUT request with the payload
        response = requests.post(url, data=payload_data, auth=auth, headers={"Content-Type": "application/json'

        if response.status_code == 201:
            print("Post request successful")

        else:
            print("POST request failed with status code:", response.status_code)
            print(url)

#### Could add test case attachments by:
## GET https://zephyrsquadserver.docs.apiary.io/#reference/attachmentresource/delete-attachment/get-single-atta
## POST /testcase/{testCaseKey}/attachments https://support.smartbear.com/zephyr-scale-server/api-docs/v1/

### Would have to be added to the iteration
```