Create work items in Visual Studio Team Services using PowerShell

Introduction

A colleague asked me an interesting question **How can I create work items in VSTS using PowerShell?** during a PowerShell session at my work place which triggered me to share a TechNet wiki article. For me Visual Studio Team Services was a foreign matters until I made my hand dirt to gain knowledge and experience.

Requirement

Team has a source file in CSV, XML or JSON format and need an **interface** which connects to team project and create work items. The below sequence diagram describes it.

```
sequenceDiagram
        Source File->Interface: "Backlog Information"
        loop Work Items
        Interface->>Team Project: "Add Backlog Item"
    end
```

Solution

As a PowerShell enthusiast I opted to fuse Visual Studio Team Services REST API with PowerShell to accomplish the task and the authentication mechanism I selected in **P**ersonal **A**ccess **T**oken (PAT) for a demo purpose.

Create a Personal Access Token

1. Sign in to either your Visual Studio Team Services account (https://{youraccount}.visualstudio.com)
2. From your home page, open your profile (Hover on top your profile badge). Go to your **security** details.
3. In the left pane choose **personal access token**
4. Click **add** - Name your token
5. Choose the life span for your token. Default is **90 days**.

Convert Personal Access Token to Base64 String

With reference to the document I developed a nifty PowerShell snippet shared below to convert the personal access token to base 64 string.

```
$Token = "Personal Access Token"
$Authentication = [Text.Encoding]::ASCII.GetBytes(":$Token")
$Authentication = [System.Convert]::ToBase64String($Authentication)
$Headers = @{
    Authorization = ("Basic {0}" -f $Authentication)
}
$Headers.Authorization
```

```
# Output(Sample)
Basic PAT1234ID=
```

REST API Endpoint URL

The REST API endpoint URL is shown below which has three mandatory parameters project, workitemtypename and version. The method used in **PATCH** with content type **application/json-patch+json**

```
PATCH
https://{instance}/DefaultCollection/{project}/_apis/wit/workitems/${workItemT
ypeName}?api-version={version}
```

Source File

The body is the main part to meet our goal. Yes, the PowerShell script will create an item in specific team project, iteration path and assign to a team member. Underneath is the sample source file which contains value and technicians used as inputs for title and assigned to respectively.

value	technician
New work item 1	Chendrayan Venkatesan
New work item 2	Team Member
New work item 3	Team Manager

Body parameter

In our case we need to add an item with title, area path, iteration path and assign to technicians. So, we need to work with four fields where area and iteration path remains same. See the JSON values converted from hash table using PowerShell

```
@(
    @{
        op    = "add"
        path  = "/fields/System.Title"
        value = $value
    }
    @{
        op    = "add"
        path  = "/fields/System.AreaPath"
        value = "Automation\Onpoint Scrum"
    }
    @{
        op    = "add"
        path  = "/fields/System.IterationPath"
        value = "Automation\FY18-M01_JULY"
    }
    @{
        op    = "add"
        path  = "/fields/System.AssignedTo"
        value = "Chendrayan Venkatesan"
    }
) | ConvertTo-Json

# Output
[
    {
        "path":  "/fields/System.Title",
        "op":  "add",
        "value":  null
    },
    {
        "path":  "/fields/System.AreaPath",
        "op":  "add",
        "value":  "Automation\\Automation Scrum"
    },
    {
        "path":  "/fields/System.IterationPath",
        "op":  "add",
        "value":  "Automation\\FY18-M01_JULY"
    },
    {
        "path":  "/fields/System.AssignedTo",
        "op":  "add",
        "value":  "Chendrayan Venkatesan"
    }
]
```

Thanks to PowerShell team for **ConvertTo-Json** cmdlet. The structure to add work items is described below

```
graph LR
        Automation-->Automation-Scrum
        Automation-Scrum-->FY18-M01_JULY
```

References

cmdlet	help
Invoke-RestMethod	*help Invoke-RestMethod -Detailed*
ConvertTo-Json	*help ConvertTo-Json Detailed*
Import-csv	*help Import-csv -Detailed*

With no more wait let us share the PowerShell script to add work items in bulk.

```
$values = Import-csv C:\Source\WIT.csv
foreach ($value in $values)
{
    $RestParams = @{
        Uri         =
"https://{account}/Automation/_apis/wit/workitems/`$product backlog item?api-
version=1.0"
        ContentType = 'application/json-patch+json'
        Headers     = @{
            Authorization = ("Basic {0}" -f $authentication)
        }
        Method      = "Patch"
        Body        = @(
            @{
                op    = "add"
                path  = "/fields/System.Title"
                value = $value.value
            }
            @{
                op    = "add"
                path  = "/fields/System.AreaPath"
                value = "Automation\Automation Scrum"
            }
            @{
                op    = "add"
                path  = "/fields/System.IterationPath"
                value = "Automation\FY18-M01_JULY"
            }
            @{
                op    = "add"
                path  = "/fields/System.AssignedTo"
                value = $value.technician
            }
        ) | ConvertTo-Json
    }
    try
    {
        Invoke-RestMethod @RestParams -Verbose
    }
    catch
    {
        $_.Exception.Message
    }
}
```