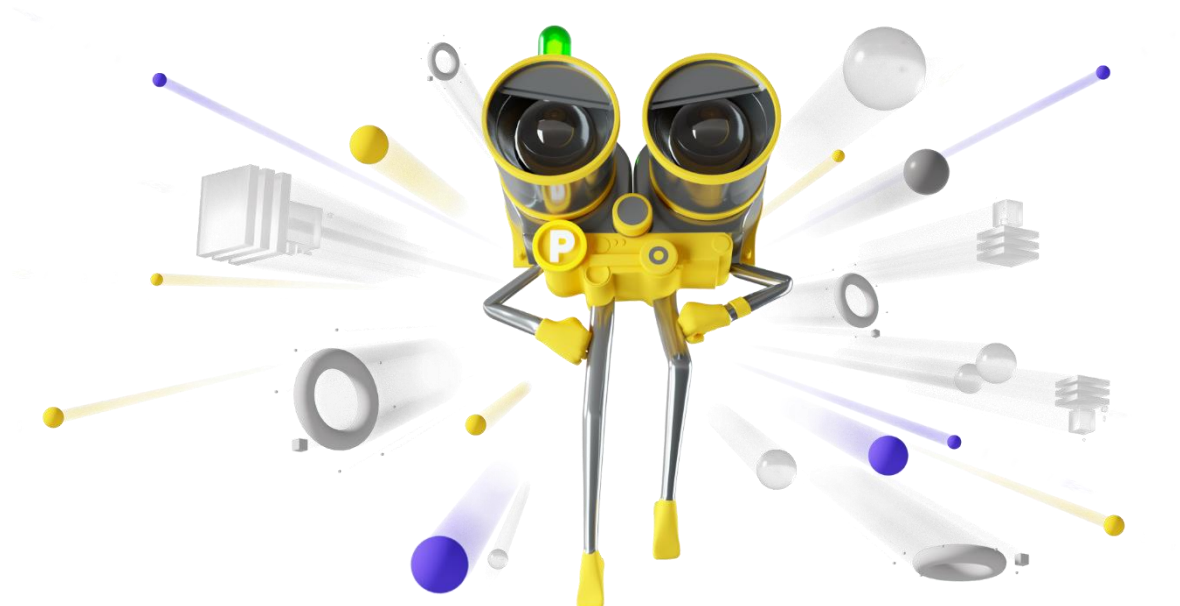


INTEGRATING YOUR TEST AUTOMATION TOOL

Guidelines & Examples

VERSION 2.2



CONTENTS

Writing the Automation Connector	2
The Connector Logic - Summary.....	3
The Cycle's Sync Settings Tab.....	5
General Guidelines for Using the API	8
Working with Tokens	9
Login - API	11
Connector Registration - API.....	15
Get Tasks - API	17
Create automation tests in Panaya - API.....	20
Working with TCodes.....	22
Report task status - API.....	25
Report the execution result of a test - API.....	27
A summary of the Connector logic and the APIs	30

WRITING THE AUTOMATION CONNECTOR

1. You can write the connector in any language and technology, as long it can communicate over HTTPS.
2. The machine on which the connector is running should be open to the internet (to communicate with Panaya) and must be able to access the automation tool.
3. The connector should be up and running 24*7 in a dedicated environment. It is up to the customer to monitor it and ensure it is running (e.g., add a watchdog to launch it when it crashed or get an alert when it is not running).
4. The connector should be robust (e.g., handle communication issues with Panaya or with the automation tool, catch exceptions, retry in case of failures).
5. Here is an example of a non-robust connector -
The connector receives 10 tests for execution and begins running the first test but crashes before progressing to the second test in the list. Suppose there is no internal mechanism that will validate that 9 tests did not run. In such a case, these 9 tests will not be sent by Panaya for execution again since Panaya's connector sent the request to perform 10 tests and is unaware of any run failures on the automation side.

THE CONNECTOR LOGIC - SUMMARY

1. Call the Login API and get a token. See below for more explanation about using tokens.
2. Call the API to register your computer. You will need to supply the connector UID (can be found in the Panaya Settings) and give a name to your connector.
This step should be done only once.
3. Every 1-5 minutes, call Panaya and ask for tasks.
4. If no tasks, Sleep for 1-5 minutes and ask again for tasks.
5. If there are tasks, iterate and handle each one:
 - Report to Panaya that you started handling the task.
 - Handle it.
 - Report to Panaya that you completed the task.
6. There can be two types of tasks:

Type 1 - request to sync tests:

- This type of task is being created when a user clicks the "Sync" button on a cycle of type "Generic Automation."
- Panaya will pass the Cycle's Key-Value pairs, as defined by the user in the Automation Cycle.
- It is up to the connector to understand the Key-Value pairs, contact the automation tool, understand which tests need to be synced into Panaya, and then call the Panaya API to create these tests.
- The connector should find the "delta" of un-synced tests, meaning that it should only create tests that were not synced yet to Panaya. In order to understand the "delta," the connector/the automation tool should have a persistency layer to keep which tests were already synced to Panaya. For example, you can add a flag to every test on the automation tool's side (e.g., in the DB) to track if the test was already synced or not.
- As part of the test creation, you should send to Panaya all information that will be needed to execute the test (Key-Value pairs for each test)
- There is no option to delete a test that was already synced to Panaya.
- There is no option to update a test that was already synced to Panaya.

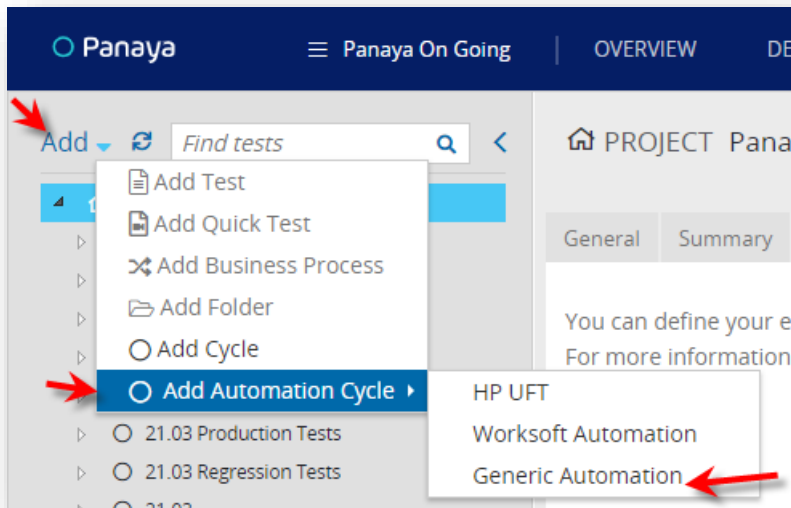
- The tests in Panaya are only a "reference" to the real test of the tool, so if a test script was updated in the tool, there is no need to update Panaya since the reference is still valid.

Type 2 - request to execute tests (the 'Create Playlist' action in Panaya):

- This type of task is being created when a user creates a "Playlist" in Panaya. A Playlist is a one-time request to execute automation tests.
- Panaya will pass for each test an ID and all the Key-Value pairs of the test.
- The connector should start the execution of the test and report to Panaya that the execution was started.
- Once the execution is completed, the connector should report the result of the execution (Pass/fail) and can send an evidence file.

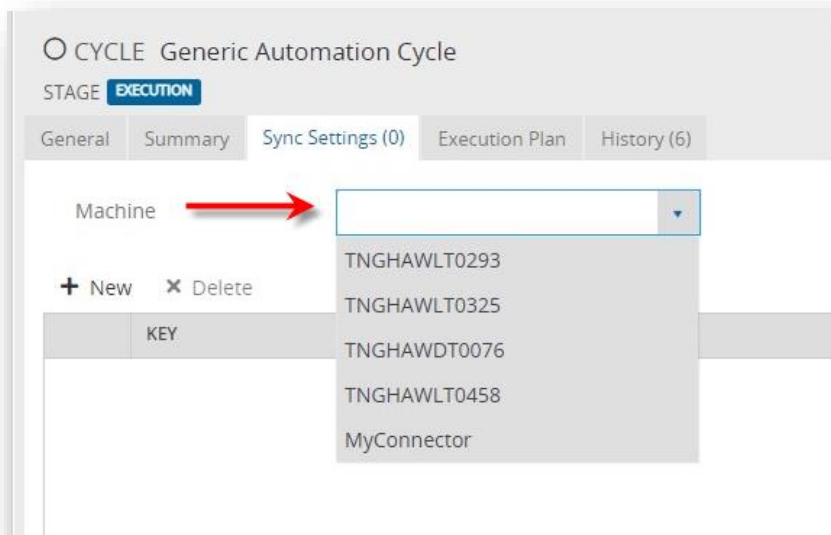
THE CYCLE'S SYNC SETTINGS TAB

To start working with Panaya Automation integration, you first need to create a cycle of type "Generic Automation" in Panaya. The purpose of this cycle is to hold a reference for your automation tests.

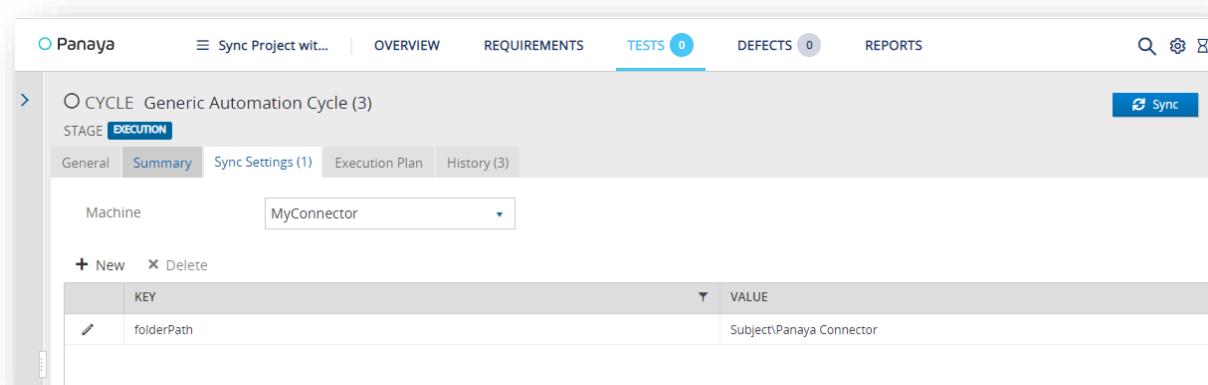


Note: Under an "Automation Cycle," you cannot add tests manually. The automation cycle is a placeholder for tests that are created by the connector (via API) only.

In the Sync Settings tab, you should select a machine:



The machine name is the "ComputerID" you provided in the "Connector Registration API" (See below).
After selecting the Machine (=connector), you can provide pairs of Key-Value:



It is not mandatory to add any key-value. If you do not add any, the "sync tests" task that you will get in the "Get Tasks" API will simply tell you "you need to sync your tests." No further details.

But in most cases, you want to provide your connector with information that will guide it which tests are needed to be synced to the cycle.

Example #1

You have many automated tests managed by your automation tool.

You want to sync into Panaya only part of these tests. So how will the connector know which part you wish to sync?

For that, you can add the following pair: Key= "Folder", value = "\\folder1\folder2\my tests". Panaya does not "understand" the meaning of this pair. It is for the connector to understand and act upon them. Now, when the user clicks on "sync" in Panaya, Panaya will add all the key-value pairs in the "Get tasks" API response, and the connector will see that pair and will know that it needs to sync only a specific folder. The user can change the value of the folder in Panaya, and when clicking "sync" again, the connector will get a different folder to sync.

Example #2

Assume you have a field in every automated test called "Dev Group." You want to sync only the

tests that belong to dev group X. You can add a key=" Dev Group" and value = "X," and then the connector should interpret that key-value and should understand that it needs to sync only tests that belong to that dev group.

Example #3

You want to sync only new tests created after a given date. You can add a key "created after," and in the value, write a date. Then the connector needs to understand this Key and apply the necessary filters in the automation tool in order to fetch and sync only tests that were created after that given date.

Bottom line – it is up to you to decide which "Keys" to use, based on your use cases. For each key to be effective, you need to "teach" the connector what to do with that key and value.

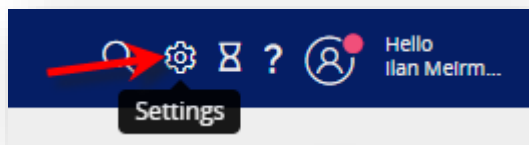
GENERAL GUIDELINES FOR USING THE API

1. In the API examples below, we are using the "my.panaya.com" domain.
If your Panaya account is using "emea.panaya.com," you should use that domain instead of "my.panaya.com."
2. Please note that the API URLs are using https://, not http://
3. In the API examples below, we are using a dummy connector ID, e.g.:

```
https://my.panaya.com/api/v1/systems/342b0bb1_1539de169f6__791e/automation/client/register
```

You should use your specific connector ID.

To find your connector ID, go to Panaya Settings:



Click on "Configuration," and toward the bottom of the page, you will see the Connector ID:



Use the "Copy Connector ID to Clipboard" button to copy the Connector ID to your clipboard.

WORKING WITH TOKENS

1. For security and authentication reasons, every API call (except the login) must contain a valid token
2. You have two options to login to Panaya via API. The first option is to provide a Panaya username and password. The second option is to use Panaya username, and a token (See more details about it below).
In both options, in the response's body you will get a new token. It can be found in the JSON of the response, under the key: "token".
3. For the next API, you should provide this token in the Request Header "X-Auth-Token." In response to this call, you will get a header "X-Auth-Token," which is a new token you should use in your next API call.
4. It continues as in #3. For each request you should provide the previous token you got. In the response, you will get a new token to be used in your next call.
5. Token expiration:
 - a. Currently, a token expires after 1 hour of inactivity, but note that this can be changed by Panaya in the future to conform to security requirements, so you should not rely on that.
 - b. For every API call, you should always check the result code. In case the HTTP Response code is 401, and in the response header, you have error_code=4016, it means that the token has expired, and you need to call the login API again.


Here is an example how a call with an expired token looks like:

Response Status: 401 Unauthorized

Response Body:

```
{  
  "status": "401",  
  "error": "Unauthorized",  
  "cause": "The authentication token has expired, please re-login.",  
  "panaya_code": "4016"  
}
```

Response Header:

KEY	VALUE
Server ⓘ	Apache-Coyote/1.1
Cache-Control ⓘ	no-cache, no-store, max-age=0, must-revalidate
Pragma ⓘ	no-cache
Expires ⓘ	0
Strict-Transport-Security ⓘ	max-age=31536000 ; includeSubDomains
X-XSS-Protection ⓘ	1; mode=block
X-Frame-Options ⓘ	DENY
X-Content-Type-Options ⓘ	nosniff
<u>error_code</u> ⓘ	4016 
Content-Type ⓘ	application/json
Content-Length ⓘ	133
Date ⓘ	Mon, 12 Apr 2021 13:29:49 GMT

LOGIN - API

There are two options to login to Panaya via API:

Option 1: Login using username and password.

Option 2: Login using username and access token.

Each of the options is described in detail below.

Why do we offer two options?

Some of Panaya customers are using SSO authentication, so for these customers, option 2 is the only way to login to Panaya via API.

Some of Panaya customers are using Panaya authorization, and for them, each of the options can be used to login to Panaya via API. Generally, these customers are more familiar with option 1.

Which option should you use?

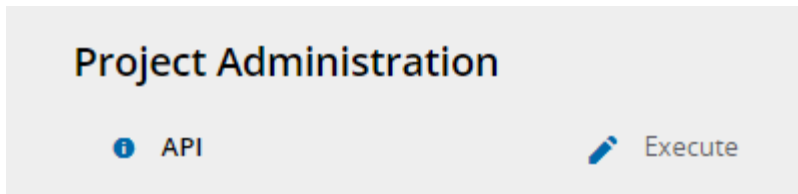
If you are writing a generic connector that will be used by several Panaya customers, we recommend that you should support both option 1 and option 2.

(Theoretically, you can support option 2 only, since all Panaya customers can use it).

If you are writing a connector for your own company, then if you are using SSO authentication, you must use option 2. Otherwise, you can use option 1 or option 2.

Notes about the Panaya user that should be used

- You should create a dedicated user for the API calls.
- This user should have permission for API Execution:



(You can use the "API User" role).

- This user should not be assigned to more than one company.

Login API Option 1 - Using Username and Password

Description:

You call the login API using that user's email and the user's password.

In the response, you will get a new token. You should use this token in your next API call (see above "Working with Tokens")

Request:

```
curl -X POST \  
https://my.panaya.com/api/login \  
-H 'Cache-Control: no-cache' \  
-H 'Content-Type: application/x-www-form-urlencoded' \  
-H 'Postman-Token: 8d146052-029a-499f-9195-466e14c33c9a' \  
-d 'username=user%40company.com&password=myPassWord2'
```

Response:

HTTP200

```
{"token":"5MFdkcdKpKfxnltDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvGHCw"}
```

Comments:

- You should write your "login retry" mechanism in a way that will not flood Panaya. Example:
 - Try to log in.
 - If failed, sleep for 30 seconds and try again.
 - If failed, sleep for 1 minute and try again.
 - If failed, sleep for 2 hours and try again.
 - If it still failed, you can quit the controller and send an internal alert to indicate that something went wrong with the connector.

Login API Option 2 – Using Username and an Access Token

Description:

You should create an access token for that user. See instructions here:

<https://success.panaya.com/Setup-Configuration/APIs/1743631181/How-to-create-your-token.htm>

You call the login API using that user's email and the user's access token.

In the response, you will get a new token. You should use this token in your next API call.

Note: The user's access token should be used with the Login API only. The next API call should use the new token you got in the response, as described above ("Working with Tokens").

You can read more about the access token API here:

<https://success.panaya.com/Setup-Configuration/APIs/993166071/How-to-authenticate-to-Panaya-via-API-with-an-Access-Token.htm>

Request:

```
curl -X POST \

https://my.panaya.com/api/accesstoken \

-H 'Cache-Control: no-cache' \

-H 'Content-Type: application/x-www-form-urlencoded' \
```

-H 'Postman-Token: 8d146052-029a-499f-9195-466e14c33c9a' \

-d 'username=user%40company.com&token=yourtoken'

Response:

HTTP200

```
{"token": "5MFdkcdKpKfxnltDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvgGHCw"}
```

Comments:

- See the comments in "Login API option 1" above.

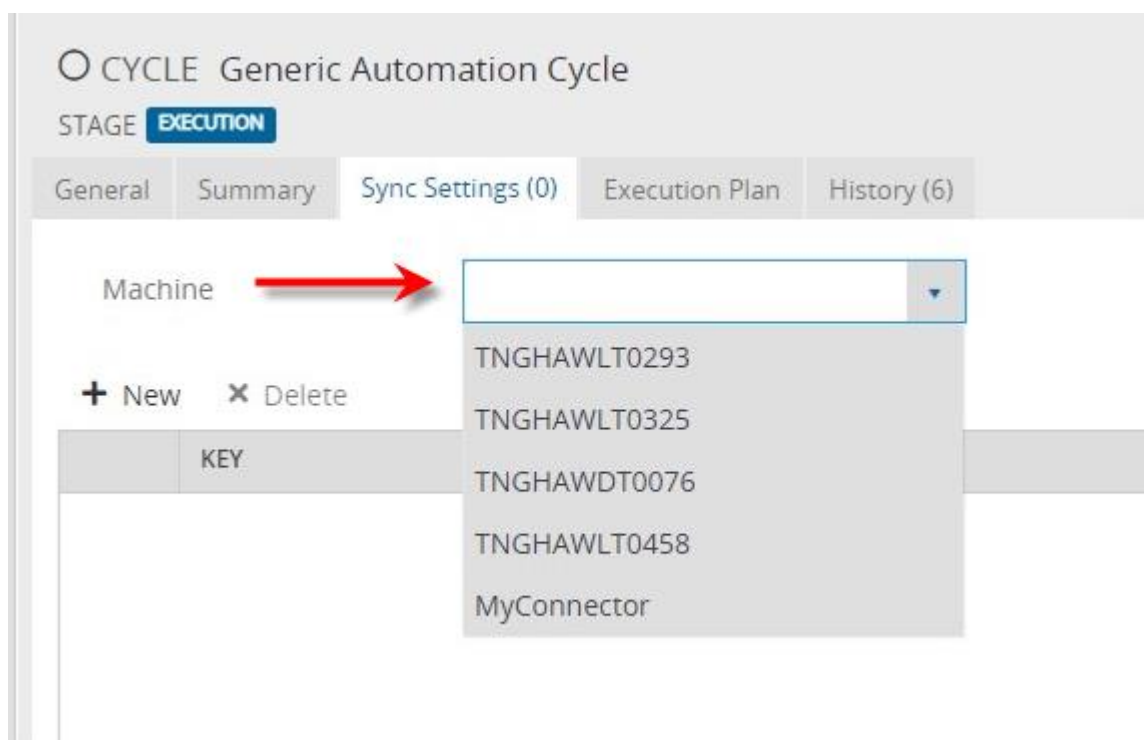
CONNECTOR REGISTRATION - API

Description:

Use this API to register your connector in Panaya, meaning – giving it a name.

This should be done only once, so once you registered your connector, you should persist the fact that you did so and avoid registration it again.

The connector name is shown in Panaya under the "Sync Settings" tab of the automation cycle:



There is an infrastructure to support multiple connectors, but this should be used in special cases only. In most cases, you should have only one connector, and you will see the name you registered under the "Machine" drop-down list.

Request:

url -X POST \

https://my.panaya.com/api/v1/systems/342b0bb1_1539de169f6__791e/automation/client/register \

-H 'Cache-Control: no-cache' \

-H 'Content-Type: application/json' \

-H 'Postman-Token: 353148f7-f3fe-435f-9f6d-cfc9779811db' \

-H 'X-Auth-Token:

5MFdkcdKpKfxnltDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvg
GHCw' \

-d '{

"computerId":"MyConnector",

"data":{"version":"UFT 1.0"}

}'

Response:

HTTP 200

Comments:

- As explained in the "General Guidelines for Using the API" section above, you should replace the connector ID in the example above (shown in red) with your specific connector ID.
- The computerID is the name of the connector. This is the name that will appear in the "Sync Setting" tab > "Machine."
- We recommend adding the automation type (e.g., the name of the automation tool) in the version field.
- The version must not start with the letters "WS."

GET TASKS - API

Description:

Use this API to check if there is a pending task for you in Panaya

There can be two types of tasks. See the high-level description above.

Request:

```
curl -X GET \
```

```
'https://my.panaya.com/api/v1/systems/342b0bb1_1539de169f6__791e/automation/tasks/?computerId= MyConnector ' \
```

```
-H 'Cache-Control: no-cache' \
```

```
-H 'Content-Type: application/x-www-form-urlencoded' \
```

```
-H 'Postman-Token: dbe64b43-fa15-46e5-9ee3-cc9c0e62fa52' \
```

```
-H 'X-Auth-Token:
```

```
5MFdkcdKpKfxnlDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvg  
GHCw'
```

Response:

Task of type 1 – request to sync tests:

```
{  
  "taskId": 100,  
  "taskItems": [  
    {  
      "$type": "GenericScanItem",  
      "id": 106,  
      "genericFields": {  
        "Agent": "Agent1",  
        "FolderPath": "/folder_1/tests",  
        "RetryFlag": "false"  
      }  
    }  
  ]  
}
```

```

    },
    "cycleName": "Automation Cycle 1",
    "cycleId": "58420"
  }
]
}

```

Tasks of type 2 – request to execute tests:

```

{
  "taskId": 110,
  "taskItems": [
    {
      "$type": "GenericExecutionItem",
      "id": 60,
      "executionOrder": 5,
      "testPath": "/Root7/folder1/107 New Test 50",
      "genericFields": {
        "key1": "val1",
        "key2": "val2",
        "key3": "val3",
        "TCodes": "ME21N"
      },
      "playlistExecutionName": "Playlist 30-Nov-2020 14:44:36",
      "plannedRun": "US Data"
    },
    {
      "$type": "GenericExecutionItem",
      "id": 59,
      "executionOrder": 4,
      "testPath": "/Root7/folder1/107 New Test 49",
      "genericFields": {
        "key1": "val1",
        "key2": "val2",
        "key3": "val3",
        "TCodes": "VA01, VA02"
      },
      "playlistExecutionName": "Playlist 30-Nov-2020 14:44:36",
    }
  ]
}

```

```
        "plannedRun": "EMEA Data"  
    }  
],  
    "sequence": false  
}
```

Comments:

- "TCodes" – in case the transaction codes are recognized in the specific system (depends on ETL data), the test will be marked as covering these transactions.
- "plannedRun" – this is the description of the test planned run. You can use it as a "data set" name in case you want to run the same test with several different data sets.

CREATE AUTOMATION TESTS IN PANAYA - API

Description:

Use this API to create new tests in Panaya (as a result of getting a task of type 1 - request to sync tests).

Request:

curl -X POST \

https://my.panaya.com/api/v1/systems/342b0bb1_1539de169f6__791e/automation/tests \

-H 'Cache-Control: no-cache' \

-H 'Content-Type: application/json' \

-H 'Postman-Token: e444557a-2afe-4e9c-91cc-ccf9160123ed' \

-H 'X-Auth-Token:

5MFdkcdKpKfxnltDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvg
GHCw' \

-d '{

```
"data":{
  "tests":[
    {
      "$type":"Panaya.Ac.Generic.GenericTestsScanning",
      "testPath":"/Root/folder1/New Test 1",
      "description":"Test Description 1",
      "genericFields":{
        "key1":"val1",
        "key2":"val2",
        "key3":"val3",
        "TCodes":" VA01,VA02"
      }
    },{
      "$type":"Panaya.Ac.Generic.GenericTestsScanning",
      "testPath":"/Root/folder1/New Test 2",
      "description":"Test Description 2",
```

```

    "genericFields":{
      "key1":"val1",
      "key2":"val2",
      "key3":"val3",
      "TCodes":" VA01,VA02"
    }
  ],
  "scanItemId":106,
  "stateResult":2,
  "logDetails":"I am syncing test to panaya"
},
"computerId":"MyConnector"
}'

```

Response:

HTTP 200

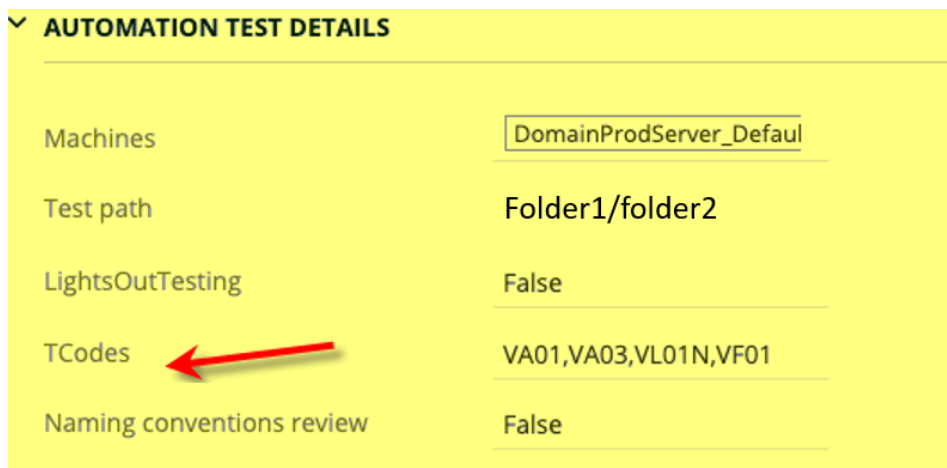
Comments:

- `scanItemId` is task item id from get task response.
- `$type` should be `"Panaya.Ac.Generic.GenericTestsScanning"` for each test.
- The 'Description' field is limited to 7,000 characters.
- This API is limited to 100 tests per API call. If you have more than 100 tests to create, you should break them into chunks of 100 tests. Each API call should create 100 tests. Please add a waiting time of 1 minute between each chunk.
- See explanation about the "TCodes" in the next paragraph.

WORKING WITH TCODES

When creating automation tests in Panaya, you can report the Transactions (TCodes) covered by each test.

1. The project you are working with must be a project that is associated with a system that you uploaded your ERP data into via the Code Box.
2. You send the TCodes as part of API to create a new test, in the “GenericFields” section. Use the “TCodes” keyword. See an example in the previous paragraph.
3. The TCodes will be shown in the Test’s General tab, under the “Automation Test Details” section:



AUTOMATION TEST DETAILS	
Machines	DomainProdServer_Default
Test path	Folder1/folder2
LightsOutTesting	False
TCodes	VA01,VA03,VL01N,VF01
Naming conventions review	False

4. When you send the TCodes as part of the test creation API, Panaya will parse this value of the “TCodes” field. The Delimiter is a comma, and spaces are being ignored.
5. The TCodes that are recognized by Panaya will be “connected” to the test. TCodes which are not recognized will be ignored. (Panaya “Recognized” TCodes are based on the latest ETL uploaded to your System). The TCodes are case insensitive.
6. No indication/error message will be provided on non-recognized TCodes (not to the user and not to the API caller).
7. The user can see the connected TCodes in the Test’s General tab, as “Covering Entry Points”:

TEST 35078 | Widget #23 - UI in all browsers

Not Run | Action

Ready to test

Progress:

General

Steps (2)

Data

Linked features (1)

Defects (0)

Automation

Attachments (0)

Comments (0)

History (10)

DESCRIPTION

Description

Test objective

Prerequisites

DETAILS

Priority

High

Business Process

Functional Area

Usage

Unknown

Module

Cross Module

Covering Entry Points

2

System

WORK PLAN

Due date

Work Effort

0.2

Time To Complete

0.2

ESSENTIAL

Path

20.12 > TDX > Lia FEATURE 13238

In Scope

Yes

Reviewed

No

Blocked

No one to handle step

Unassigned steps

2

CREATION

Created By

Liat Kimhi

Created On

02-Sep-2020

OWNERSHIP

IT Owner

... select value ...

Business Owner

... select value ...

Once the user clicks on the number of entry points, the entry points (TCodes) will be shown:

Panaya

● ● ● ● ●

23

Covering Entry Points

Object name: Add

Remove

ENTRY POINT	DESCRIPTION	TYPE	POPULATED BY
ME21N	Create Purchase Order	Report	Manually
VA01	Create Sales Order	Transaction	Manually

Close

8. The user can manually add new TCodes or remove existing TCodes using the above form. Any change to this form will not affect the "TCodes" fields in the "Automation Test Details" section.
9. The TCodes field will be parsed and processed (creating the covering entry points) only when the test is created via the API. If the user will change the TCodes field, it will have no effect on the Covering Entry Points.
10. There is no API to update a test, so there is no API to update the TCodes after the test was created.
11. The module and the usage of the test won't be affected by the entry points created.
12. Please note that there are some limitations around the "covering entry points" field. It is not available as a filter in the test list. You can add it as a column, but you can't sort nor group the list by this field.

REPORT TASK STATUS - API

Description:

Use this API to report the status of the processing of a task you got from Panaya

Request:

```
curl -X PUT \
```

```
https://my.panaya.com/api/v1/systems/342b0bb1_1539de169f6__791e/automation/tasks/11 \
```

```
-H 'Cache-Control: no-cache' \
```

```
-H 'Content-Type: application/json' \
```

```
-H 'Postman-Token: caa0c198-3d2c-4e38-9c95-8740d0379a96' \
```

```
-H 'X-Auth-Token:
```

```
5MFdkcdKpKfxnltDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvg  
GHCw' \
```

```
-d '{
```

```
  "computerId": "MyConnector",
```

```
  "data": {
```

```
    "stateResult": "1"
```

```
  }
```

```
}'
```

Response:

HTTP 200

Comments:

- The URL should include the task ID you want to report status to.
- Available values for the 'stateResult' field:
 - 1 – In Progress
 - 2 – Completed
 - 3 – Error

REPORT THE EXECUTION RESULT OF A TEST - API

Description:

Use this API to report the results of a test execution (as a result of getting a task of type 2 – execute tests).

Request:

curl -X PUT \

https://my.panaya.com/api/v1/systems/342b0bb1_1539de169f6__791e/automation/tests/execution \

-H 'Cache-Control: no-cache' \

-H 'Content-Type: application/json' \

-H 'Postman-Token: 4daa5d0f-30a0-4d2d-8e7d-1061db627683' \

-H 'X-Auth-Token:

5MFdkcdKpKfxnltDj8Z2+uO4TamC/0OL2AJV/avKATrABm5moBw5mLyn0uet2p4Vv6rZ2ZtSHc2p1eXcvGHCw' \

-d '{

"data": {

"zippedResults":

"UESDBBQAAAAIAOFbykwYMOcdPwEAABcEAAAZAAAAAbG10dGxldGV4dGZpbGVmb3J6aXAxLnR4dNWTTWwCQBCG74L/YQgeJeChIN4KtiU3aZuTigzZ1SzdD8lOKCL+984mG016KL14cHPI7Dtfuw87Z4DxCAASX6DNSJpMJltknkyjSkjyXfpaE8uzTtbusJSESntWOcYAK77zCiRMFud2Q9ITR61jH17nmxkCJnQ6Si6zQosnTJ+LN N9T+hnsPvhMVtID+mJjkZI+uIIYvEvuMoU/sujxYptwjhVSyZ02vEonOC/8jRSq8pObIFu1V1LAa20LU56ihlCX K242wXjaWSzK2gZrG+axbC5ua607z6UxtpdIpnDmWJOSgshonP921Zf3bk+zZDzi4PZr0EWM0HK8goT1le OtfQcR7kuxBxBuQbEHEQYUp49/57c8C/ZOahT/unNrbLtB7l0nzGP+cDxh1sn98bwWbSEOniD8foNhMpt X+ANQSwMEFAAAAAAgA5FvKTBgw5x0/AQAAFWQAABkAAAABsaXR0bGV0ZXh0ZmlsZWZvcnppcDIudHh01 ZNna8JAEIbvgv9hCB4l4KEg3gq2JTdpm5OKDNnVLN0PyU4olv73ziYbTXoovXhwc8jsO1+7DztngPEIABJfo M1Imkwki2SeTKNKSPjd+loTy7NO1u6wlIRKe1Y5xgArvwMKJEwW53ZD0hNHrWMfXuebGQlmdDpKLrNCi ydMn4s031P6Gdl++ExW2UP6YkmRkj64ghi8S+4yhT+y4nFim3COFVLJnTa8Sic4L/yNFKryk5uUW7VXUsBr bQtSsqKGUJcjbBeNpZLMraDOsb5rFsLm5rrTvPpTG2l0imcOZYk6wayGic/3bVI/duT7NkPOLg9mvQRYz

```

QcryChPWV4619BxHuS7EH5BsQcRBhSnj3/ntzwL9k5qFP+6c2tsu0HsjSfMY/5wPGHWyf3xvBZtIQ6eIPx
+g2Eym1f4A1BLAQIUABQAAAAIAOFbykwYMOcdPwEAABcEAAAZAAAAAAAAAAAAEIAAAAAAAAAABsaXR0b
GV0ZXh0ZmlsZWZvcnppcDEudHh0UESBAhQAFAAAAAgA5FvKTBgw5x0/AQAAFWQAABkAAAAAAAAAAQ
AgAAAAAdgEAAGxpDHRsZXRleHRmaWxlZm9yemlwMi50eHRQSwUGAAAAAIAAgCOAAAA7AIAAAAA",
"executionItemId": 5,

"stateResult": 2,

"result":2,

"logDetails": "some logs stuff string",

"duration": "0.504",

"executionStartTime": "06-JUN-2018 11:10:11"

},

"computerId": "MyConnector"

}

,

```

Response:

HTTP 2000

```

{

  "executionItemId": 5

}

```

Comments:

- You can send zipped or HTML results as evidence.
- We have two status fields for the execution of a test: **result** and **stateResult**.
- The **"StateResult"** is the technical status of the execution of the test. It is not related to the actual test logical result (pass/fail).
 - Here are the possible values:
 - 0 - Not Run

- 1- In Progress
 - 2 – Completed
 - 3 – Error
- 2 (Completed) means – the connector was able to execute the test, and the test execution was completed. The test result can be passed or failed, but from the technical perspective, the execution was "completed."
 - 3 (Error) - it does not mean that the test was run and completed with a "Failed" status. It means that the connector was not able to execute the test (e.g., a test was synced to Panaya, and after that, it was deleted on automation tool's side. When you run it from Panaya., the connector will not find the test and should report status 3 (Error).
 - 1 (In progress) means that the automation tool is currently running this test.
 - 0 (Not run) – The automation tool has not started to execute the test so far.
- The "**result**" field is the logical status of the test that was executed.
 - Here are the possible values:
 - 0 - Not Run
 - 1- In Progress
 - 2- Failed
 - 3 - Passed
 - It can be "Passed" or "Failed" - the test was executed completely, but the logical test result is "Failed."
 - It can be "not run" (which generally comes with StateResults=Not run)
 - It can be in progress (which generally comes with StateResult = in progress)
- If all tests in the playlist are in the same status, the playlist also gets the status. Otherwise, the status is in progress.

A SUMMARY OF THE CONNECTOR LOGIC AND THE APIS

1. Login to Panaya – api/login (POST)
2. One time only: Register the connector –
/api/v1/systems/{{MySystemUid}}/automation/client/register (POST)
3. Every 1-5 minutes, check for new tasks using the API:
/api/v1/systems/{{MySystemUid}}/automation/tasks/?computerId=UFT Server 1 (GET)
 - No new task – sleep for 1-5 minutes.
 - If task(s) exist:
 - i. If **scan task** - update the specific cycle with the relevant fetched tests from the automation tool. Only the delta of newly created tests should be sent.
/api/v1/systems/{{MySystemUid}}/automation/tests (POST)
 - ii. If **execution task** – execute the required test on the automation server and report results /api/v1/systems/{{MySystemUid}}/automation/tests/execution (PUT)
 - iii. Update task status to completed once the task completed.
/api/v1/systems/{{MySystemUid}}/automation/tasks/100 (POST) ("stateResult":"2")