Precondition script:

(stepStatus.allPriorIn(SUCCESS)) && properties.equals('build\_envName','qa')

stepStatus.allPriorIn(SUCCESS, SUCCESS\_WARN) && properties.get('RepositoryName') == 'webcomponent' && properties.get('BranchName') != 'mc-dev'

stepStatus.allPriorIn(SUCCESS, SUCCESS\_WARN) && properties.containsValue('Branch', 'uplan-\*')

stepStatus.allPriorIn(SUCCESS, SUCCESS\_WARN) && (properties.get("Warning") == "true")

Post-processing script:

TFS cleanup

properties.put("Status", "Success");

if (properties.get("exitCode") != 0) {

properties.put("Status", "Failure");

scanner.register("(?i)No pending changes were found", function(lineNumber, line) {

var errors = properties.get("Error");

if (errors == null) {

errors = new java.util.ArrayList();

properties.put("Error", errors);

properties.put("Status", "Success");

}

errors.add(line+"\n");

});

scanner.scan();

}

else {

scanner.register("(?i)Collection Not Found", function(lineNumber, line) {

var errors = properties.get("Error");

if (errors == null) {

errors = new java.util.ArrayList();

properties.put("Error", errors);

properties.put("Status", "Failure");

}

errors.add(line+"\n");

});

scanner.scan();

}

++++++++++++++++++++++

Update version num:

Current\_Version=${p:VersionNumber}

echo "Version is $Current\_Version"

+++++++++++++++++++

SAFE build check

properties.put("Status", "Success");

if (properties.get("exitCode") != 0) {

properties.put("Status", "Failure");

scanner.register("(?i)The given base directory is not a directory", function(lineNumber, line) {

var errors = properties.get("Error");

if (errors == null) {

errors = new java.util.ArrayList();

properties.put("Error", errors);

properties.put("Status", "Success");

}

errors.add(line+"\n");

});

scanner.scan();

}

else {

scanner.register("(?i)The given base directory is not a directory", function(lineNumber, line) {

var errors = properties.get("Error");

if (errors == null) {

errors = new java.util.ArrayList();

properties.put("Error", errors);

properties.put("Status", "Failure");

}

errors.add(line+"\n");

});

scanner.scan();

}

++++++++++++++++++++++++

Magic- poc

var exit = properties.get('exitCode');

commandOut.println('Verify Post-Proc');

scanner.register("regex", function(lineNumber, line) {

commandOut.println('Scanner Register');

var thing = 'do stuff';

});

if (exit == 0) {

commandOut.println('The exit code in zero');

properties.put('Status', 'Success');

} else {

commandOut.println('Before Regi');

scanner.register("^No pending changes were found for", function(lineNumber, line) {

properties.put('Status', 'Success');

commandOut.println('Update Status');

});

scanner.scan();

}

++++++++++++++++++++  
Notification template:

## BEGIN SECTION Subject

#set($project = $workflow.Project)

#set($stamp = $workflow.BuildLife.latestStamp())

#set($env = $buildLife.properties.EnvName)

#if($stamp != "")

$env:: $workflow.Name:: Build

#else

$env:: $workflow.Name:: Build

#end

#if($workflow.Status.equalsIgnoreCase("Complete"))

Success

#else

$workflow.Status

#end

## END SECTION Subject

## BEGIN SECTION Body

## PROPERTY Content-Type: text/html

## PROPERTY X-Priority: 3

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title>Process</title>

<STYLE TYPE="text/css">

<!--

table.data-table td {

vertical-align: top;

}

table.data-table

{

font-family: arial, helvetica, sans-serif;

font-size: 12px;

background-color: #567596;

}

table.data-table caption

{

padding-top: 10px;

padding-bottom: 10px;

text-align: left;

}

table.data-table th

{

text-align: center;

background-color: #cfdbef;

height: 25px;

}

table.data-table td

{

vertical-align: top;

}

table.data-table tr.odd

{

background-color: #ffffff;

}

table.data-table tr.even

{

background-color: #f6f6f6;

}

.data-table-button-bar

{

padding-top: 10px;

padding-bottom: 10px;

}

.data-table-container

{

padding-top: 10px;

padding-bottom: 10px;

}

-->

</STYLE>

</head>

<body>

<h2> Build Process: $workflow.Name</h2>

<table>

<tr><td>Env:</td>

<td>$buildLife.properties.EnvName</td>

</tr>

<tr><td>RequestedBy:</td>

<td>$request.getRequesterName() $request.user.getActualName()</td>

</tr>

<tr><td>ProjectName:</td>

<td>$request.getProperties().get("ProjectName")</td>

</tr>

<tr><td>Interface:</td>

<td>$request.getProperties().get("InterfaceName")</td>

</tr>

<tr><td>ArchiveName:</td>

<td>$request.getProperties().get("ArchiveFiles")</td>

</tr>

</table>

<div class="data-table-container">

<table class="data-table" cellpadding="4" cellspacing="1" width="100%">

<table-body>

<tr class="data-table-head">

<th scope="col" align="left" valign="middle"><strong>Job Name</strong></th>

<th scope="col" align="left" valign="middle"><strong>Agent Name</strong></th>

<th scope="col" align="left" valign="middle">Status</strong></th>

</tr>

#foreach($trace in $workflow.JobTraceList)

<tr bgcolor="#ffffff">

<td>$trace.Name</td>

<td align="center">$trace.Agent</td>

#set($status = $trace.Status)

<td align="center">

#if($trace.statusIsSuccess())

<strong style="color: green">

#elseif($trace.statusIsFailure())

<strong style="color: red">

#else

<strong style="color: blue">

#end

$status</strong>

</td>

</tr>

#foreach($step in $trace.StepList)

<tr bgcolor="#ffffff">

<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Step - $step.Name</td>

<td align="center">$trace.Agent</td>

#set($stepStatus = $step.Status)

<td align="center">

#if($step.statusIsSuccess())

<strong style="color: green">

#elseif($step.statusIsFailure())

<strong style="color: red">

#else

<strong style="color: blue">

#end

$stepStatus</strong>

</td>

</tr>

#end

#end

</table-body>

</table>

</div>

<a href="$workflow.Url">Build details</a> | <a href="$util.UserSubscriptionsUrl">Update subscriptions</a>

</body>

</html>

## END SECTION Body

++++++++++++++++++

Get Change set items- -- groovy script

String sampleString = "${p:changeset}";

StringBuffer sbout = new StringBuffer()

StringBuffer sberr = new StringBuffer()

def srcDir = new File("D:\\UCB\_PARIS\\${p:Branch}\\Database\\PARIS\_DEV")

String[] commitids = sampleString.split(",");

System.out.println("The number of commitids is: " + commitids.length);

for (String commitid : commitids) {

System.out.println(commitid);

def command = "tf changeset ${p:commitid} /noprompt /server:http://tfs.inbcu.com:8080/tfs/PARIS\_Projects | findstr /c:\"merge, edit\" /c:\"merge, branch\" /c:\"edit \""

println ("[$command]")

Process process2 = ['cmd', '/c', command].execute(null, srcDir)

process2.waitForProcessOutput sbout, sberr

def value = process2.exitValue()

println "${sberr.toString()}"

println "${sbout.toString()}"

String buildfiles = sbout.toString();

buildfiles = buildfiles.replaceAll("merge, edit","")

buildfiles = buildfiles.replaceAll(" merge, branch","")

buildfiles = buildfiles.replaceAll(" edit","")

println "${buildfiles}";

outProps.put( "buildlife/changedItems", buildfiles )

}

+++++++++++++++++++++

Get changeset items- autobuild – groovy

String sampleString = "${p:buildlife/changeSetScmIds}";

StringBuffer sbout = new StringBuffer()

StringBuffer sberr = new StringBuffer()

def srcDir = new File("D:\\UCB\_PARIS\\${p:Branch}\\Database\\PARIS\_DEV")

String[] commitids = sampleString.split(",");

System.out.println("The number of commitids is: " + commitids.length);

for (String commitid : commitids) {

System.out.println(commitid);

def command = "tf changeset ${p:commitid} /noprompt /server:http://tfs.inbcu.com:8080/tfs/PARIS\_Projects | findstr /c:\"merge, edit\" /c:\"merge, branch\" /c:\"edit \""

println ("[$command]")

Process process2 = ['cmd', '/c', command].execute(null, srcDir)

process2.waitForProcessOutput sbout, sberr

def value = process2.exitValue()

println "${sberr.toString()}"

println "${sbout.toString()}"

String buildfiles = sbout.toString();

buildfiles = buildfiles.replaceAll("merge, edit","")

buildfiles = buildfiles.replaceAll(" merge, branch","")

buildfiles = buildfiles.replaceAll(" edit","")

println "${buildfiles}";

outProps.put( "buildlife/changedItems", buildfiles )

}

++++++++++++++++++

Powershell script to copy the files.

$changepath = '${p:buildlife/changedItems}'

Write-Host "changes are $changepath"

if (!$changepath ) {

Write-Host "variable is null"

exit 1

}

else {

ForEach($value in $changepath.Split("`n`r")) {

$source = $value.Trim()

if ($source -eq "") {

Write-Host "This is empty string"

} else {

$srcdir = 'D:\UCB\_PARIS\${p:Branch}\Database\PARIS\_DEV'

$changescript\_srcdir = 'D:\UCB\_PARIS\${p:Branch}\Database\PARIS\_DEV\Change\_Scripts'

$dstdir = 'D:\UCB\_PARIS\Publish\_DB'

$Change\_Scripts\_dstdir = 'D:\UCB\_PARIS\Publish\_DB\Change\_Scripts'

Write-Host "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

Write-Host "$value"

$folder = Split-Path $source -Leaf

$parent = Split-Path $source

$parentfolder = Split-Path $parent -Leaf

if ($parentfolder -eq "PARIS\_DEV") {

Write-Host "\*\*\*\*\*\*\*\*Not required\*\*\*\*\*\*\*\*\*\*\*\*"

}

elseif ($parentfolder -eq "Types") {

Write-Host "\*\*\*\*\*\*\*\*Not required\*\*\*\*\*\*\*\*\*\*\*\*"

}

elseif ($parentfolder -like "PAR-\*") {

Write-Host "copying files from Change\_Scripts\PAR-"

$lasttwo = Join-Path $parentfolder $folder

$srcfile = $lasttwo -replace "\\","/"

New-Item -Path $Change\_Scripts\_dstdir -Name "$parentfolder" -ItemType "directory" -Force

if (Test-Path -Path $changescript\_srcdir\$srcfile) {

Copy-Item -Path $changescript\_srcdir\$srcfile -Destination $Change\_Scripts\_dstdir\$parentfolder\ -Force

}

else {

Write-Host "file $changescript\_srcdir\$srcfile doesn't exists"

}

}

else {

$lasttwo = Join-Path $parentfolder $folder

$srcfile = $lasttwo -replace "\\","/"

Write-Host "copy file from $srcdir\$srcfile to to $dstdir\$parentfolder\"

New-Item -Path $dstdir -Name "$parentfolder" -ItemType "directory" -Force

Write-Host "the copy command is Copy-Item -Path $srcfile -Destination $dstdir\$parentfolder\ -Force"

if (Test-Path -Path $srcdir\$srcfile) {

Copy-Item -Path $srcdir\$srcfile -Destination $dstdir\$parentfolder\ -Force

}

else {

Write-Host " File $srcdir\$srcfile doesn't exists"

}

}

}

}

}

++++++++++++++++++++++

Evaluate env- groovy

String ucdenv = ""

String branch = "${p:triggeredBranch}";

if ((branch.startsWith("rel"))) {

ucdenv = "DEV\_Node\_2";

} else if ((branch.startsWith("dev"))) {

ucdenv = "DEV\_Node\_1";

} else {

ucdenv = ""

}

outProps.put( 'buildlife/ucd.env', ucdenv )

outProps.put( 'request/ucd.env', ucdenv )

++++++++++++++++

Evaluate Env

def RSTDEV = "RTS Dev"

def RTSQA = "RTS QA"

def RTSUAT = "RTS UAT"

def SunlightDEV = "Sunlight Dev"

def SunlightQA = "Sunlight QA"

def SunlightUAT = "Sunlight UAT"

def Staging = "Staging"

String sampleString = "${p:ucd.env}";

String[] Environments = sampleString.split(",");

System.out.println("The number of environments is: " + Environments.length);

for (String Environment : Environments) {

System.out.println(Environment);

if (Environment == RSTDEV) {

outProps.setProperty( "buildlife/ucd.env1", RSTDEV )

}

if (Environment == RTSQA) {

outProps.setProperty( "buildlife/ucd.env2", RTSQA )

}

if (Environment == RTSUAT) {

outProps.setProperty( "buildlife/ucd.env3", RTSUAT )

}

if (Environment == Staging) {

outProps.setProperty( "buildlife/ucd.env4", Staging )

}

if (Environment == SunlightDEV) {

outProps.setProperty( "buildlife/ucd.env5", SunlightDEV )

}

if (Environment == SunlightQA) {

outProps.setProperty( "buildlife/ucd.env6", SunlightQA )

}

if (Environment == SunlightUAT) {

outProps.setProperty( "buildlife/ucd.env7", SunlightUAT )

}

}

+++++++++++++++++++++

Post commit process to trigger build upon specific branch:

import groovy.json.JsonSlurper

def jsonSlurper = new JsonSlurper()

def object = jsonSlurper.parseText('${p:postData}')

def TAG = (object.ref);

def ReqTYPE = (object.ref\_type);

def Reponame = (object.repository.full\_name);

def TAG\_Name = TAG[-1];

def Repository = Reponame.split('/')[1];

System.out.println(" The tag is: " + TAG);

System.out.println(" The Request type is:" + ReqTYPE);

System.out.println(" The tag is: " + TAG\_Name);

System.out.println(" The Reponame is:" + Reponame);

System.out.println(" The Repository is:" + Repository);

if (ReqTYPE.equals("tag") && TAG\_Name.equals("0") && Repository.equals("Score-Olympic")) {

outProps.setProperty( "buildlife/GHETag", TAG );

outProps.setProperty( "buildlife/GHEReq", ReqTYPE );

outProps.setProperty( "buildlife/Repository", Repository );

System.out.println("updated TAG property and executing curl for TAG $TAG and repository $Repository");

def TRIGGER\_URL = "http://100.117.38.68:8080/trigger";

def CODE = "9667753ba8c8286280fe7edcdb576a9838f868af";

def process = [ 'bash', '-c', "curl --retry 1 -k -d \'code=$CODE&GHETAG=$TAG\' \"$TRIGGER\_URL\"" ].execute();

process.waitFor();

println process.err.text;

println process.text

} else if (ReqTYPE.equals("tag") && TAG\_Name.equals("0") && Repository.equals("Score-Remote")) {

outProps.setProperty( "buildlife/GHETag", TAG );

outProps.setProperty( "buildlife/GHEReq", ReqTYPE );

System.out.println("updated TAG property and executing curl for TAG $TAG and repository $Repository");

outProps.setProperty( "buildlife/Repository", Repository );

def TRIGGER\_URL = "http://100.117.38.68:8080/trigger";

def CODE = "1c21dfacb2a7dcb79e76f330fbc15cb24aa6f090";

def process = [ 'bash', '-c', "curl --retry 1 -k -d \'code=$CODE&GHETAG=$TAG\' \"$TRIGGER\_URL\"" ].execute();

process.waitFor();

println process.err.text;

println process.text

} else if (ReqTYPE.equals("tag") && TAG\_Name.equals("0") && Repository.equals("Score-Talent")) {

outProps.setProperty( "buildlife/GHETag", TAG );

outProps.setProperty( "buildlife/GHEReq", ReqTYPE );

outProps.setProperty( "buildlife/Repository", Repository );

System.out.println("updated TAG property and executing curl for TAG $TAG and repository $Repository");

def TRIGGER\_URL = "http://100.117.38.68:8080/trigger";

def CODE = "a3c9a0aecd55798bb9db790454f2cd27325aa75d";

def process = [ 'bash', '-c', "curl --retry 1 -k -d \'code=$CODE&GHETAG=$TAG\' \"$TRIGGER\_URL\"" ].execute();

process.waitFor();

println process.err.text;

println process.text

} else if (ReqTYPE.equals("tag") && TAG\_Name.equals("0") && Repository.equals("Score-PickADoor")) {

outProps.setProperty( "buildlife/GHETag", TAG );

outProps.setProperty( "buildlife/GHEReq", ReqTYPE );

outProps.setProperty( "buildlife/Repository", Repository );

System.out.println("updated TAG property and executing curl for TAG $TAG and repository $Repository");

def TRIGGER\_URL = "http://100.117.38.68:8080/trigger";

def CODE = "6eaf910c9f4e01435418dc5c10f6ddddef7eb789";

def process = [ 'bash', '-c', "curl --retry 1 -k -d \'code=$CODE&GHETAG=$TAG\' \"$TRIGGER\_URL\"" ].execute();

process.waitFor();

println process.err.text;

println process.text

} else if (ReqTYPE.equals("tag") && TAG\_Name.equals("0") && Repository.equals("ccasportsdotnet")) {

outProps.setProperty( "buildlife/GHETag", TAG );

outProps.setProperty( "buildlife/GHEReq", ReqTYPE );

System.out.println("updated TAG property and executing curl for TAG $TAG and repository $Repository");

outProps.setProperty( "buildlife/Repository", Repository );

def TRIGGER\_URL = "http://100.117.38.68:8080/trigger";

def CODE = "828ba327ad7b146acf61afde56f813a1127ff8f0";

def process = [ 'bash', '-c', "curl --retry 1 -k -d \'code=$CODE&GHETAG=$TAG\' \"$TRIGGER\_URL\"" ].execute();

process.waitFor();

println process.err.text;

println process.text

} else {

outProps.setProperty( "buildlife/GHEReq", ReqTYPE );

outProps.setProperty( "buildlife/Repository", Repository );

System.out.println("updated branch and repo property")

}

++++++++++++++++++++++

Azure build to generate zip

Configuration=Release

DeployOnBuild=true

WebPublishMethod=Package

PackageAsSingleFile=true

DesktopBuildPackageLocation=WebApp.zip

DeployIisAppPath=Default Web Site

Jira acli to update jira issues

cd /d D:\Jira-CLI\atlassian-cli-9.1.0

set PATH=C:\ProgramData\Oracle\Java\javapath;${env/PATH}

acli jira --server ${p:jira-url} --user ${p:jira\_user} --password ${p:jira-password} --action runFromIssueList --jql "project = 'SCOR' AND issuetype in (Bug, Story, Production\ Issue) AND status = Resolved AND fixVersion in ('${p:fixVersion}') AND labels = BuildReady ORDER BY assignee ASC" --common "--issue @issue@" -i "-a modifyFieldValue --field "labels" --findReplace \"BuildReady:QAReady\"" -i "-a setFieldValue --field assignee --value \"${p:Jira\_QA\_User}\""

++++++++++++++++++++  
${p:request/requestType}