Tutorial: using PowerShell scripts for Continuous Integration

In Windows scenarios, or in general in environments where PowerShell is available, it is relatively easy to interact with Xray using its REST API.

PowerShell scripts can be used in Continuous Integration, for example to submit the test automation results.

This may be quite useful whenever Jenkins nor Bamboo are available.

The following are just some examples of scripts that you can use, and customize, to easily submit test results.

For other formats, you should be able to adapt them accordingly.

> **Please note**
>
> Please use the latest version of PowerShell (i.e. >= 6.0) since previous versions had limitations concerning HTTP multipart support.

Submit results using Xray JSON format

In this example, we'll be making use of Xray's REST API for importing results using Xray's JSON format (more info here: Import Execution Results - REST).

It will create a Test Execution, with fixVersion "v3.0", Revision "123" for Test Environment "Chrome", assigned to the Test Plan "CALC-2208".

submit_xray.ps1

```
try {
 $user = 'admin'
 $pass = 'password'
 $jira_base_url = 'http://yourjiraserverbaseurl'
 $version = 'v3.0'
 $revision = '123'
 $environment = 'Chrome'
 $testplan = 'CALC-2208'


 $pair = "$($user):$($pass)"
 $encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
 $basicAuthValue = "Basic $encodedCreds"


$json = @"
{
    "info" : {
        "summary" : "Execution of automated tests for release v1.3",
        "description" : "This execution is automatically created when importing execution results from an
external source",
        "version" : "$($version)",
        "revision" : "$($revision)",
        "testEnvironments": ["$($environment)"],
        "testPlanKey" : "$($testplan)"
    },
    "tests" : [
        {
            "testKey" : "CALC-2",
            "start" : "2017-08-30T11:47:35+01:00",
            "finish" : "2017-08-30T11:50:56+01:00",
            "comment" : "Successful execution",
            "status" : "PASS"
        },
        {
            "testKey" : "CALC-73",
            "start" : "2017-08-30T11:47:35+01:00",
            "finish" : "2017-08-30T11:50:56+01:00",
            "comment" : "Unsuccessful execution",
            "status" : "FAIL"
        }
     ]
}
"@


 $uri = "$($jira_base_url)/rest/raven/1.0/import/execution"
 $res = Invoke-WebRequest -Uri $uri -Body $json -Method POST -ContentType "application/json" -Headers @
{"Authorization" = $basicAuthValue} }
catch {
write-host $_.Exception.Message
}
```

Submit JUnit results

In this example, we'll be making use of Xray's REST API for importing results for JUnit (more info here: Import Execution Results - REST).

It assumes the existence of a JUnit file named "junit.xml". Don't forget to update the credentials and Jira's base URL along with the project key where you want the Test Execution to be created in.

submit_junit.ps1

```
try {
 $user = 'admin'
 $pass = 'password'
 $jira_base_url = 'http://yourjiraserverbaseurl'
 $project_key = 'CALC'
 $multipartFile = "junit.xml"

 $pair = "$($user):$($pass)"
 $encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
 $basicAuthValue = "Basic $encodedCreds"
 $multipartContent = New-Object System.Net.Http.MultipartFormDataContent

 $FsMode = [System.IO.FileMode]::Open
 $FsAccess = [System.IO.FileAccess]::Read
 $FsSharing = [System.IO.FileShare]::Read
 $FileStream = New-Object System.IO.FileStream($multipartFile, $FsMode, $FsAccess, $FsSharing)
 $fileHeader = New-Object System.Net.Http.Headers.ContentDispositionHeaderValue("form-data")
 $fileHeader.Name = "file"
 $fileHeader.FileName = $multipartFile
 $fileContent = New-Object System.Net.Http.StreamContent($FileStream)
 $fileContent.Headers.ContentDisposition = $fileHeader
 $fileContent.Headers.ContentType = [System.Net.Http.Headers.MediaTypeHeaderValue]::Parse("text/xml")
 $multipartContent.Add($fileContent)
 $uri = "$($jira_base_url)/rest/raven/1.0/import/execution/junit?projectKey=$($project_key)"
 $res = Invoke-WebRequest -Uri $uri -Body $multipartContent -Method POST -Headers @{"Authorization" =
$basicAuthValue} }
catch {
write-host $_.Exception.Message
}
```

Submit JUnit results and customize Test Execution

In this example, we'll be making use of Xray's REST API for importing results for JUnit, namely the multipart endpoint (more info here: Import Execution Results - REST).

By using the multipart endpoint for JUnit, we're able to customize fields of the Test Execution that is going to be created in Jira. For that we need to specify a JSON content, either inline or in a file, and submit it in the same request. The format of this JSON object follows the syntax of Jira's own REST API for creating issues. In the provided script, you can

It assumes the existence of a JUnit file named "junit.xml". Don't forget to update the credentials and Jira's base URL along with the project key where you want the Test Execution to be created in. You may need to know some values beforehand, such as the id of the Test Execution Issue Type in Jira.

submit_junit_multipart.ps1

```
try
{

 $user = 'admin'
 $pass = 'password'
 $jira_base_url = 'http://yourjiraserverbaseurl'
 $project_key = 'CALC'
 $multipartFile = "junit.xml"
 $infoFile = "exec_info.json"

 $pair = "$($user):$($pass)"
 $encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
 $basicAuthValue = "Basic $encodedCreds"
 $multipartContent = New-Object System.Net.Http.MultipartFormDataContent
 $FsMode = [System.IO.FileMode]::Open
 $FsAccess = [System.IO.FileAccess]::Read
 $FsSharing = [System.IO.FileShare]::Read
```

```
 $FileStream = New-Object System.IO.FileStream($multipartFile, $FsMode, $FsAccess, $FsSharing)
 $fileHeader = New-Object System.Net.Http.Headers.ContentDispositionHeaderValue("form-data")
 $fileHeader.Name = "file"
 $fileHeader.FileName = $multipartFile
 $fileContent = New-Object System.Net.Http.StreamContent($FileStream)
 $fileContent.Headers.ContentDisposition = $fileHeader
 $fileContent.Headers.ContentType = [System.Net.Http.Headers.MediaTypeHeaderValue]::Parse("text/xml")
 $multipartContent.Add($fileContent)


# additional information for the Test Execution issue; it follows the syntax of Jira REST API for updating
fields
$json = @"
{
    "fields": {
        "summary": "Test Execution for junit Execution",
        "project": {
            "key": "$($project_key)"
        },
        "issuetype": {
            "id": "9"
        },
        "components" : [
            {
            "name":"ui"
            },
            {
            "name":"core"
            }
        ]
    }
}
"@


$fileHeader = New-Object System.Net.Http.Headers.ContentDispositionHeaderValue("form-data")
$fileHeader.Name = "info"
$fileHeader.FileName = $infoFile
$stream = [IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes($json))
$infoContent = New-Object System.Net.Http.StreamContent($stream)
$infoContent.Headers.ContentDisposition = $fileHeader
$infoContent.Headers.ContentType = [System.Net.Http.Headers.MediaTypeHeaderValue]::Parse("application/json")
$multipartContent.Add($infoContent)

<#
 # in case you want to read  the  Json metainformation from a file instead
 $FsMode = [System.IO.FileMode]::Open
 $FsAccess = [System.IO.FileAccess]::Read
 $FsSharing = [System.IO.FileShare]::Read
 $FileStream = New-Object System.IO.FileStream($infoFile, $FsMode, $FsAccess, $FsSharing)
 $fileHeader = New-Object System.Net.Http.Headers.ContentDispositionHeaderValue("form-data")
 $fileHeader.Name = "info"
 $fileHeader.FileName = $infoFile
 $fileContent = New-Object System.Net.Http.StreamContent($FileStream)
 $fileContent.Headers.ContentDisposition = $fileHeader
 $fileContent.Headers.ContentType = [System.Net.Http.Headers.MediaTypeHeaderValue]::Parse("application/json")
 $multipartContent.Add($fileContent)
#>

 $uri = "$($jira_base_url)/rest/raven/1.0/import/execution/junit/multipart)"
 $res = Invoke-WebRequest -Uri $uri -Body $multipartContent -Method POST -Headers @{"Authorization" =
$basicAuthValue} }
catch {
 write-host $_.Exception.Message
}
```