TAP REST-API

A plugin providing an API that enables remotely spawning multiple TAP sessions. Each of the sessions are able to load testplans, run testplans and return traces and results. This project is build using ASP.NET Core and is therefore cross-platform.

About the REST-API Plugin

The TAP REST-API Plugin enables remote control of TAP and running multiple TAP sessions. This provides an opportunity to develop a GUI that can remote control multiple servers/instruments testing several DUTs.

The REST-API has two modes, each providing support for a specific use case:

1. Session
2. Proxy

The mode is chosen by specifying appropriate command line arguments, for more information see the *getting started* guide.

Session

The **session** mode is best described as a TAP instance with no GUI. This mode uses HTTP requests to enable remote control of TAP functionalities such as loading a test plan, changing external parameters and test step property values or running a test plan. Log messages, platform interactions and result listener information is also available through WebSockets.

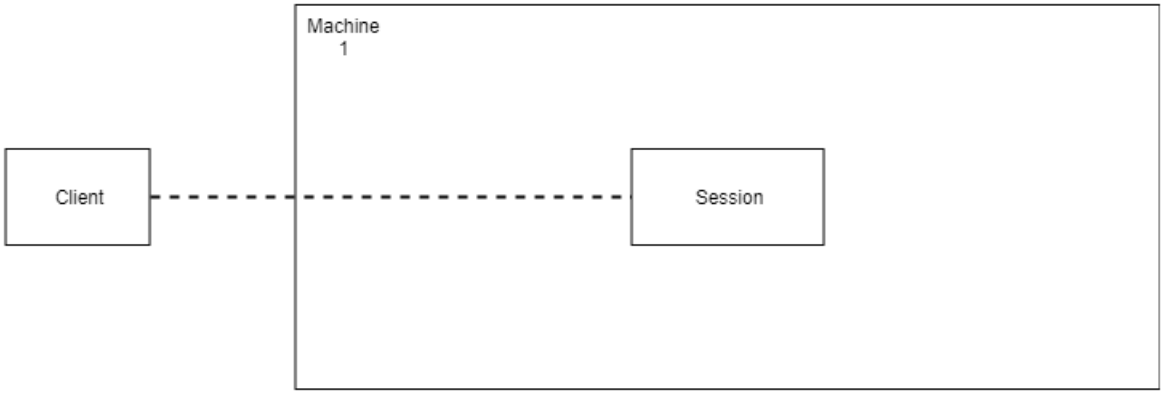

Figure: Session

The use case for spawning the REST-API in **session** mode describes the situation when a single TAP instance needs to be remotely controlled.

Proxy

The **proxy** mode can manage multiple sessions on a machine. It can spawn and close sessions and redirect requests and WebSockets to appropriate sessions.

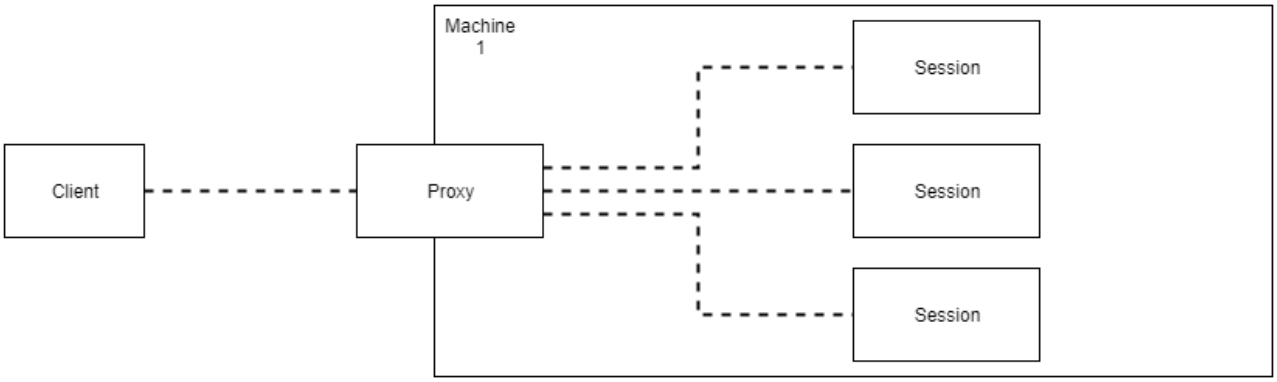

Figure: Proxy

The use case for spawning the REST-API in **proxy** mode describes the situation when multiple sessions need to be managed on a machine. This is the default mode.

Getting started

Dependencies

- TAP 9.0 or newer installed
- .NET Core 2.1 SDK installed (Linux)
- .NET Framework 4.6.2 installed (Windows Only)
- Visual Studio 2017.7 or newer installed **(Only for building source code)**

License requirements

The following licenses is required:

- KS8106A *(REST-API Plugin only supports network licenses on Linux)*

To connect to a license server and retrieve a network license

1. Add an environment variable with the name "LM_LICENSE_FILE" and the value "@{ServerAddress}".

To setup a license server

1. You will need to setup the Keysight Floating License Server to have clients checkout licenses from it. Link to documentation .
2. Add the purchased license file to the server.

Steps

1. Make sure you have the dependencies needed - TAP and .NET Core (specified above)
2. Download and install the REST API plugin
3. Run the following command in the TAP folder: `tap restapi`

Command Line Arguments

Command Line Argument	Command	Description
`Session` (-s)	**`-s`**	Spawns REST-API in Session mode.
`Port` (-p)	**`-p 9000`**	Specifies the port of the REST-API.
`Address` (-a)	**`-p http://192.168.0.22`**	Specifies the address of the REST-API (Default: all available).
`Quiet` (-q)	**`-q`**	Only show error and warning messages.
`Verbose` (-v)	**`-v`**	Verbose log messages. Includes debug messages.
`Mvc-logs` (-m)	**`-m`**	Log messages from AspNetCore MVC to SessionLogs.
`Password` (-w)	**`-w MySecretPassword`**	Sets Proxy session password. Used to remotely close Proxy by posting to `DELETE Proxy`.
`Log-Destination` (-l)	**`-l C:\Temp`** OR **`-l C:\Temp\MySpecificLogName.txt`**	Set destination of log files from Proxy and Sessions. Can be both a directory or a full path including filename
`Port-Range` (-r)	**`-r 15000-17000`**	Sets the port range used by spawned sessions. Format is <min>-<max>. Default is 7000-9000.
`Clean` (-r)	**`--clean`**	Cleans temporary folder used by the REST-API Proxy on startup.
`Session-Directory`	**`--session-directory`**	Specifies where to put temporary session installations

Plugin contents

The REST-API plugin contains the following additional examples and documentation:

- Packages (folder)
  - REST-API Plugin (folder)
    - JavascriptExample (folder)
      - JavascriptExamples.html (A simple HTML file containing html consuming the REST-API, open this to see the javascript client example)
        - RestAndSocket.js (Javascript used by the JavascriptExamples.html)
        - stylesheet.css (Simple CSS stylesheet used by the JavascriptExamples.html)
    - Postman (folder)
      - PostmanTestCollection.json (REST-API Test collection. Can be opened in Postman)
      - PostmanSettings.json (Postman environment - ip and port of request can be changed in this file)
    - REST-API-Documentation.pdf (This documentation)

Test Collection

The PostmanTestCollection.json can be imported in Postman (www.getpostman.com) to inspect the available requests.

To run the json files as a test, use Newman. It can be downloaded via node package manager using the **npm install newman –global** command.

Example: To run the REST-API tests using newman, use: **`newman run PostmanTestCollection.json -e PostmanSettings.json`**

Download links:

- Newman
- Postman

REST endpoints

Proxy endpoints

URI	Description
{url}/session	Spawns, register and closes TAP REST-API Sessions
{url}/proxy	Proxy commands
{url}/images	Create images and retrieve images
{url}/settings	Retrieves and sets Proxy settings. Sessions spawned using the Proxy inherit these settings

Session endpoints (*All require a Session ID in header when used through proxy mode*)

URI	Description
{url}/plan	Load testplan from XML and retrieve loaded testplan in XML
{url}/plan/step	Gets and sets teststep values (Property grid editing)
{url}/plan/steps	Gets testplan structure in JSON and add, remove, reorder teststeps in testplan
{url}/plan/steps/allowaschild	Gets teststep insertion info
{url}/plan/externalparameters	Retrieves and sets the external parameters of the plan
{url}/plan/run	Runs the loaded testplan
{url}/plan/run/abort	Aborts the running testplan
{url}/plan/waitforcompletion	Waits for testplan execution to complete and returns test run verdict
{url}/steptypes	Retrieves all installed TestStep types
{url}/session	Retrieves and sets session information
{url}/settings	Retrieves and sets session settings
{url}/validation	Retrieves loaded test plan validation errors
{url}/breakpoint	Gets and sets breakpoint at steps
{url}/jumptostep	Jump to specific step in plan in break mode

Websocket endpoints

URI	Type	Description
{url}/logs	Session	Subscribe to log messages (TraceListener)
{url}/results	Session	Subscribe to results (ResultListener)
{url}/interactions	Session	Subscribe to 'Platform Interactions' and answer them
{url}/events	Session	Subscribe to events (TestPlanChanged, TestStepPropertyChanged)
{url}/proxyevents	Proxy	Subscribe to session start and stop events

The JavascriptExamples.html contains plain javascript consuming the REST-API and Websockets.

SESSION Endpoint

[POST] SESSION

Request Type: POST

URL: {url}/session

Headers:

Key	Value
watchdog	<timeout in ms>

Watchdog is a feature that enables sessions to be kept alive by the user. This is primarily to avoid "zombie-sessions" (Sessions that are no longer monitored by the proxy). The "watchdog" functionality is optional. Default is no timeout, i.e sessions live forever or until manually shut down.

To disable the watchdog for a session started with a watchdog value, include the watchdog header with a value of "0" in any request to the desired session.

To extend the timeout, include the watchdog header with a value of the new timeout in ms, in any request to the desired session.

Returns:

- Status Code 201 (Created) : Created a TAP Session
- Status Code 400 (Bad Request) : If REST-API is not installed as a TapPackage and session can't be spawned (Happens when building from source).

A GUID (Session ID) in reponse body (plain text format)

[POST] SESSION with image id

Request Type: POST

URL: {url}/session/{imageId} (See {url}/images)

Returns:

- Status code 400 (BadRequest) : Unknown image id
- Status Code 200 (Created) : Created a TAP Session

A GUID (Session ID) in reponse body (plain text format)

[GET] SESSION

Request Type: GET

URL: {url}/session

Returns:

- Status Code 200 (Ok) : Returns session IDs

An ID array of currently running sessions in response body (JSON format)

[DELETE] SESSION

Request Type: DELETE

URL: {url}/session/{SessionId}

Returns:

- Status Code 200 (Ok) : Shutdown the TAP Session

Closes the TAP REST-API session

[POST] SESSION/REGISTER

Request Type: POST

URL: {url}/session/register/{port}

Started sessions has the opportunity to register to a proxy and thereby making them available to have routed requests through the proxy.

This proxy endpoint will automatically identify the IP of the request, however the listening port must be specified as in the {port} part of the URL.

Returns:

- Status Code 200 (Ok) : Proxy successfully registered session
- Status Code 400 (Bad Request) : Port not specified
- Status Code 400 (Bad Request) : IP is local and port is the same as Proxy.

IMAGES Endpoint

[POST] IMAGES

Request Type: POST

URL: {url}/images

An image can be specified by posting a JSON image configuration or a TestPlan or a ComponentSettings file.

Example of JSON image configuration request body format (use Content-Type application/json):

```
{
    "Packages": [
        {
            "Architecture": "x86",
            "Name": "TAP Base",
            "Version": "8.4"
        },
        {
            "Name": "REST-API Bundle",
            "Version": "1.0.72"
        },
        {
            "Name": "UXM_Driver",
            "Version": "1.4.227"
        },
        {
            "Name": "Demonstration",
            "Version": "8.0.28"
        }
    ],
    "Repositories": [
        "http://keysighttaprepository.azurewebsites.net/",
        "http://plugins.tap.aalborg.keysight.com",
        "http://plugins.tap.aalborg.keysight.com:8086/"
    ]
}
```

Example of TestPlan or ComponentSetting request body format (use Content-Type application/xml):

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.6.165.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="a6722707-40e9-4e16-98b4-cc88bdf0abeb"
Version="8.6.165.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
  <Package.Dependencies>
    <Package Name="TAP Base" Version="8.6.165-beta+3772be61" />
  </Package.Dependencies>
</TestPlan>
```

Example of multipart request with multiple testplans, settings and/or json images (use Content-Type 'multipart/mixed; boundary=<BOUNDARY>')

Format is: {–}—

Where curly braces indicates 1 or more times.

```
--<BOUNDARY>
{
    "Packages": [
```

```
        {
            "Architecture": "x86",
            "Name": "TAP Base",
            "Version": "8.4"
        },
        {
            "Name": "REST-API Bundle",
            "Version": "1.0.72"
        },
        {
            "Name": "UXM_Driver",
            "Version": "1.4.227"
        },
        {
            "Name": "Demonstration",
            "Version": "8.0.28"
        }
    ],
    "Repositories": [
        "http://keysighttaprepository.azurewebsites.net/",
        "http://plugins.tap.aalborg.keysight.com",
        "http://plugins.tap.aalborg.keysight.com:8086/"
    ]
}
--<BOUNDARY>
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.6.165.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="a6722707-40e9-4e16-98b4-cc88bdf0abeb"
Version="8.6.165.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
  <Package.Dependencies>
    <Package Name="TAP Base" Version="8.6.165-beta+3772be61" />
  </Package.Dependencies>
</TestPlan>
--<BOUNDARY>--
```

Returns:

- Status Code 200 (Ok) : Created an image

An id (Image ID) in reponse body (plain text format)

[GET] IMAGES

Request Type: GET

URL: {url}/images

Returns: - Status Code 200 (Ok) : Returns image list

An ID array of images in response body (JSON format). Example:

```
[
    {
        "Repositories": [
            "http://keysighttaprepository.azurewebsites.net/",
            "http://plugins.tap.aalborg.keysight.com",
            "http://plugins.tap.aalborg.keysight.com:8086/"
        ],
        "Packages": [
            {
                "Name": "TAP Base",
                "Version": "8.4",
                "Architecture": "x86",
                "OS": null
            },
            {
```

```
                "Name": "REST-API Bundle",
                "Version": "1.0.72",
                "Architecture": "Unknown",
                "OS": null
            },
            {
                "Name": "UXM_Driver",
                "Version": "1.4.227",
                "Architecture": "Unknown",
                "OS": null
            },
            {
                "Name": "Demonstration",
                "Version": "8.0.28",
                "Architecture": "Unknown",
                "OS": null
            }
        ],
        "Id": "57b19820231c75e44e8f8f600db72c0f"
    }
]
```


[GET] IMAGES/Proxy

Request Type: GET

URL: {url}/images/proxy

Returns:

- Status Code 200 (Ok) : Returns proxy image ID in plain text
- Status Code 204 (NoContent) : Empty body, indicates that the Proxy installation is currently not available as an image.

[POST] IMAGES/Proxy/{imageId}

Request Type: POST

URL: {url}/images/proxy/{imageId}

Returns:

- Status Code 200 (Ok) : Shutdown the TAP Proxy Server and starts installation modification with the body content: "Proxy shutdown"
- Status Code 400 (Bad Request) : All sessions must be closed before modifying proxy installation
- Status Code 400 (Bad Request) : No image ID specified
- Status Code 400 (Bad Request) : Unknown image id

Closes the proxy application and starts proxy installation modification of packages. Proxy will restart at same address and port when completed. Images and their IDs can be retrieved from GET Images endpoint .

Proxy Endpoint

[DELETE] Proxy

Request Type: DELETE

URL: {url}/proxy

Body: Must contain the Proxy password that was set as the "password" command line argument when spawning the application.

Returns:

- Status Code 200 (Ok) : Shutdown the TAP Proxy Server with the body content: "Proxy shutdown"
- Status Code 403 (Forbidden) : Wrong password specified in body content of request

Closes ALL TAP REST-API sessions and closes the proxy application. You will not be able to spawn new REST-API sessions and the proxy application will need to be manually started again.

IMAGES Endpoint

[POST] IMAGES

Request Type: POST

URL: {url}/images

An image can be specified by posting a JSON image configuration or a TestPlan or a ComponentSettings file.

Example of JSON image configuration request body format (use Content-Type application/json):

```
{
    "Packages": [
        {
            "Architecture": "x86",
            "Name": "TAP Base",
            "Version": "8.4"
        },
        {
            "Name": "REST-API Bundle",
            "Version": "1.0.72"
        },
        {
            "Name": "UXM_Driver",
            "Version": "1.4.227"
        },
        {
            "Name": "Demonstration",
            "Version": "8.0.28"
        }
    ],
    "Repositories": [
        "http://keysighttaprepository.azurewebsites.net/",
        "http://plugins.tap.aalborg.keysight.com",
        "http://plugins.tap.aalborg.keysight.com:8086/"
    ]
}
```

Example of TestPlan or ComponentSetting request body format (use Content-Type application/xml):

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.6.165.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="a6722707-40e9-4e16-98b4-cc88bdf0abeb"
Version="8.6.165.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
  <Package.Dependencies>
    <Package Name="TAP Base" Version="8.6.165-beta+3772be61" />
  </Package.Dependencies>
</TestPlan>
```

Example of multipart request with multiple testplans, settings and/or json images (use Content-Type 'multipart/mixed; boundary=<BOUNDARY>')

Format is: {–}—

Where curly braces indicates 1 or more times.

```
--<BOUNDARY>
{
    "Packages": [
```

```
        {
            "Architecture": "x86",
            "Name": "TAP Base",
            "Version": "8.4"
        },
        {
            "Name": "REST-API Bundle",
            "Version": "1.0.72"
        },
        {
            "Name": "UXM_Driver",
            "Version": "1.4.227"
        },
        {
            "Name": "Demonstration",
            "Version": "8.0.28"
        }
    ],
    "Repositories": [
        "http://keysighttaprepository.azurewebsites.net/",
        "http://plugins.tap.aalborg.keysight.com",
        "http://plugins.tap.aalborg.keysight.com:8086/"
    ]
}
--<BOUNDARY>
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.6.165.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="a6722707-40e9-4e16-98b4-cc88bdf0abeb"
Version="8.6.165.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
  <Package.Dependencies>
    <Package Name="TAP Base" Version="8.6.165-beta+3772be61" />
  </Package.Dependencies>
</TestPlan>
--<BOUNDARY>--
```

Returns:

- Status Code 200 (Ok) : Created an image

An id (Image ID) in reponse body (plain text format)

[GET] IMAGES

Request Type: GET

URL: {url}/images

Returns: - Status Code 200 (Ok) : Returns image list

An ID array of images in response body (JSON format). Example:

```
[
    {
        "Repositories": [
            "http://keysighttaprepository.azurewebsites.net/",
            "http://plugins.tap.aalborg.keysight.com",
            "http://plugins.tap.aalborg.keysight.com:8086/"
        ],
        "Packages": [
            {
                "Name": "TAP Base",
                "Version": "8.4",
                "Architecture": "x86",
                "OS": null
            },
            {
```

```
                "Name": "REST-API Bundle",
                "Version": "1.0.72",
                "Architecture": "Unknown",
                "OS": null
            },
            {
                "Name": "UXM_Driver",
                "Version": "1.4.227",
                "Architecture": "Unknown",
                "OS": null
            },
            {
                "Name": "Demonstration",
                "Version": "8.0.28",
                "Architecture": "Unknown",
                "OS": null
            }
        ],
        "Id": "57b19820231c75e44e8f8f600db72c0f"
    }
]
```


[GET] IMAGES/Proxy

Request Type: GET

URL: {url}/images/proxy

Returns:

- Status Code 200 (Ok) : Returns proxy image ID in plain text
- Status Code 204 (NoContent) : Empty body, indicates that the Proxy installation is currently not available as an image.

[POST] IMAGES/Proxy/{imageId}

Request Type: POST

URL: {url}/images/proxy/{imageId}

Returns:

- Status Code 200 (Ok) : Shutdown the TAP Proxy Server and starts installation modification with the body content: "Proxy shutdown"
- Status Code 400 (Bad Request) : All sessions must be closed before modifying proxy installation
- Status Code 400 (Bad Request) : No image ID specified
- Status Code 400 (Bad Request) : Unknown image id

Closes the proxy application and starts proxy installation modification of packages. Proxy will restart at same address and port when completed. Images and their IDs can be retrieved from GET Images endpoint .

SETTINGS Endpoint

[GET] SETTINGS

Request Type: GET

URL: {url}/settings

Headers:

Key	Value
Id	{SessionId}

Returns: - Status Code 200 (Ok) : Returns a settings object where root elements is the settings group


```
{
  "Bench": {
    "Connections": {
      "Value": {
        "0": {
          "Display": {
            "Name": "RF Connection",
            "Description": null,
            "Collapsed": false,
            "Order": -10000,
            "Groups": []
          },
          "Properties": {
            "CableLoss": {
              "Display": {
                "Name": "Cable Loss",
                "Description": null,
                "Collapsed": false,
                "Order": -10000,
                "Groups": []
              },
              "ReadOnly": false,
              "Value": {},
              "TypeName": "List<CableLossPoint>",
              "Errors": []
            },
            "Name": {
              "Display": {
                "Name": "Name",
                "Description": null,
                "Collapsed": false,
                "Order": -100000,
                "Groups": []
              },
              "Value": "",
              "ReadOnly": false,
              "TypeName": "String",
              "Errors": []
            },
            "Port1": null,
            "Port2": null,
            "Via": {
              "Display": {
                "Name": "Via",
                "Description": null,
                "Collapsed": false,
                "Order": -99998,
                "Groups": []
              },
              "ReadOnly": false,
              "Value": {},
              "TypeName": "List<ViaPoint>",
              "Errors": []
            }
```

```
          }
        }
      },
      "AvailableTypes": [
        {
          "TypeName": "Keysight.Tap.RfConnection",
          "Display": {
            "Name": "RF Connection",
            "Description": null,
            "Collapsed": false,
            "Order": -10000,
            "Groups": []
          }
        },
        {
          "TypeName": "Keysight.Tap.DirectionalRfConnection",
          "Display": {
            "Name": "Directional RF Connection",
            "Description": null,
            "Collapsed": false,
            "Order": -10000,
            "Groups": []
          }
        }
      ],
      "Display": {
        "Name": "Connections",
        "Description": "Connection Settings",
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      }
    },
    "Profile": "Dennis",
    "Profiles": [
      "Default",
      "Dennis"
    ],
    "DUTs": {
      "Value": {},
      "AvailableTypes": [
        {
          "TypeName": "Keysight.Tap.Plugins.Demo.ResultsAndTiming.TimeDut",
          "Display": {
            "Name": "Timing DUT",
            "Description": "Dummy DUT with configurable open, action and close times.\nUseful for demonstrating the
Timing Analyzer.",
            "Collapsed": false,
            "Order": -10000,
            "Groups": [
              "Demo",
              "Results And Timing"
            ]
          }
        }
      ],
      "Display": {
        "Name": "DUTs",
        "Description": "DUT Settings.",
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      }
    },
    "Instruments": {
      "Value": {},
      "AvailableTypes": [
        {
          "TypeName": "Keysight.Tap.Plugins.BasicSteps.RawSCPIInstrument",
          "Display": {
            "Name": "Generic SCPI Instrument",
            "Description": "Allows you to configure a VISA based connection to a SCPI instrument.",
            "Collapsed": false,
            "Order": -10000,
            "Groups": []
```

```
          }
        },
        {
          "TypeName": "Keysight.Tap.Plugins.Demo.Battery.PowerAnalyzer",
          "Display": {
            "Name": "Power Analyzer",
            "Description": "Simulated power analyzer instrument used for charge/discharge demo steps.",
            "Collapsed": false,
            "Order": -10000,
            "Groups": [
              "Demo",
              "Battery Test"
            ]
          }
        },
        {
          "TypeName": "Keysight.Tap.Plugins.Demo.Battery.TemperatureChamber",
          "Display": {
            "Name": "Temperature Chamber",
            "Description": "Simulated temperature chamber instrument used for SetTemperature demo step.",
            "Collapsed": false,
            "Order": -10000,
            "Groups": [
              "Demo",
              "Battery Test"
            ]
          }
        },
        {
          "TypeName": "Keysight.Tap.Plugins.Demo.ResultsAndTiming.TimeInst",
          "Display": {
            "Name": "Timing Instrument",
            "Description": "Dummy instrument with configurable open, action and close times.\nUseful for demonstrating
the Timing Analyzer.",
            "Collapsed": false,
            "Order": -10000,
            "Groups": [
              "Demo",
              "Results And Timing"
            ]
          }
        }
      ],
      "Display": {
        "Name": "Instruments",
        "Description": "Instrument Settings",
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      }
    }
  },
  "": {
    "Engine": {
      "Display": {
        "Name": "Engine",
        "Description": "Engine Settings",
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      },
      "Properties": {
        "SessionLogPath": {
          "Text": "SessionLogs/SessionLog <Date>.txt",
          "Expanded": "SessionLogs/SessionLog 2018-03-23 13-01-05.txt",
          "Errors": []
        },
        "PromptForMetaData": {
          "Display": {
            "Name": "Allow Metadata Dialog",
            "Description": "When the test plan starts, show a dialog to prompt the user for data specified by the plugin
to associate with the results.",
            "Collapsed": false,
            "Order": 2,
            "Groups": [
              "General"
```

```
        ]
      },
      "Value": false,
      "ReadOnly": false,
      "TypeName": "Boolean",
      "Errors": []
    },
    "AbortTestPlan": {
      "Value": [
        "1"
      ],
      "Values": {
        "0": "Step Fail",
        "1": "Step Error"
      },
      "Display": {
        "Name": "Abort Run If",
        "Description": "Allows aborting the test plan run if a step fails or causes an error.",
        "Collapsed": false,
        "Order": 1,
        "Groups": [
          "General"
        ]
      },
      "ReadOnly": false,
      "TypeName": "Enum",
      "Errors": []
    },
    "OperatorName": {
      "Display": {
        "Name": "Name",
        "Description": "Name of the operator. This name will be saved along with the results.",
        "Collapsed": false,
        "Order": 20,
        "Groups": [
          "Operator"
        ]
      },
      "Value": "deasmuss",
      "ReadOnly": false,
      "TypeName": "String",
      "Errors": []
    },
    "StationName": {
      "Display": {
        "Name": "Station",
        "Description": "Name of the test station. This name will be saved along with the results.",
        "Collapsed": false,
        "Order": 21,
        "Groups": [
          "Operator"
        ]
      },
      "Value": "5TL33H2",
      "ReadOnly": false,
      "TypeName": "String",
      "Errors": []
    }
  }
},
"Results": {
  "Value": {},
  "AvailableTypes": [
    {
      "TypeName": "Keysight.Tap.LogResultListener",
      "Display": {
        "Name": "Text Log",
        "Description": "Save the log from a test plan run as a text file.",
        "Collapsed": false,
        "Order": -10000,
        "Groups": [
          "Text"
        ]
      }
    },
```

```
        {
          "TypeName": "Keysight.Tap.Plugins.BasicSteps.NotifyingResultListener",
          "Display": {
            "Name": "Notifier",
            "Description": "Notifies you when an action has completed.",
            "Collapsed": false,
            "Order": -10000,
            "Groups": [
              "Action"
            ]
          }
        }
      ],
      "Display": {
        "Name": "Results",
        "Description": "Results Settings",
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      }
    }
  }
}
```

[POST] SETTINGS

Request Type: POST

URL: {url}/settings

Headers:

Key	Value
Id	{SessionId}
Content-Type	application/json

Body The relevant content of the GET Settings body response.

Returns:

- Status Code 200 (Ok): Returns the updated settings object.

To change bench profile:

Change the "Profile" value. For example to change from "Dennis" profile to "Default" profile:

```
{
  "Bench": {
    "Profile": "Default",
    "Profiles": [
      "Default",
      "Dennis"
    ],
  }
}
```

To add a profile:

Append a profile name to the "Profiles" array. Example:

```
{
  "Bench": {
    "Profile": "Default",
    "Profiles": [
      "Default",
      "Dennis",
      "New Profile"
```

```
    ],
  }
}
```

To remove a profile:

Remove the entry from the "Profiles" array. Example:

```
{
  "Bench": {
    "Profile": "Default",
    "Profiles": [
      "Default"
    ],
  }
}
```

To remove a resource from the list:

Set the list entry to `null`. For example, to remove the Connection:

```
{
  "Bench": {
    "Connections": {
      "Value": {
        "0": null
      }
    }
  }
}
```

To add a resource to the list:

Add an entry and use the typename from the "AvailableTypes" array to create the desired object. To add a generic SCPI instrument to "Instruments":

```
{
  "Bench": {
    "Instruments": {
      "Value": {
          "0" : {
              "TypeName": "Keysight.Tap.Plugins.BasicSteps.RawSCPIInstrument"
          }
      }
    }
  }
}
```

PLAN Endpoint

[POST] PLAN

Request Type: POST

URL: {url}/plan

Headers:

Key	Value
Id	{SessionId}
Content-Type	text/plain
TestPlanName	<TestPlanName>

"TestPlanName" header is optional. If not specified, the testplan name will have the name of the session ID.

Body: XML TestPlan. Example:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.0.158.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="42847eba-1a7c-432b-a864-49e62f69223a"
Version="8.0.158.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
</TestPlan>
```

Returns: Status code 200, empty body.

[GET] PLAN

Request Type: GET

URL: {url}/plan

Headers:

Key	Value
Id	{SessionId}

Returns: A TestPlan in XML format. Example:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.0.158.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="42847eba-1a7c-432b-a864-49e62f69223a"
Version="8.0.158.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
</TestPlan>
```

STEP Endpoint

[GET] STEP

Request Type: GET

URL: {url}/plan/step/{stepId} (e.g. http://localhost:20116/plan/step/42847eba-1a7c-432b-a864-49e62f69223a)

Headers:

Key	Value
Id	{SessionId}

Returns: Detailed information about a TestStep in JSON format.

In case of this C# TestStep:

```
[Display("Set Verdict", Group:"Basic Steps", Description: "Reports a verdict for the step.")]
public class VerdictStep : TestStep
{
    [Display("Resulting Verdict", Order:-1, Description: "The verdict the step should produce when run.")]
    public VerdictOutput VerdictOutput { get; set; }

    [Display("Request Abort", Description: "Setting this property to true will cause the test plan to abort as quickly
as possible.")]
    public bool RequestAbort { get; set; }

    public VerdictStep()
    {
        VerdictOutput = BasicSteps.VerdictOutput.Pass;
        RequestAbort = false;
    }

    public override void Run()
    {
        UpgradeVerdict(VerdictOutput.ToTestStepVerdict());
        if (RequestAbort)
            throw new TestPlan.AbortException("Verdict Step requested to abort the test plan execution.");
    }
}
```

Returned JSON:


```
{
  "TypeName": "Keysight.Tap.Plugins.BasicSteps.VerdictStep",
  "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
  "Enabled": true,
  "Name": "Set Verdict",
  "Display": {
    "Name": "Set Verdict",
    "Description": "Reports a verdict for the step.",
    "Collapsed": false,
    "Order": -10000.0,
    "Groups": [
      "Basic Steps"
    ]
  },
  "Properties": {
    "VerdictOutput": {
      "Value": "1",
      "Values": {
        "0": "Not Set",
        "1": "Pass",
        "2": "Inconclusive",
        "3": "Fail",
        "4": "Error"
      },
```

```
      "Display": {
        "Name": "Resulting Verdict",
        "Description": "The verdict the step should produce when run.",
        "Collapsed": false,
        "Order": -1.0,
        "Groups": []
      },
      "ReadOnly": false,
      "TypeName": "Enum",
      "ExternalParameter": {
        "Enabled": false,
        "Name": ""
      },
      "Errors": []
    },
    "RequestAbort": {
      "Display": {
        "Name": "Request Abort",
        "Description": "Setting this property to true will cause the test plan to abort as quickly as possible.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": []
      },
      "Value": false,
      "ReadOnly": false,
      "TypeName": "Boolean",
      "ExternalParameter": {
        "Enabled": false,
        "Name": ""
      },
      "Errors": []
    }
  }
}
```


[POST] STEP

Request Type: POST

URL: {url}/plan/step/{stepId} (e.g. http://localhost:20116/plan/step/42847eba-1a7c-432b-a864-49e62f69223a)

Headers:

Key	Value
Id	{SessionId}
Content-Type	application/json

Body: The body should contain the JSON structure returned by the [GET] Step request with desired changed values.

If we want to change the value of the RequestAbort property to true and the value of the VerdictOutput property to fail, we include the following in the request body:

```
{
    "Properties": {
        "VerdictOutput": {
            "Value": "Fail"
        },
        "RequestAbort": {
            "Value": true
        }
    }
}
```

Note, the detailed information of the step is omitted - this is optional.

- To add a property as an external parameter to the testplan, change the ExternalParameter.Enabled to true (name is optional) and use the result as body in this request.

Returns: Detailed information about a TestStep in JSON format with updated values. In this case, the following is returned:


```
{
  "Properties": {
    "RequestAbort": {
      "Value": true,
      "ExternalParameter": {
        "Enabled": false,
        "Name": ""
      },
      "TypeName": "Boolean",
      "ReadOnly": false,
      "Display": {
        "Name": "Request Abort",
        "Description": "Setting this property to true will cause the test plan to abort as quickly as possible.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": []
      },
      "Errors": []
    },
    "VerdictOutput": {
      "Value": "1",
      "Values": {
        "0": "Not Set",
        "1": "Pass",
        "2": "Inconclusive",
        "3": "Fail",
        "4": "Error"
      },
      "Display": {
        "Name": "Resulting Verdict",
        "Description": "The verdict the step should produce when run.",
        "Collapsed": false,
        "Order": -1.0,
        "Groups": []
      },
      "ReadOnly": false,
      "TypeName": "Enum",
      "ExternalParameter": {
        "Enabled": false,
        "Name": ""
      },
      "Errors": [
        "Input string was not in a correct format."
      ]
    }
  },
  "Display": {
    "Name": "Set Verdict",
    "Description": "Reports a verdict for the step.",
    "Collapsed": false,
    "Order": -10000.0,
    "Groups": [
      "Basic Steps"
    ]
  },
  "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
  "Enabled": true,
  "Name": "Set Verdict",
  "TypeName": "Keysight.Tap.Plugins.BasicSteps.VerdictStep"
}
```


Working with Enumerable types

Enumerable types count lists, enumerables and arrays, but are all displayed similarly through the REST-API. Therefore, the following example using only a list type the teststep `StepWithList`, illustrates the principle for all types.

```
public class StepWithList : TestStep
{
    public List<string> StringList { get; set; } = new List<string>() { "Hello", "World", "List" };
```

```
    public override void Run()
    {
        throw new NotImplementedException();
    }
}
```

The REST-API returns the following JSON on a GET /plan/step request:

```
{
  "TypeName": "RestOutputDocGen.StepWithList",
  "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
  "Enabled": true,
  "Name": "StepWithList",
  "Properties": {
    "StringList": {
      "ReadOnly": false,
      "Value": {
        "0": "Hello",
        "1": "World",
        "2": "List"
      },
      "TypeName": "List<String>",
      "ExternalParameter": {
        "Enabled": false,
        "Name": ""
      },
      "Errors": []
    }
  }
}
```

By using the POST /plan/step you can add, remove and edit entries in the enumerables.

To add an item to the list, specify the new entry with a higher index, as such:

```
{
  "Properties": {
    "StringList": {
      "ReadOnly": false,
      "Value": {
        "0": "Hello",
        "1": "World",
        "2": "List",
        "3": "Test"
      },
      "TypeName": "List<String>",
      "Errors": []
    }
  },
  "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
  "Enabled": true,
  "Name": "StepWithList",
  "TypeName": "Tap.Plugins.RestApi.Tests.SetStepTests+StepWithList"
}
```

To remove an item, set the value of an entry to null:

```
{
  "Properties": {
    "StringList": {
      "ReadOnly": false,
      "Value": {
        "0": "Hello",
        "1": null,
        "2": "List",
      },
      "TypeName": "List<String>",
      "Errors": []
    }
  },
  "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
  "Enabled": true,
  "Name": "StepWithList",
  "TypeName": "Tap.Plugins.RestApi.Tests.SetStepTests+StepWithList"
```

```
}
```

To edit the value of an item, simply alter the value:

```
{
  "Properties": {
    "StringList": {
      "ReadOnly": false,
      "Value": {
        "0": "Hello",
        "1": "New",
        "2": "List",
      },
      "TypeName": "List<String>",
      "Errors": []
    }
  },
  "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
  "Enabled": true,
  "Name": "StepWithList",
  "TypeName": "Tap.Plugins.RestApi.Tests.SetStepTests+StepWithList"
}
```

All existing entries not to be removed or edited can be left out from the POST request.

STEPS endpoint

[GET] STEPS

Request Type: GET

URL: {url}/plan/steps

Headers:

Key	Value
Id	{SessionId}

QueryStrings:

- properties

Returns: A TestPlan in JSON format.

If this XML TestPlan is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.1.297.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="0592b959-07eb-46d9-9cb4-ac20175d6ca0"
Version="8.1.297.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.LogStep" Id="f0a9dea6-588a-48d3-82aa-9b296f32e05d"
Version="8.1.297.0">
      <ChildTestSteps />
      <Enabled>True</Enabled>
      <LogMessage></LogMessage>
      <Name>Log Output</Name>
      <Severity>Info</Severity>
    </TestStep>
  </Steps>
</TestPlan>
```

The GET request: **{url}/plan/steps** returns the following:


```
{
  "IsRunning": false,
  "Aborted": false,
  "ChildTestSteps": [
    {
      "Verdict": "NotSet",
      "Enabled": true,
      "Name": "Delay",
      "Id": "0592b959-07eb-46d9-9cb4-ac20175d6ca0",
      "Display": {
        "Name": "Delay",
        "Description": "Delays for a specified amount of time.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Basic Steps"
        ]
      },
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
      "ChildTestSteps": []
    },
    {
      "Verdict": "NotSet",
```

```
      "Enabled": true,
      "Name": "Log Output",
      "Id": "f0a9dea6-588a-48d3-82aa-9b296f32e05d",
      "Display": {
        "Name": "Log Output",
        "Description": "Outputs a specified message to the log with a specified severity.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Basic Steps"
        ]
      },
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.LogStep",
      "ChildTestSteps": []
    }
  ]
}
```


The endpoint supports specifying properties to be added. This enables the functionality in TAP GUI to display properties and their values in the TestPlan panel. Properties to be included must be specified with their name specified in the DisplayAttribute. For example, the DelayStep has the property "DelaySecs" which has the name "Time Delay" in its DisplayAttribute It is this name that we specify.

The GET request: **{url}/plan/steps?properties=Time Delay&properties=Log Severity** returns the following:


```
{
  "IsRunning": false,
  "Aborted": false,
  "ChildTestSteps": [
    {
      "Time Delay": {
        "Unit": {
          "PreScaling": 1.0,
          "StringFormat": "",
          "Unit": "s",
          "UseEngineeringPrefix": false,
          "UseRanges": true
        },
        "Value": "0.1 s",
        "RawValue": "0.1",
        "Display": {
          "Name": "Time Delay",
          "Description": "The time to delay in the test step.",
          "Collapsed": false,
          "Order": -10000.0,
          "Groups": []
        },
        "ReadOnly": false,
        "TypeName": "Double",
        "ExternalParameter": {
          "Enabled": false,
          "Name": ""
        }
      },
      "Verdict": "NotSet",
      "Enabled": true,
      "Name": "Delay",
      "Id": "0592b959-07eb-46d9-9cb4-ac20175d6ca0",
      "Display": {
        "Name": "Delay",
        "Description": "Delays for a specified amount of time.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Basic Steps"
        ]
      },
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
      "ChildTestSteps": []
    },
    {
      "Verdict": "NotSet",
      "Enabled": true,
      "Name": "Log Output",
```

```
      "Id": "f0a9dea6-588a-48d3-82aa-9b296f32e05d",
      "Display": {
        "Name": "Log Output",
        "Description": "Outputs a specified message to the log with a specified severity.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Basic Steps"
        ]
      },
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.LogStep",
      "ChildTestSteps": []
    }
  ]
}
```


[POST] STEPS

Request Type: POST

URL: {url}/plan/steps

Headers:

Key	Value
Id	{SessionId}
Content-Type	application/json

Body: The body should contain the JSON structure similar to the returned by the [GET] Steps request with desired changes.

This simple testplan is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.0.158.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="42847eba-1a7c-432b-a864-49e62f69223a"
Version="8.0.158.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
</TestPlan>
```

Using the [GET] Steps request with the session ID fetches the following JSON:


```
{
  "IsRunning": false,
  "Aborted": false,
  "ChildTestSteps": [
    {
      "Verdict": "NotSet",
      "Enabled": true,
      "Name": "Delay",
      "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
      "Display": {
        "Name": "Delay",
        "Description": "Delays for a specified amount of time.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Basic Steps"
        ]
      },
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
      "ChildTestSteps": []
    }
```

```
  ]
}
```

The request is able to add teststeps, remove teststeps, reorder teststeps and change name and enabled values.

Add a teststep

To add a teststep, specify a fully qualified typename of an installed teststep on the server. E.g. to add a Log Output teststep after the delay step, the body could look like:

```
{
    "Locked": false,
    "IsRunning": false,
    "Aborted": false,
    "ChildTestSteps": [
        {
            "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
            "Enabled": true,
            "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
            "Name": "Delay",
            "Display": {
                "Name": "Delay",
                "Description": "Delays for a specified amount of time.",
                "Collapsed": false,
                "Order": -10000,
                "Groups": [
                    "Basic Steps"
                ]
            },
            "ChildTestSteps": []
        },
        {
            "TypeName": "Keysight.Tap.Plugins.BasicSteps.LogStep"
        }
    ]
}
```

Remove a teststep

To remove a teststep, remove the step (or just the ID) E.g. to remove the delay step, leaving the testplan empty:

```
{
    "IsRunning": false,
    "Aborted": false,
    "ChildTestSteps": [
    ]
}
```

Reorder teststeps

To reorder teststeps, change the order in json. Extending the example from Add a teststep , the new teststep could be placed before the delay step. This works similarly with already added teststeps.

```
{
    "IsRunning": false,
    "Aborted": false,
    "ChildTestSteps": [
        {
            "TypeName": "Keysight.Tap.Plugins.BasicSteps.LogStep"
        },
        {
            "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
            "Enabled": true,
            "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
            "Name": "Delay",
            "Display": {
                "Name": "Delay",
                "Description": "Delays for a specified amount of time.",
                "Collapsed": false,
```

```
                "Order": -10000,
                "Groups": [
                    "Basic Steps"
                ]
            },
            "ChildTestSteps": []
        }
    ]
}
```

Change values

To set the name or the enabled status of teststeps, change them and optionally leave out other details of the step. ID is required.

```
{
    "ChildTestSteps": [
        {
            "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
            "Enabled": false,
            "Name": "New Name",
            "ChildTestSteps": []
        }
    ]
}
```

Multiple actions

Multiple actions can be conducted in one request. E.g change values of the delay step and add a repeat step, with a log output step as child:

```
{
    "ChildTestSteps": [
        {
            "TypeName": "Keysight.Tap.Plugins.BasicSteps.RepeatStep",
            "ChildTestSteps": [
                {
                    "TypeName": "Keysight.Tap.Plugins.BasicSteps.LogStep"
                }
            ]
        }
        {
            "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
            "Enabled": false,
            "Name": "New Name",
            "ChildTestSteps": []
        }
    ]
}
```

Returns: Updated testplan steps information. The returned JSON from the multiple actions request is:

```
{
  "IsRunning": false,
  "Aborted": false,
  "ChildTestSteps": [
    {
      "Verdict": "NotSet",
      "Enabled": true,
      "Name": "Repeat",
      "Id": "62197ead-a320-49e2-b33d-811284956b86",
      "Display": {
        "Name": "Repeat",
        "Description": "Repeats its child steps a fixed number of times or until the verdict of a child step changes to
a specified state.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Flow Control"
        ]
      },
```

```
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.RepeatStep",
      "ChildTestSteps": [
        {
          "Verdict": "NotSet",
          "Enabled": true,
          "Name": "Log Output",
          "Id": "c73a7207-781f-49b4-8f52-dccd17dc6c81",
          "Display": {
            "Name": "Log Output",
            "Description": "Outputs a specified message to the log with a specified severity.",
            "Collapsed": false,
            "Order": -10000.0,
            "Groups": [
              "Basic Steps"
            ]
          },
          "TypeName": "Keysight.Tap.Plugins.BasicSteps.LogStep",
          "ChildTestSteps": []
        }
      ]
    },
    {
      "Verdict": "NotSet",
      "Enabled": false,
      "Name": "New Name",
      "Id": "42847eba-1a7c-432b-a864-49e62f69223a",
      "Display": {
        "Name": "Delay",
        "Description": "Delays for a specified amount of time.",
        "Collapsed": false,
        "Order": -10000.0,
        "Groups": [
          "Basic Steps"
        ]
      },
      "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
      "ChildTestSteps": []
    }
  ]
}
```

STEPS/ALLOWASCHILD endpoint

[GET] STEPS/ALLOWASCHILD

Request Type: GET

URL: {url}/plan/steps/allowaschild/{stepType}

Headers:

Key	Value
Id	{SessionId}

Returns: TestStep type insertion options. I.e. if the teststep can be added as child to the TestPlan or as child of any of the currently existing TestSteps.

If this XML TestPlan is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.2.493.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="47c1a236-d348-449c-ba78-8be09f639015"
Version="8.2.493.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.RepeatStep" Id="f6c5e438-e0c6-4a1b-b882-5fcb5f9a2ca3"
Version="8.2.493.0">
      <Action>Fixed_Count</Action>
      <ChildTestSteps />
      <Count>3</Count>
      <Enabled>True</Enabled>
      <MaxCount>
        <IsEnabled>False</IsEnabled>
        <Value>3</Value>
      </MaxCount>
      <Name>Repeat</Name>
      <TargetStep />
      <TargetVerdict>Fail</TargetVerdict>
    </TestStep>
  </Steps>
</TestPlan>
```

The GET request: **{url}/plan/steps/allowaschild/Keysight.Tap.Plugins.BasicSteps.RepeatStep** returns the following:

```
{
  "PlanChild": true,
  "StepChild": [
    "f6c5e438-e0c6-4a1b-b882-5fcb5f9a2ca3"
  ]
}
```

[POST] STEPS/ALLOWASCHILD

Request Type: POST

URL: {url}/plan/steps/allowaschild

Headers:

Key	Value
Id	{SessionId}
Content-Type	application/json

Body: An array of TestStep IDs.

This simple testplan is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.2.493.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="47c1a236-d348-449c-ba78-8be09f639015"
Version="8.2.493.0">
      <ChildTestSteps />
      <DelaySecs>0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.RepeatStep" Id="f6c5e438-e0c6-4a1b-b882-5fcb5f9a2ca3"
Version="8.2.493.0">
      <Action>Fixed_Count</Action>
      <ChildTestSteps />
      <Count>3</Count>
      <Enabled>True</Enabled>
      <MaxCount>
        <IsEnabled>False</IsEnabled>
        <Value>3</Value>
      </MaxCount>
      <Name>Repeat</Name>
      <TargetStep />
      <TargetVerdict>Fail</TargetVerdict>
    </TestStep>
  </Steps>
</TestPlan>
```

Using the [POST] Steps/AllowAsChild request with an array of containing the DelayStep IDs fetches the following JSON:

```
{
  "PlanChild": true,
  "StepChild": [
    "f6c5e438-e0c6-4a1b-b882-5fcb5f9a2ca3"
  ]
}
```

EXTERNALPARAMETERS endpoint

[GET] EXTERNAL PARAMETERS

Request Type: GET

URL: {url}/plan/externalparameters

Headers:

Key	Value
Id	{SessionId}

Returns: Detailed information about the external parameters.

If this simple testplan is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.0.158.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DelayStep" Id="42847eba-1a7c-432b-a864-49e62f69223a"
Version="8.0.158.0">
      <ChildTestSteps />
      <DelaySecs external="ExtTimeDelay">0.1</DelaySecs>
      <Enabled>True</Enabled>
      <Name>Delay</Name>
    </TestStep>
  </Steps>
</TestPlan>
```

The following JSON is returned:

```
{
  "ExtTimeDelay": {
    "Unit": {
      "PreScaling": 1.0,
      "StringFormat": "",
      "Unit": "s",
      "UseEngineeringPrefix": false,
      "UseRanges": true
    },
    "Value": "0.1 s",
    "RawValue": "0.1",
    "Errors": [],
    "ReadOnly": false,
    "TypeName": "Double",
    "ExternalParameter": {
      "Name": "ExtTimeDelay",
      "Enabled": true
    },
    "StepProperties": [
      {
        "StepId": "42847eba-1a7c-432b-a864-49e62f69223a",
        "PropertyDisplayName": "Time Delay",
        "PropertyName": "DelaySecs",
        "StepName": "Delay"
      }
    ]
  }
}
```

[POST] EXTERNAL PARAMETERS

Request Type: POST

URL: {url}/plan/externalparameters

Headers:

Key	Value
Id	{SessionId}
Content-Type	application/json

Actions - To remove the property from the external parameter, delete the entry from the "StepProperties" array and use the result as body in a POST {url}/plan/externalparameters. - To completely remove the external parameter, switch the ExternalParameter.Enabled to false and use the result as body in a POST {url}/plan/externalparameters. - To rename the external parameter, change the ExternalParameter.Name property and use the result as body in a POST {url}/plan/externalparameters. - To add an external parameter, see GET/POST Step documentation.

Returns: Updated information about the external parameters. Change the value and use the JSON in the body of the request, e.g:

```
{
    "ExtTimeDelay": {
        "Value": "5"
    }
}
```

Returns the following JSON:

```
{
  "ExtTimeDelay": {
    "Value": "5 s",
    "StepProperties": [
      {
        "StepId": "42847eba-1a7c-432b-a864-49e62f69223a",
        "PropertyDisplayName": "Time Delay",
        "PropertyName": "DelaySecs",
        "StepName": "Delay"
      }
    ],
    "TypeName": "Double",
    "ReadOnly": false,
    "ExternalParameter": {
      "Name": "ExtTimeDelay",
      "Enabled": true
    },
    "Unit": {
      "PreScaling": 1.0,
      "StringFormat": "",
      "Unit": "s",
      "UseEngineeringPrefix": false,
      "UseRanges": true
    },
    "RawValue": "5",
    "Errors": []
  }
}
```

POST RUN

Endpoint to run loaded testplan.

Request Type: POST

URL: {url}/plan/run

Headers:

Key	Value
Id	{SessionId}

Returns: Status Code 200 if plan starts to run

GET RUN

Endpoint to retrieve run status.

Request Type: GET

URL: {url}/plan/run

Headers:

Key	Value
Id	{SessionId}

Returns: Status Code 200 and one of the following status messages in body (plain text):

- "Aborting" - Indicates the testplan has retrieved signal to abort, but has not yet aborted.
- "Running" - Indicates the testplan is running.
- "Ready" - Indicates the testplan ready to run.

POST ABORT

Endpoint to signal the running testplan to abort.

Request Type: POST

URL: {url}/plan/run/abort

Headers:

Key	Value
Id	{SessionId}

Returns: Status Code 200 - plan was signalled to abort

WAIT FOR COMPLETION endpoint

[GET] WAITFORCOMPLETION

Request Type: GET

URL: {url}/plan/waitforcompletion

Headers:

Key	Value
Id	{SessionId}

QueryStrings:

- timeout (miliseconds, integer)

Returns:

- Status Code 202 (Accepted) : Timeout reached before testplan finished
- Status Code 400 (Bad Request) : TestPlan has not started or testplan thread crashed.
- Status Code 200 (Ok) : Retrieved last TestPlan execution details as specified below.

JSON formatted **Verdict** and **FailedToStart** information.

If this simple testplan is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.1.313.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.VerdictStep" Id="c48364ab-1650-4e3b-b07d-3979316d9dfa"
Version="8.1.313.0">
      <ChildTestSteps />
      <Enabled>True</Enabled>
      <Name>Set Verdict</Name>
      <RequestAbort>False</RequestAbort>
      <VerdictOutput>Pass</VerdictOutput>
    </TestStep>
  </Steps>
</TestPlan>
```

After running the testplan , the following is returned using a GET request {url}/plan/waitforcompletion?timeout=5000:

```
{
  "Verdict": "NotSet",
  "FailedToStart": false
}
```

Omitting the timeout querystring ([GET] {url}/plan/waitforcompletion) assumes timeout of zero and the request will return immediately with either TestRun completed info (status code 200) or with no content (status code 204).

GET INSTALLED TESTSTEP TYPES

Request Type: GET

URL: {url}/steptypes/{stepId}

Headers:

Key	Value
Id	{SessionId}

Returns: All installed TestStep types in JSON format. All step types has two boolean values, CanAddChild and CanAddSibling, which indicated whether the steptype can be added respectivily as a child or sibling to the testplan or a step.

{url}/steptypes without specifying a {stepId} will return all installed step types with CanAddChild and CanAddSibling values in relation to the testplan

{url}/steptypes/{stepId} with specified {stepId} value will return all installed step types with CanAddChild and CanAddSibling values in relation to the teststep with id matching {stepId} in the loaded testplan.

The GET request: **{url}/steptypes** returns the following:


```
[
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
    "Display": {
      "Name": "Delay",
      "Description": "Delays for a specified amount of time.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Basic Steps"
      ]
    },
    "CanAddChild": false,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.DialogStep",
    "Display": {
      "Name": "Dialog",
      "Description": "Used to interact with the user.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Basic Steps"
      ]
    },
    "CanAddChild": false,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.IfStep",
    "Display": {
      "Name": "If Verdict",
      "Description": "Runs its child steps only when the verdict of another step has a specific value.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Flow Control"
      ]
    },
    "CanAddChild": false,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.SweepLoopRange",
```

```
    "Display": {
      "Name": "Sweep Loop (Range)",
      "Description": "Loops all of its child steps while sweeping a specified parameter/setting over a range.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Flow Control"
      ]
    },
    "CanAddChild": false,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.LockStep",
    "Display": {
      "Name": "Lock",
      "Description": "Locks the execution of child steps based on a specified mutex.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Flow Control"
      ]
    },
    "CanAddChild": false,
    "CanAddSibling": true
  }
]
```


The GET request: **{url}/steptypes/{stepId}**, where stepId is a RepeatStep, returns the following:


```
[
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.DelayStep",
    "Display": {
      "Name": "Delay",
      "Description": "Delays for a specified amount of time.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Basic Steps"
      ]
    },
    "CanAddChild": true,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.DialogStep",
    "Display": {
      "Name": "Dialog",
      "Description": "Used to interact with the user.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Basic Steps"
      ]
    },
    "CanAddChild": true,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.IfStep",
    "Display": {
      "Name": "If Verdict",
      "Description": "Runs its child steps only when the verdict of another step has a specific value.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Flow Control"
      ]
    },
    "CanAddChild": true,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.SweepLoopRange",
```

```
    "Display": {
      "Name": "Sweep Loop (Range)",
      "Description": "Loops all of its child steps while sweeping a specified parameter/setting over a range.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Flow Control"
      ]
    },
    "CanAddChild": true,
    "CanAddSibling": true
  },
  {
    "TypeName": "Keysight.Tap.Plugins.BasicSteps.LockStep",
    "Display": {
      "Name": "Lock",
      "Description": "Locks the execution of child steps based on a specified mutex.",
      "Collapsed": false,
      "Order": -10000.0,
      "Groups": [
        "Flow Control"
      ]
    },
    "CanAddChild": true,
    "CanAddSibling": true
  }
]
```

SESSION INFORMATION Endpoint

[GET] SESSION

Request Type: GET

URL: {url}/session

Headers:

Key	Value
Id	{SessionId}

Returns: - Status Code 200 (Ok) : Returns session information (Name, LastVerdict, TestPlanRunId)

If you use a database resultlistener, this endpoint exposes the `TestPlanRunId` which can be used to query the database for the specific test plan execution.

Example:


```
{
  "Properties": {
    "Name": {
      "Display": {
        "Name": "Name",
        "Description": null,
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      },
      "Value": "cac4cfaa-0294-4ba9-8b0a-1b09c326ba60_7001",
      "ReadOnly": false,
      "TypeName": "String"
    },
    "LastVerdict": {
      "Value": "0",
      "Values": {
        "0": "Not Set",
        "1": "Pass",
        "2": "Inconclusive",
        "3": "Fail",
        "4": "Aborted",
        "5": "Error"
      },
      "Display": {
        "Name": "LastVerdict",
        "Description": null,
        "Collapsed": false,
        "Order": -10000,
        "Groups": []
      },
      "ReadOnly": true,
      "TypeName": "Enum"
    },
    "TestPlanRunId": "24ab1568-f064-41dd-959f-ba757398f75c"
  },
  "TypeName": "Keysight.Tap.Plugins.RestApi.TapNode.TapSession"
}
```


[POST] SESSION

Request Type: POST

URL: {url}/session

Headers:

Key	Value
Id	{SessionId}
Content-Type	application/json

Body: GET request body with modified “Value” fields.

Returns:

- Status Code 200 (Ok) : Returns modified session information

Validation Errors endpoint

[GET] VALIDATION

Request Type: GET

URL: {url}/plan/validation

Headers:

Key	Value
Id	{SessionId}

Returns: All validation errors in loaded test plan

If the following testplan, containing several validation rule violations, is loaded:

```
<?xml version="1.0" encoding="utf-8"?>
<TestPlan type="Keysight.Tap.TestPlan" Locked="False" Version="8.4.529.0">
  <Steps>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.DialogStep" Id="e85030cd-5dd3-4e49-9416-9184738701d4"
Version="8.4.529.0">
      <Buttons>OkCancel</Buttons>
      <ChildTestSteps />
      <DefaultAnswer>False</DefaultAnswer>
      <Enabled>True</Enabled>
      <Message></Message>
      <Name>Dialog</Name>
      <NegativeAnswer>Not_Set</NegativeAnswer>
      <PositiveAnswer>Not_Set</PositiveAnswer>
      <Timeout>5</Timeout>
      <Title></Title>
      <UseTimeout>False</UseTimeout>
    </TestStep>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.SweepLoopRange" Id="9af8af62-0d63-4b32-9afc-62f1d89fe1ad"
Version="8.4.529.0">
      <ChildTestSteps />
      <Enabled>True</Enabled>
      <Name>Sweep Loop (Range)</Name>
      <SweepBehavior>Exponential</SweepBehavior>
      <SweepEnd>100</SweepEnd>
      <SweepPoints>0</SweepPoints>
      <SweepPropertyName></SweepPropertyName>
      <SweepStart>0</SweepStart>
      <SweepStop>100</SweepStop>
    </TestStep>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.RepeatStep" Id="94f67277-12d1-4ff7-a608-d6b6c0aae3e0"
Version="8.4.529.0">
      <Action>While</Action>
      <ChildTestSteps />
      <Count>3</Count>
      <Enabled>True</Enabled>
      <MaxCount>
        <IsEnabled>False</IsEnabled>
        <Value>3</Value>
      </MaxCount>
      <Name>Repeat</Name>
      <TargetStep />
      <TargetVerdict>Fail</TargetVerdict>
    </TestStep>
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.LockStep" Id="36c5a109-856c-42ce-beae-baf3baca5a21"
Version="8.4.529.0">
      <ChildTestSteps />
      <Enabled>True</Enabled>
      <LockName>\</LockName>
      <Name>Lock</Name>
      <SystemWide>True</SystemWide>
    </TestStep>
```

```
    <TestStep type="Keysight.Tap.Plugins.BasicSteps.SCPIRegexStep" Id="5e354cd2-6f75-4632-89fd-23917fe113b3"
Version="8.4.529.0">
      <Action>Command</Action>
      <AddToLog>True</AddToLog>
      <Behavior>GroupsAsDimensions</Behavior>
      <ChildTestSteps />
      <DimensionTitles></DimensionTitles>
      <Enabled>True</Enabled>
      <Instrument />
      <LogHeader></LogHeader>
      <Name>SCPI</Name>
      <Query>*IDN?</Query>
      <RegularExpressionPattern>
        <IsEnabled>False</IsEnabled>
        <Value>(.*)</Value>
      </RegularExpressionPattern>
      <ResultName>Regex result</ResultName>
      <ResultRegularExpressionPattern>
        <IsEnabled>False</IsEnabled>
        <Value>(.*)</Value>
      </ResultRegularExpressionPattern>
      <VerdictOnMatch>Pass</VerdictOnMatch>
      <VerdictOnNoMatch>Fail</VerdictOnNoMatch>
    </TestStep>
  </Steps>
</TestPlan>
```

The following JSON is returned:


```
[
  {
    "StepId": "e85030cd-5dd3-4e49-9416-9184738701d4",
    "Error": [
      "Title is empty",
      "Message is empty"
    ],
    "StepName": "Dialog"
  },
  {
    "StepId": "9af8af62-0d63-4b32-9afc-62f1d89fe1ad",
    "Error": [
      "Sweep start value must be non-zero.",
      "Sweep points must be bigger than 1.",
      "Sweep start and end value must have the same sign."
    ],
    "StepName": "Sweep Loop (Range)"
  },
  {
    "StepId": "94f67277-12d1-4ff7-a608-d6b6c0aae3e0",
    "Error": [
      "No step selected"
    ],
    "StepName": "Repeat"
  },
  {
    "StepId": "36c5a109-856c-42ce-beae-baf3baca5a21",
    "Error": [
      "Lock Name does not support '\\'"
    ],
    "StepName": "Lock"
  },
  {
    "StepId": "5e354cd2-6f75-4632-89fd-23917fe113b3",
    "Error": [
      "",
      "Resource is not set on property 'Instrument'"
    ],
    "StepName": "SCPI"
  }
]
```

Breakpoint endpoint

[GET] Breakpoint

Request Type: GET

URL: {url}/plan/breakpoint

Headers:

Key	Value
Id	{SessionId}

Returns:

- Status Code 200 (Ok) - All breakpoints in json array format

```
{
  "0": "42847eba-1a7c-432b-a864-49e62f69223a",
  "1": "49f2c872-fe24-4fda-b7df-156ba1b870c9"
}
```

[SET] Breakpoint

Request Type: GET

URL: {url}/plan/breakpoint/{stepId}

Headers:

Key	Value
Id	{SessionId}

Returns:

- Status Code 200 (Ok) - All breakpoints in json array format

If you specify {stepId} as `e6f38ba5-0dfc-4d31-bea5-7001e65c6d19` in the request, the step will be added to the list of test steps to break at and return the following JSON:

```
{
  "0": "42847eba-1a7c-432b-a864-49e62f69223a",
  "1": "49f2c872-fe24-4fda-b7df-156ba1b870c9",
  "2": "e6f38ba5-0dfc-4d31-bea5-7001e65c6d19"
}
```

If you specify an already existing ID to the request, that ID will be removed.

Jumptostep endpoint

[POST] JUMPTOSTEP

Request Type: POST

URL: {url}/plan/jumptostep/{stepId}

Headers:

Key	Value
Id	{SessionId}

Returns:

- Status Code 200 (Ok)

Websocket Logs

URL:

{url}/logs

QueryStrings:

Key	Value	Description	Example
Id	{SessionId}	Specify the session to connect	{url}/logs?id=d6c81036-e29c-4bee-83e9-4b7c41d63070

Server -> Client

Example of data sent from Server to Client:

```
{
    "EventType": 40,
    "Source": "TestPlan",
    "Timestamp": 636530783652657583,
    "DurationNS": 0,
    "Message": "Repeat \\ Delay PrePlanRun completed. [0 ms]"
}
```

EventType displays the log event type and includes: - 10 : Error - 20 : Warning - 30 : Information - 40 : Debug

Source is the log source identifier this event was logged from.

Timestamp is a date and time expressed in the number of 100-nanosecond intervals that have elapsed since January 1, 0001 at 00:00:00.000 in the Gregorian calendar. As an example to use this in Javascript as a Date object, the number can be subtracted by 621355968000000000 (100-nanosecond representation of the time span between January 1, 0001 and unix start time - 1 January, 1970) and divided by 10000 to convert from 100 nanosecond to ms. A Javascript Date object can be initialized using this calculated value in ms from 1 January, 1970.

```
var time = new Date((json.Timestamp - 621355968000000000) / 10000.0);
```

Message is the message for the event.

Client -> Server

This socket has no client to server communication implemented.

Websocket Results

URL:

{url}/results

QueryStrings:

Key	Value	Description	Example
Id	{SessionId}	Specify the session to connect	{url}/results?id=d6c81036-e29c-4bee-83e9-4b7c41d63070
Data	true	Specify the websocket connection to receive results (ResultPublished)	{url}/results?id=d6c81036-e29c-4bee-83e9-4b7c41d63070&data=true

Server -> Client

Example of data sent from Server to Client:

TestPlanRun


```
{
    "Status": "TestPlanRunStart",
    "Id": "1b5f6b76-ba04-4b26-86df-0e20de44fc8a",
    "Verdict": "NotSet",
    "Duration": "0 ms",
    "Parameters": [{
        "Group": "Test Plan",
        "Name": "Hash",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": "C3AA2A5DA13CED36"
    }, {
        "Group": "Version",
        "Name": "Keysight.Tap.Plugins.RestApi",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": "1.0.0.0 - a2e8b04b"
    }, {
        "Group": "Version",
        "Name": "Keysight.Tap.Engine",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": "8.1.318.0 - 97475c3f"
    }]
}
```


Id is the ID of this test run. Can be used to uniquely identify a TestStepRun or TestPlanRun

Verdict is the resulting verdict.

Duration is the length of time it took to run.

Parameters is a list of parameters associated with this run that can be used by ResultListener.

Status is the result type name.

TestStepRun


```
{
    "Status": "TestStepRunStart",
    "Id": "bfe4c91e-1fcf-40fc-b7bc-4d3a6e7879a6",
    "Verdict": "NotSet",
    "Duration": "0 ms",
    "Parameters": [{
        "Group": "",
```

```
        "Name": "Repeat",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": "Fixed_Count"
    }, {
        "Group": "",
        "Name": "Equals",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": "Fail"
    }, {
        "Group": "",
        "Name": "Count",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": 3
    }, {
        "Group": "",
        "Name": "Max Count",
        "MacroName": null,
        "ParentLevel": 0,
        "Value": "(disabled)3"
    }],
    "ParentId": "767955f7-34c1-4afa-a762-811f8d7a0f18",
    "TestStepId": "b9d96a09-48a4-4b7c-9527-e1d1fd513ad5",
    "TestStepName": "Repeat"
}
```


ParentId is the ID of the parent.

TestStepId is the ID of the teststep of the TestStepRun

TestStepName is the name of the teststep of the TestStepRun

ResultPublished

If querystring data was set to true when subscribing to /results websocket, the "ResultPublished" data will also be transferred.

The following JSON is received when executing a TestPlan with a SineResult TestStep, from the Demonstration plugin.


```
{
  "Status": "ResultPublished",
  "StepRunId": "969fabf1-d089-4744-8a43-1ff819ca69b5",
  "ResultTable": {
    "Name": "SineResults",
    "Rows": 50,
    "Columns": [
      {
        "Name": "PointIndex",
        "ObjectType": "Result Column",
        "Data": {
          "0": "0",
          "1": "1",
          "2": "2",
          "3": "3",
          "4": "4",
          "5": "5",
          "6": "6",
          "7": "7",
          "8": "8",
          "9": "9",
          "10": "10",
          "11": "11",
          "12": "12",
          "13": "13",
          "14": "14",
          "15": "15",
          "16": "16",
          "17": "17",
          "18": "18",
          "19": "19",
          "20": "20",
```

```
      "21": "21",
      "22": "22",
      "23": "23",
      "24": "24",
      "25": "25",
      "26": "26",
      "27": "27",
      "28": "28",
      "29": "29",
      "30": "30",
      "31": "31",
      "32": "32",
      "33": "33",
      "34": "34",
      "35": "35",
      "36": "36",
      "37": "37",
      "38": "38",
      "39": "39",
      "40": "40",
      "41": "41",
      "42": "42",
      "43": "43",
      "44": "44",
      "45": "45",
      "46": "46",
      "47": "47",
      "48": "48",
      "49": "49"
    },
    "TypeCode": "Int32"
  },
  {
    "Name": "Value",
    "ObjectType": "Result Column",
    "Data": {
      "0": "20",
      "1": "21.253332335643",
      "2": "22.4868988716485",
      "3": "23.6812455268468",
      "4": "24.8175367410172",
      "5": "25.8778525229247",
      "6": "26.8454710592869",
      "7": "27.7051324277579",
      "8": "28.4432792550202",
      "9": "29.0482705246602",
      "10": "29.5105651629515",
      "11": "29.8228725072869",
      "12": "29.9802672842827",
      "13": "29.9802672842827",
      "14": "29.8228725072869",
      "15": "29.5105651629515",
      "16": "29.0482705246602",
      "17": "28.4432792550202",
      "18": "27.7051324277579",
      "19": "26.8454710592869",
      "20": "25.8778525229247",
      "21": "24.8175367410172",
      "22": "23.6812455268468",
      "23": "22.4868988716485",
      "24": "21.253332335643",
      "25": "20",
      "26": "18.746667664357",
      "27": "17.5131011283515",
      "28": "16.3187544731532",
      "29": "15.1824632589828",
      "30": "14.1221474770753",
      "31": "13.1545289407131",
      "32": "12.2948675722421",
      "33": "11.5567207449798",
      "34": "10.9517294753398",
      "35": "10.4894348370485",
      "36": "10.1771274927131",
      "37": "10.0197327157173",
      "38": "10.0197327157173",
```

```
          "39": "10.1771274927131",
          "40": "10.4894348370485",
          "41": "10.9517294753398",
          "42": "11.5567207449799",
          "43": "12.2948675722421",
          "44": "13.1545289407131",
          "45": "14.1221474770753",
          "46": "15.1824632589828",
          "47": "16.3187544731532",
          "48": "17.5131011283515",
          "49": "18.746667664357"
        },
        "TypeCode": "Double"
      }
    ]
  }
}
```


ResultTable contains the name of the result table, rows and columns

Rows is the amount of rows in each column

Column is an array of column entities, containing name, type and the result data

Data is the result data. Property name indicate index, whereas the value is the published data

Client -> Server

This socket has no client to server communication implemented.

Websocket PlatformInteractions

URL:

{url}/interactions

QueryStrings:

Key	Value	Description	Example
Id	{SessionId}	Specify the session to connect	{url}/interactions?id=d6c81036-e29c-4bee-83e9-4b7c41d63070

Server -> Client

Example of data sent from Server to Client:

A testplan containing a Dialog teststep transmits the following json:

```
{
    "Title": "WebSockets",
    "Timeout": "00:00:05",
    "RequestType": "Custom",
    "Guid": "0cc6882f-5b44-49cf-85bf-c95f32910270",
    "Requests": [{
        "Message": "Use Websockets?",
        "Response": {
            "Value": "-1",
            "Values": {
                "0": "Ok",
                "1": "Cancel"
            },
            "ReadOnly": false,
            "TypeName": "Enum"
        }
    }]
}
```

Title is the interaction title.

Timeout is the timeout of the interaction. An answer must be provided within the specified timeframe, or the default answer will be posted.

RequestType is the type of input request. Types include: - Custom - CustomModal - Metadata

Guid is the identifier for the REST-API to distinquish between multiple requests.

Requests contains the requests to be answered. These requests utilizes the same adapter system as the {url}/plan/step endpoint.

Client -> Server

To post an answer to the above request, change the “Value” of the request to the desired result and send back the whole json snippet.

```
{
    "Title": "WebSockets",
    "Timeout": "00:00:05",
    "RequestType": "Custom",
    "Guid": "0cc6882f-5b44-49cf-85bf-c95f32910270",
    "Requests": [{
        "Message": "Use Websockets?",
        "Response": {
            "Value": "0", // CHANGE THIS
            "Values": {
                "0": "Ok",
                "1": "Cancel"
```

```
            },
            "ReadOnly": false,
            "TypeName": "Enum"
        }
    }]
}
```

Websocket Events

URL:

{url}/Events

QueryStrings:

Key	Value	Description	Example
Id	{SessionId}	Specify the session to connect	{url}/logs?id=d6c81036-e29c-4bee-83e9-4b7c41d63070

Server -> Client

Example of data sent from Server to Client when test plan has been changed:

```
{
    "Event": "TestPlanChanged"
}
```

Example of data sent from Server to Client when a test step had its property changed:

```
{
    "Event": "TestStepChanged",
    "TestStepId": "0bf63f86-ce4c-423b-a2b3-9992b0cee7d1"
}
```

Event displays the event type. Currently supported: - TestPlanChanged - TestStepChanged

Client -> Server

This socket has no client to server communication implemented.

Websocket Events

URL:

{url}/proxyevents

Server -> Client

Example of data sent from Server to Client when a session has been started:

```
{
    "Session": {
        "Address": "127.0.0.1",
        "Port": "7000",
        "Id": "d42706a0-2fc6-466b-b880-6ae48bad023e",
        "Origin": "Process",
        "Image": "7E653B8478D014D0EB3CAB4A02CAC84053136A3D"
    },
    "Status": "SessionStarted"
}
```

Status is the type of event. We currently support: - SessionStarted - SessionStopped

Session contains info on IP (*Address*) and *Port* of the running session.

Id is the Proxy assigned id to this session.

Origin indicates what type of session creator that spawned this session, or if the session "SelfRegistered"

Image specifies the session image. This value will be null if session is not running in a known image configuration.

Client -> Server

This socket has no client to server communication implemented.