From: Bhatnagar, Rohit
To: Bhatnagar, Rohit
Subject: All OC Groovy Scripts
Date: Wednesday, October 13, 2021 10:34:48 AM
Attachments: image001.png

Backed out to be invoked via shared library only to prevent misuse….

```
#groovy
///All groovy scripts defined in OC
///---------- AgentsLabelsAndExecutors ---------
#!Groovy - Rohit K. Bhatnagar
import java.text.SimpleDateFormat
def iCount = 1
def iExecutors = 0
def iAgents = 0

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"
//------------------------------------------------
```

```
/*** BEGIN META {
  "name" : "Show labels overview and executors count",
  "comment" : "Show an overview of all labels defined & executors count for ea
ch slave",
  "author" : "Rohit K. Bhatnagar"
} END META**/
import jenkins.model.Jenkins;
```

```
println "=============================================================================
======================"
println "Displays Labels defined and Slaves which have these Labels alongwith
their total executors"
println "=============================================================================
======================"
println ">>>>>>>>> CD Master - ${java.net.InetAddress.getLocalHost()} <<<<<<<<
<<<<<<<<<<<<<<<<<<<<"

def uniqueLabels = []
def slave_label_map = [:]
def executors_map = [:]
for (slave in Jenkins.instance.slaves)
{
    println "${iCount++},${slave.nodeName},${slave.labelString},${slave.numExe
cutors}" //Prints the LABELS associated and EXECUTORS count in a single line f
or each slave
```

```
    words = slave.labelString.split()
    def labelListForSlave = []
    words.each()
    {
        labelListForSlave.add(it);
        uniqueLabels.add(it)
    }
    slave_label_map.put(slave.name, labelListForSlave)
    executors_map.put(slave.name, slave.numExecutors)
}
//Adding this line to add 'EXECUTORS' as the last column entry to LABELS
uniqueLabels.add('EXECUTORS')

uniqueLabels.unique()

maxLen=0
uniqueLabels.each()
{
    if (it.length() > maxLen)
    {
        maxLen=it.length()
    }
}

def vertLabels = []

for (int idx=0; idx < maxLen; idx++)
{
    def vertLabel="|"
    uniqueLabels.each()
    {
        if (idx < it.length())
        {
            vertLabel+="${it[idx]}|"
        }
        else
        {
            vertLabel+=" |"
        }
    }
    vertLabels.add(vertLabel)
}


def FIXED_FIRST_COLUMN = 40

vertLabels.each()
{
```

```
    printSign(FIXED_FIRST_COLUMN-1, " ")
    print "${it}\n"
}
printLine()

//Loop to display Slave-Name and associated label and executors
for ( entry in slave_label_map )
{
    def slaveName = entry.key
    print "${slaveName}"
    iAgents++
    printSign(FIXED_FIRST_COLUMN - entry.key.size()-1, " ")
    print "|"
    uniqueLabels.each()
    {
        lab ->
        boolean found = false
        entry.value.each()
        {
            valueList ->
            if(lab.equals(valueList))
            {
                found = true
            }
        }
        if(found)
        {
            print "X"
        }
        else
        {
            if(!lab.equals('EXECUTORS')) //Added to adjust for 'EXECUTORS' ent
ry
                print " "
        }
        if(!lab.equals('EXECUTORS')) //Added to adjust for 'EXECUTORS' entry
            print "|"
    }
    //Added to print the executors against each slave
    for( execEntry in executors_map )
    {
        if(execEntry.key == slaveName)
        {
            //println "${execEntry.key} - ${slaveName}"
            iExecutors += execEntry.value
            print "${execEntry.value}"
            print "|"
        }
```

```
    }
    printLine()
}


def printSign(int count, String sign)
{
    for (int i = 0; i < count; i++)
    {
        print sign
    }
}

def printLine()
{
    print "\n";
    //printSign(120, "-")
    //print "\n";
}
```

```
println "${java.net.InetAddress.getLocalHost()} - Total Agents - ${iAgents} an
d total Executors - ${iExecutors}"
```

```
//-------------------------------------------------
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"

use(groovy.time.TimeCategory)
{
    def duration = endDate - startDate
    print "Elapsed time: Days: ${duration.days}, Hours: ${duration.hours}, Min
utes: ${duration.minutes}, Seconds: ${duration.seconds}"
}
//==============================
/// ------------ AllBuildsAndTheirAge ------------
#!groovy
//Display list of projects that were built more than 1 day ago.

import hudson.slaves.*
import java.util.concurrent.*
import org.jenkinsci.plugins.workflow.job.WorkflowJob
import java.util.Calendar
import java.text.SimpleDateFormat
import java.util.Date

//import java.util.GregorianCalendar
```

```
jenkins = Hudson.instance
controller = Jenkins.getInstance().getComputer('').getHostName()
println "================================================================================
================================================"
println "Counts of all builds and their respective age including whether they
are disabled, frozen/patch titled or never run!"
println "Report collected on the Controller with hostname - ${controller}"
println "================================================================================
================================================"

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"
```

```
Calendar c = Calendar.getInstance()
Calendar sevenDays = Calendar.getInstance()
Calendar fifteenDays = Calendar.getInstance()
Calendar thirtyDays = Calendar.getInstance()
Calendar fortyFiveDays = Calendar.getInstance()
Calendar ninetyDays = Calendar.getInstance()
Calendar oneHundredEightyDays = Calendar.getInstance()
Calendar oneYear = Calendar.getInstance()

now = c.instance
def date = new Date()
c.setTime(date);

sevenDays.add(Calendar.DATE, -7)
fifteenDays.add(Calendar.DATE, -15)
thirtyDays.add(Calendar.DATE, -30)
fortyFiveDays.add(Calendar.DATE, -45)
ninetyDays.add(Calendar.DATE, -90)
oneHundredEightyDays.add(Calendar.DATE, -180)
oneYear.add(Calendar.DATE, -365)

println("Starting report generation at - ${now.time}")
println "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++"
def allItems = Jenkins.instance.getAllItems()
def iItemCount = 0

//println("item count=${allItems.size()}")

// counts is an array to collect stats
//      0       1       2       3       4       5       6       7       8   9
  10
```

```
// |one year|180 days|90 days|45 days|30 days|15 days|7 days|0-
7 days|disabled|freeze|never
def counts = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] as int[]
def build_time;

for (item in allItems) {
    //println("\t ${iItemCount++} - ${item.name}")
    try {
        if (item.name ==~ /(?i)(freeze|patch).*/) {
            counts[9]++
            continue
        }
        if (item.disabled) {
            counts[8]++
            continue
        }

    //Skip if item is say folder name
    try{
        build_time = item.getLastBuild()
    }catch (Exception e){
        //Do nothing as this maybe a folder
        continue;
    }

        if (build_time != null) {
        build_time = item.getLastBuild().getTimestamp()

            if (build_time.compareTo(oneYear) < 1) {
                counts[0]++
                continue
            }
            if (build_time.compareTo(oneHundredEightyDays) < 1 && build_time.c
ompareTo(oneYear) >= 1) {
                counts[1]++
                continue
            }
            if (build_time.compareTo(ninetyDays) < 1 && build_time.compareTo(o
neHundredEightyDays) >= 1) {
                counts[2]++
                continue
            }
            if (build_time.compareTo(fortyFiveDays) < 1 && build_time.compareT
o(ninetyDays) >= 1) {
                counts[3]++
                continue
            }
            if (build_time.compareTo(thirtyDays) < 1 && build_time.compareTo(f
```

```
ortyFiveDays) >= 1) {
                counts[4]++
                continue
            }
            if (build_time.compareTo(fifteenDays) < 1 && build_time.compareTo(
thirtyDays) >= 1) {
                counts[5]++
                continue
            }
        if (build_time.compareTo(sevenDays) < 0 && build_time.compareTo(fiftee
nDays) >= 1) {
                counts[6]++
                continue
            }
        if (build_time.compareTo(sevenDays) >= 1) {
                counts[7]++
                continue
            }
        } else {
            //never built
            counts[10]++
        }
    }
    catch (Exception e) {
        println "Exception raised ... - ${e}"
    }
}

println("\t|one year\t|180 days\t|90 days\t|45 days\t|30 days\t|15 days\t|7 da
ys\t\t|0-7 days\t|disabled\t\t|frozen\t\t|never\t\t|")
println("\t=========================================================================
=====================================================")
println("\t|one year+ : ${counts[0]}\t|180+ days : ${counts[1]}\t|90+ days : $
{counts[2]}\t|45+ days : ${counts[3]}\t|30+ days : ${counts[4]}\t|\
15+ days : ${counts[5]}\t|7+ days : ${counts[6]}\t|0-
7 days : ${counts[7]}\t|disabled : ${counts[8]}\t|freeze : ${counts[9]}\t|neve
r : ${counts[10]}\t|")
println("Total all time builds in ${controller} are : ${counts.sum() - (counts
[10] + counts[8] + counts[9])} where total item count is : ${allItems.size}")

println "#########################################################################
###########"
println("\tOne year, 180 days, 90 days, 45 days, 30 days, 15 days, 7 days, 0-
7 days, disabled, frozen, never")
println("\t${counts[0]}, ${counts[1]}, ${counts[2]}, ${counts[3]}, ${counts[4]
}, ${counts[5]}, ${counts[6]}, ${counts[7]}, ${counts[8]}, ${counts[9]}, ${cou
nts[10]}");
```

```
//-------------------------------------------------
println "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++"
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"

//====================
// ----------------- CollectJavaVersionsOnNodes ------------------
#!Groovy

import java.text.SimpleDateFormat
def iCount = 1

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"

//Master HostName and IP Address
println "Master HostName and IP Address - ${java.net.InetAddress.getLocalHost(
)}"

//-------------------------------------------------------------------------
//All Nodes - Java Version
//-------------------------------------------------------------------------

println "-------------------------------------------------------------------
------"
println "All Nodes - Java Instances installed as well as default version"
println "-------------------------------------------------------------------
------"

import hudson.model.Node
import hudson.model.Slave
import jenkins.model.Jenkins
import hudson.util.RemotingDiagnostics

Jenkins jenkins = Jenkins.instance
for (Node node in jenkins.nodes) {
  // Make sure slave is online
  if (!node.toComputer().online) {
    println "Node '$node.nodeName' is currently offline - skipping check"
    continue;
  } else {
    props = node.toComputer().getSystemProperties();
    println "Node '$node.nodeName' is running " + props.get('java.runtime.vers
ion');

    String agent_name = node.nodeName;
```

```
    try {
        //groovy script you want executed on an agent
        // String groovy_script = '''
        // println System.getenv("PATH")
        // println "uname -a".execute().text
        // '''.trim()

        boolean isUnix = Boolean.TRUE.equals(node.toComputer().isUnix());
        //println "Linux instance - ${isUnix}"
        String groovy_script
        if(isUnix)
        {
            groovy_script = '''
            println "ls -latrch /sys_apps_01/java/".execute().text
            '''.trim()
        }
        else
        {
            groovy_script = '''
            println "dir /cygdrive/c/'Program Files'/Java".execute().text
            '''.trim()
        }

        String result = RemotingDiagnostics.executeGroovy(groovy_script, node.
channel);
        println result;
    }
    catch(Exception exp)
    {
        println "${exp.message}"
    }
  }
}

//---------------------------------------------------------------------------
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"

use(groovy.time.TimeCategory)
{
    def duration = endDate - startDate
    print "Elapsed time: Days: ${duration.days}, Hours: ${duration.hours}, Min
utes: ${duration.minutes}, Seconds: ${duration.seconds}"
}

//============================
// ---------------- collecttuxedoifany -----------------
```

```
#!Groovy

import hudson.util.RemotingDiagnostics;
```

```
script = 'def proc = "ls -1 /sys_apps_01/tuxedo11".execute(); proc.waitFor();
println proc.in.text';

for (slave in Jenkins.instance.slaves) {
  println slave.name;
  try {
    println RemotingDiagnostics.executeGroovy(script, slave.getChannel());
  } catch (all) {
    all.printStackTrace();
  }
}

//====================================
// ----------------- CompressWorkSpace --------------------
// ++++++++++++ DANGEROUS +++++++++++++++++++
def sout = new StringBuilder(), serr = new StringBuilder()
def proc = 'ls -latrch /sys_apps_01/jenkins2/'.execute()
//def proc = 'tar cvzf /sys_apps_01/jenkins2/workspace-
e059704.tar.gz /sys_apps_01/jenkins2/workspace/* >workspace-
e059704.txt'.execute()
proc.consumeProcessOutput(sout, serr)
//proc.waitForProcessOutput()
proc.waitForOrKill(1000)
//println "out> $sout err> $serr"
println "$sout"
println " ------- ***************************** -------------- "
println "$serr"

//==================================
/// ---------------- ControllerNodesExecutorTotals ------------------------
import com.cloudbees.opscenter.server.model.*;
import com.cloudbees.opscenter.server.clusterops.steps.*;
import hudson.remoting.*;

def cjoc = getHost(new LocalChannel(), OperationsCenter.class.simpleName, Oper
ationsCenter.class.simpleName)

cjoc.masters = []
Jenkins.instance.getAllItems(ConnectedMaster.class).each {
  cjoc.masters.add(getHost(it.channel, it.class.simpleName, it.encodedName))
}
```

```
try{
  cjoc.summary = [
    masters:cjoc.masters.size() + 1, //masters + cjoc
    masterCores:cjoc.masters*.cores.sum() + cjoc.cores,
    executors:cjoc.nodes*.executors.sum() + cjoc.masters*.nodes*.executors.sum
().sum(),
    knownCloudExecutors:cjoc.masters*.clouds*.executorsCap.sum().findAll{it}.s
um(0) + cjoc.clouds*.executorsCap.findAll{it}.sum(0)
  ]
}catch(e){}

def getHost(channel, type, name){
  def host
  if(channel){
    def stream = new ByteArrayOutputStream();
    def listener = new StreamBuildListener(stream);
    channel.call(new MasterGroovyClusterOpStep.Script("""
      //master, regular slaves, and shared slaves
      def nodes = []
      (Jenkins.instance.computers.grep {
          it.class.superclass?.simpleName != 'AbstractCloudComputer' &&
          it.class.superclass?.simpleName != 'AbstractCloudSlave' &&
          it.class.simpleName != 'EC2AbstractSlave'
        } + Jenkins.instance.getAllItems(com.cloudbees.opscenter.server.model.
SharedSlave.class)
      ).each {
        nodes.add([type:it.class.simpleName, name:it.displayName, executors:it
.numExecutors])
      }

      //clouds
      def clouds = []
      Jenkins.instance.clouds.each {
        def cloud = [type:it.descriptor.displayName, name:it.displayName]
        try{
          cloud.executorsCap = it.templates?.inject(0, {a, c -
> a + (c.numExecutors * c.instanceCap)})
        }catch(e){}
        try{
          cloud.executorsPerNode = it.numExecutors
        }catch(e){}
        clouds.add(cloud)
      }

      //shared clouds
      Jenkins.instance.getAllItems(com.cloudbees.opscenter.server.model.Shared
Cloud.class).each {
        //TODO may need to check either numExectors or numExecutors * instance
```

```
 caps
       clouds.add([type:it.class.simpleName, name:it.displayName, executorsPe
rNode:it.cloud.numExecutors])
      }

      def host = [type:'$type', name:'$name', url:Jenkins.instance.rootUrl, co
res:Runtime.runtime.availableProcessors(), nodes:nodes, clouds:clouds, offline
:false]

      return new groovy.json.JsonBuilder(host).toString()
    """, listener, "host-script.groovy", [:]));
    host = new groovy.json.JsonSlurper().parseText(stream.toString().minus("Re
sult: "));
  } else {
    host = [type:type, name:name, offline:true]
  }
  return host;
}

return new groovy.json.JsonBuilder(cjoc).toPrettyString()

//=======================================
// ------------------- DisplayFolderCredentials --------------
//Displays only username credentials for global and each top level folders bas
ed upon a condition
import com.cloudbees.plugins.credentials.Credentials
import com.cloudbees.hudson.plugins.folder.AbstractFolder;

String strMaster = java.net.InetAddress.getLocalHost();
println "============================================================================
============================"
println "Executing on ${strMaster}"
println "============================================================================
============================"

println "Top-
Folder#, Credential#, TopFolderName, CredentialID, UserName, Password, Descrip
tion"
println "============================================================================
================"

Set<Credentials> allCredentials = new HashSet<Credentials>()

def creds = com.cloudbees.plugins.credentials.CredentialsProvider.lookupCreden
tials(com.cloudbees.plugins.credentials.Credentials)

allCredentials.addAll(creds)
```

```
/********** GLOBAL **********************/
println "------ GLOBAL (Username with password) --------"
for (c in allCredentials)
{
    if ( c.descriptor.displayName == 'Username with password')
    {
        //if(c.username.startsWith('StlCD'))
            //println "${c.id}, ${c.descriptor.displayName}, ${c.username}, ${c.password}, ${c.description}"
            println "${c.id}, ${c.descriptor.displayName}, ${c.username}, ${c.password}"
    }
}
/********** GLOBAL **********************/

println('')
println('')
println('------ TOP-LEVEL-FOLDERS (Username with password) --------')

int iTopCount = 0, iFolderCount = 0
def topItems = Jenkins.instance.getAllItems(com.cloudbees.hudson.plugins.folder.Folder)
//def topItems = Jenkins.instance.getItems(com.cloudbees.hudson.plugins.folder.Folder)
topItems.each
{
    f ->
    //println "${f}"
    allCredentials.clear()
    //Set<Credentials> folderCredentials = new HashSet<Credentials>()
    creds = com.cloudbees.plugins.credentials.CredentialsProvider.lookupCredentials(com.cloudbees.plugins.credentials.Credentials, f)
    //println (creds);
    allCredentials.addAll(creds)
    ++iTopCount
    iFolderCount = 0
    //displayCreds(f.name, allCredentials, iTopCount, iFolderCount);
    displayCreds(f, allCredentials, iTopCount, iFolderCount);
}

//void displayCreds(String folderName, HashSet<Credentials> folderCredentials, int iTCount, int iFCount)
void displayCreds(def folderName, HashSet<Credentials> folderCredentials, int iTCount, int iFCount)
{
    for (c in folderCredentials)
    {
        if ( c.descriptor.displayName == 'Username with password')
```

```
        {
            ////if(c.username.startsWith('StlCD'))
            if(c.id.endsWith('BB'))
                println "${iTCount}, ${++iFCount}, ${folderName.getFullDisplay
Name()}, ${c.id}, ${c.username}, ${c.password}"
                //println "${iTCount}, ${++iFCount}, ${folderName.getFullDispl
ayName()}, ${c.id}, ${c.username}, ${c.password}, ${c.description}"
        }
    }
}
return null
//============================================
// ---------------- JobsOlderThanThreshold ----------------
#!groovy
//Collects and display list of projects that were built more than X days ago.

import java.text.SimpleDateFormat
def iCount = 0

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"
//-------------------------------------------------

String strMaster = java.net.InetAddress.getLocalHost();
println "================================================================================
================================================"
println "Find builds older than 'X' days to mark as 'Disabled'/'Deleted' or si
mply identify that they are already 'Disabled'"
println ">>>>>>>>> CD Master - ${strMaster} <<<<<<<<<<<<<<<<<<<<<<<<<<<<<"
println "================================================================================
================================================"

import hudson.slaves.*
import java.util.concurrent.*
import org.jenkinsci.plugins.workflow.job.WorkflowJob

import hudson.model.*
import hudson.maven.*
import hudson.tasks.*
import jenkins.model.Jenkins
import hudson.maven.reporters.*
import hudson.plugins.emailext.*

//Sending email from script console
import javax.mail.Session
import javax.mail.Message
import javax.mail.Transport
```

```
import javax.mail.internet.MimeMessage
import javax.mail.internet.InternetAddress

jenkins = Hudson.instance
def now=Calendar.instance;

def Date before30Days
def Date before90Days
def Date currentTime
use( groovy.time.TimeCategory )
{
  currentTime = new Date()
  before30Days = new Date().plus(-30)
  before90Days = new Date().plus(-90)
}

println "Current Time - ${currentTime}, Threshold to disable - ${before30Days}
, Threshold to delete - ${before90Days}";

def list2Disable=[];
def list2Delete=[];
def listAlreadyDisabled=[];

def allItems = Jenkins.instance.getAllItems(WorkflowJob)
def iItemCount = 0;

for (item in allItems)
{
  try
  {
    // Ignore project that contains freeze or patch case insensitive
    if (item.name ==~ /(?i)(freeze|patch).*/)
    {
        //println("\t Ignored as it is a freeze or patch build");
    }
    else if (!item.disabled && item.getLastBuild() != null)
    {
        if (new Date(item.getLastBuild().startTimeInMillis).before(before90Day
s)) //Builds older than 30 days
        {
            list2Delete<< item;
        }
        if (new Date(item.getLastBuild().startTimeInMillis).before(before30Day
s)) //Builds older than 30 days
        {
            list2Disable<< item;
        }
    }
```

```
    else if (item.disabled)
    {
        listAlreadyDisabled<< item;
    }
  }
  catch(Exception e)
  {
      println "Exception raised ... - ${e.message}";
  }
}

String strHTMLReportStart = "<!doctype html><html lang=\"en\"> <body>";
String strHTMLTableStart = "<table style=100% border=\"1\">";
String strHTMLTableEnd = "</table>";
String strHTMLCaptionStart = "<caption><b>";
String strHTMLCaptionEnd = "</b></caption>";
String strHTMLCaption = "";
String strHTMLTableHeader = "<tr><th>S.No.</th><th>Name</th><th>Job#</th>
<th>Build_Time</th><th>Result</th><th>Executed_By</th><th>Build_Folder</th>
</tr>";
String strHTMLReportEnd = "</body></html>"
String strHTMLLine = "<hr style=\"height:2px;border-
width:0;color:gray;background-color:gray\">";
String strHTMLData = "";
String strHTMLBlankRow = "<tr/>"
String strHTMLReport = "";

//------------------ Lets begin with HTML report -----------
strHTMLReport = strHTMLReportStart;
if (list2Delete.size()>0)
{
  strHTMLCaption = "";
  println("Please take a look at following projects which need to be deleted a
s they are over 90 days old!");
  strHTMLCaption = "Builds older than ${before90Days}";
  def i = 0;
  println("\t S.No., Name, Job#, Build_Time, Result, Executed_By, Build_Folder
");
  strHTMLTableRow = "";
  for (item in list2Delete)
  {
      println("\t ${i++}, ${item.name}, ${item.displayName}, ${item.getLastBui
ld().getNumber()}, ${item.getLastBuild().time}, ${item.getLastBuild().getResul
t()}, ${findCause(item.getLastBuild())}, ${item.getLastBuild().getRootDir()}")
;
      strHTMLTableRow += "<tr><td>${i++}</td><td>${item.name}</td>
<td>${item.getLastBuild().getNumber()}</td>
<td>${item.getLastBuild().time}</td>
```

```
<td>${item.getLastBuild().getResult()}</td>
<td>${findCause(item.getLastBuild())}</td>
<td>${item.getLastBuild().getRootDir()}</td></tr>";
  }

  strHTMLReport += strHTMLCaptionStart + strHTMLCaption + strHTMLCaptionEnd +
strHTMLTableStart + strHTMLTableHeader + strHTMLTableRow + strHTMLTableEnd + s
trHTMLBlankRow + strHTMLLine + strHTMLBlankRow;
}

if (list2Disable.size()>0)
{
  strHTMLCaption = "";
  println("Please take a look at following projects which need to be disabled
as they are over 30 days old!");
  strHTMLCaption = "Builds older than ${before30Days}";
  def i = 0;
  //println("\t S.No., Name, Job#, Build_Time, Result, Executed_By");
  println("\t S.No., Name, Job#, Build_Time, Result, Executed_By, Build_Folder
");
  strHTMLTableRow = "";
  for (item in list2Disable)
  {
    //println("\t ${i++}, ${item.name}, ${item.getLastBuild().getNumber()}, ${
item.getLastBuild().time}, ${item.getLastBuild().getResult()}, ${findCause(ite
m.getLastBuild())}");
    println("\t ${i++}, ${item.name}, ${item.getLastBuild().getNumber()}, ${it
em.getLastBuild().time}, ${item.getLastBuild().getResult()}, ${findCause(item.
getLastBuild())}, ${item.getLastBuild().getRootDir()}");
    strHTMLTableRow += "<tr><td>${i++}</td><td>${item.name}</td>
<td>${item.getLastBuild().getNumber()}</td>
<td>${item.getLastBuild().time}</td>
<td>${item.getLastBuild().getResult()}</td>
<td>${findCause(item.getLastBuild())}</td>
<td>${item.getLastBuild().getRootDir()}</td></tr>";
  }

  strHTMLReport += strHTMLCaptionStart + strHTMLCaption + strHTMLCaptionEnd +
strHTMLTableStart + strHTMLTableHeader + strHTMLTableRow + strHTMLTableEnd + s
trHTMLBlankRow + strHTMLLine + strHTMLBlankRow;
}

if (listAlreadyDisabled.size()>0)
{
  strHTMLCaption = "";
  println("Please take a look at following projects which are already disabled
...");
  strHTMLCaption = "Jobs currently marked DISABLED!";
```

```
  def i = 0;
  strHTMLTableRow = "";
  String strHTMLTableDHdr = "<tr><th>S.No.</th><th>Name</th></tr>";
  for (item in list2Disable)
  {
    println("\t ${i++} - ${item.name}");
    strHTMLTableRow += "<tr><td>${i++}</td><td>${item.name}</td></tr>"
    //println("\t ${i++}, ${item.name}, ${item.getLastBuild().getNumber()}, ${
item.getLastBuild().time}, ${item.getLastBuild().getResult()}, ${findCause(cur
rentBuild)}");
  }

  strHTMLReport += strHTMLCaptionStart + strHTMLCaption + strHTMLCaptionEnd +
strHTMLTableStart + strHTMLTableDHdr + strHTMLTableRow + strHTMLTableEnd + str
HTMLBlankRow + strHTMLLine + strHTMLBlankRow;
}

println "Lets send the email notification out..."
try
{
    // String strMessage = "Build counts before ${before30Days} - ${list2Disab
le.size()}    \n\
    // Build counts before ${before90Days} - ${list2Delete.size()}    \n\
    // Jobs counts which are already disabled - ${listAlreadyDisabled.size()}"
    String strHTMLTableCHdr = "<tr><th>Build count before ${before90Days}</th>
</tr>"
    String strHTMLTableCRow = "<tr><td>${list2Delete.size()}</td></tr>";
    String strHTMLCounts = strHTMLCaptionStart + "Builds counts..." + strHTMLC
aptionEnd + strHTMLTableStart + strHTMLTableCHdr + strHTMLTableCRow + strHTMLT
ableEnd + strHTMLBlankRow;
    strHTMLTableCHdr = "<tr><th>Build count before ${before30Days}</th></tr>"
    strHTMLTableCRow = "<tr><td>${list2Disable.size()}</td></tr>";
    strHTMLCounts += strHTMLTableStart + strHTMLTableCHdr + strHTMLTableCRow +
 strHTMLTableEnd + strHTMLBlankRow;
    strHTMLTableCHdr = "<tr><th>Jobs disabled</th></tr>"
    strHTMLTableCRow = "<tr><td>${listAlreadyDisabled.size()}</td></tr>";
    strHTMLCounts += strHTMLTableStart + strHTMLTableCHdr + strHTMLTableCRow +
 strHTMLTableEnd + strHTMLBlankRow;

    strHTMLReport += strHTMLCounts + strHTMLReportEnd;
    //Send final email for the controller
    //sendEmail(Rohit.Bhatnagar@mastercard.com, "Build details for " + strMast
er, strMessage);
    sendEmail(Rohit.Bhatnagar@mastercard.com, "Build details for " + strMaster
, strHTMLReport);
}
catch(Exception e)
{
```

```
  println "Exception raised when sending email... - ${e.message}";
}

//--------------------------------------------------
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"

use(groovy.time.TimeCategory)
{
    def duration = endDate - startDate
    print "Elapsed time: Days: ${duration.days}, Hours: ${duration.hours}, Min
utes: ${duration.minutes}, Seconds: ${duration.seconds}"
}

return;

//------------ https://medium.com/faun/how-to-get-jenkins-build-job-details-
b8c918087030 ---------
def findCause(upStreamBuild) {
    //Check if the build was triggered by SCM change
    scmCause = upStreamBuild.getCause(hudson.triggers.SCMTrigger.SCMTriggerCau
se)
    if (scmCause != null) {
        return scmCause.getShortDescription()
    }//Check if the build was triggered by timer
    timerCause = upStreamBuild.getCause(hudson.triggers.TimerTrigger.TimerTrig
gerCause)
    if (timerCause != null) {
        return timerCause.getShortDescription()
    }//Check if the build was triggered by some jenkins user
    usercause = upStreamBuild.getCause(hudson.model.Cause.UserIdCause.class)
    if (usercause != null) {
        return usercause.getUserId()
    }//Check if the build was triggered by some jenkins project(job)
    upstreamcause = upStreamBuild.getCause(hudson.model.Cause.UpstreamCause.cl
ass)
    if (upstreamcause != null) {
        job = Jenkins.getInstance().getItemByFullName(upstreamcause.getUpstrea
mProject(), hudson.model.Job.class)
        if (job != null) {
            upstream = job.getBuildByNumber(upstreamcause.getUpstreamBuild())
            if (upstream != null) {
                return upstream
            }
        }
    }
    return;
}
```

```
//---------------------- Send email to recipient -------------
def sendEmail(def sTo, def sSubject, def sMsg)
{
    try
    {
        println "${sTo} - ${sSubject} - ${sMsg}";
        def descriptor = Jenkins.instance.getDescriptor("hudson.tasks.Mailer")

        Session session = descriptor.createSession();
        MimeMessage msg = new MimeMessage(session);

        InternetAddress fromAddress = new InternetAddress(descriptor.getAdminA
ddress());

        msg.setFrom(fromAddress)

        //msg.setRecipients(MimeMessage.RecipientType.TO, (InternetAddress[])
[new InternetAddress('Rohit.Bhatnagar@mastercard.com')].toArray());
        msg.setRecipients(MimeMessage.RecipientType.TO, (InternetAddress[]) [n
ew InternetAddress("${sTo}")].toArray());

        String charset = descriptor.getCharset();
        //msg.setSubject("Test", charset);
        msg.setSubject("${sSubject}", charset);
        //msg.setText("Hello from Jenkins!", charset)
        //msg.setText("${sMsg}", charset)
        msg.setText("${sMsg}", "utf-8", "html");
        //msg.setContent("${sMsg}", "text/html") //we should invoke the setCon
tent(Object obj, String type) method of the MimeMessage object. The setContent
() method specifies the mime type of the content explicitly, and for HTML form
at, the type parameter must be "text/html".

        Transport transporter = session.getTransport("smtp");
        transporter.connect();
        transporter.send(msg);
    }
    catch(Exception  e)
    {
        println "Exception raised when preparing email content... - ${e.messag
e}";
    }

    return;
}

//=============================================
// ---------------- LdapCredentials ---------------
```

```
import java.text.SimpleDateFormat

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"
//-------------------------------------------------

String strMaster = java.net.InetAddress.getLocalHost();
println "================================================================="
println "Displays all Credential Stores entries for all top level folders"
println ">>>>> CD Master - ${strMaster} <<<<<<<<<<<<<<<<<<"
println "================================================================="

//Displays all credentials global and inside each top level folder
import com.cloudbees.plugins.credentials.Credentials
import com.cloudbees.hudson.plugins.folder.AbstractFolder;

Set<Credentials> allCredentials = new HashSet<Credentials>()

def creds = com.cloudbees.plugins.credentials.CredentialsProvider.lookupCreden
tials(com.cloudbees.plugins.credentials.Credentials)

allCredentials.addAll(creds)

/***********/
println "------ GLOBAL --------"
println()
for (c in allCredentials)
{
    if(c.descriptor.displayName == 'Secret text')
        println "${strMaster}^Global^${c.id}^${c.descriptor.displayName}^^${c.
description}"
    else if ( c.descriptor.displayName == 'Username with password')
        println "${strMaster}^Global^${c.id}^${c.descriptor.displayName}^${c.u
sername}^${c.description}"
    else if ( c.descriptor.displayName == 'Secret file')
        println "${strMaster}^Global^${c.id}^${c.descriptor.displayName}^${c.f
ileName}^${c.description}"
    else if ( c.descriptor.displayName == 'Certificate')
        println "${strMaster}^Global^${c.id}^${c.descriptor.displayName}^${c.k
eyStore}^${c.description}"
    else if ( c.descriptor.displayName == 'SSH Username with private key')
        println "${strMaster}^Global^${c.id}^${c.descriptor.displayName}^${c.u
sername}^${c.description}"
    else
        println("======= CredentialID - ${c.id}^Scope - ${c.scope}^Descriptor
- ${c.descriptor.displayName}^Type - ${c}")
}
```

```
println()
println("------ Top Folder --------")
println()
def topFolderName="AccessManagement" //CD5 Folder

//def topItems = Jenkins.instance.getItemByFullName(topFolderName^AbstractFold
er)
def topItems = Jenkins.instance.getItems(com.cloudbees.hudson.plugins.folder.F
older)
topItems.each
{
    f ->
    Set<Credentials> folderCredentials = new HashSet<Credentials>()
    folderCredentials = com.cloudbees.plugins.credentials.CredentialsProvider.
lookupCredentials(com.cloudbees.plugins.credentials.Credentials, f)

    for (c in folderCredentials)
    {
    //println("CredentialID - ${c.id}^Scope - ${c.scope}^Descriptor - ${c.desc
riptor.displayName}^Type - ${c}")
    if(c.scope == null)
    {
       if(c.descriptor.displayName == 'Secret text')
        println "${strMaster}^${f.name}^${c.id}^${c.descriptor.displayName}^^$
{c.description}"
       else if ( c.descriptor.displayName == 'Username with password')
        println "${strMaster}^${f.name}^${c.id}^${c.descriptor.displayName}^${
c.username}^${c.description}"
       else if ( c.descriptor.displayName == 'Secret file')
        println "${strMaster}^${f.name}^${c.id}^${c.descriptor.displayName}^${
c.fileName}^${c.description}"
       else if ( c.descriptor.displayName == 'Certificate')
        println "${strMaster}^${f.name}^${c.id}^${c.descriptor.displayName}^${
c.keyStore}^${c.description}"
       else if ( c.descriptor.displayName == 'SSH Username with private key')
        println "${strMaster}^${f.name}^${c.id}^${c.descriptor.displayName}^${
c.username}^${c.description}"
       else
        println("======= ${f.name}^CredentialID - ${c.id}^Scope - ${c.scope}^D
escriptor - ${c.descriptor.displayName}^Type - ${c}")
    }
    }
}
println()

//--------------------------------------------------
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"
```

```
use(groovy.time.TimeCategory)
{
    def duration = endDate - startDate
      print "Elapsed time: Days: ${duration.days}^Hours: ${duration.hours}^Min
utes: ${duration.minutes}^Seconds: ${duration.seconds}"
}

return null;

//==============================
// ------------- New Jobs in Last Hour ------------------
#!groovy

//Author - Rohit K. Bhatnagar
//Purpose - Collects fresh builds executed in last HR on all Masters connected
 to the CJOC
import java.text.SimpleDateFormat
import groovy.time.TimeCategory

def iItemCount = 1
def iJobCount = 1
def iTotalBuilds = 0
def iTotalItems = 0
def iIsBuilding = 0
def iOlderThan60Mins = 0
def iNoBuilds = 0 //These should fall under NULLPointerExceptions
def iLastHrBuilds = 0 //Build counts in last hr only

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"

//-------------------------------------------------------------------------
-----
//Collects build status of all new jobs against a top level item in the last 1
 Hour
//-------------------------------------------------------------------------
-----

def jenkins = Jenkins.getInstance()
def topItems = jenkins.getItems()
topItems.each
{
  topI ->
  iTotalItems++ //Counts total top level items
  def allItemJobs = topI.getAllJobs()
  iJobCount = 1
```

```
  def String strPrint = ""
  allItemJobs.each
  {
    itemJob ->
    try
    {
      iTotalBuilds++ //Counts total builds in entire master
      if(itemJob.building)
        iIsBuilding++;
      def build_time = itemJob.getFirstBuild().startTimeInMillis;
      //println "${build_time} - ${new Date(build_time)}"
      def Date before60Minutes
      use( groovy.time.TimeCategory )
      {
        before60Minutes = new Date() - 60.minutes
      }
      //println "${before60Minutes}"
      def buildB4Time = new Date(build_time).before(before60Minutes);

      if(!buildB4Time)
      {
        iLastHrBuilds++
        //println "${iLastHrBuilds} - ${itemJob.getName()}"
        def firstBuild = itemJob.getFirstBuild()
        def String jobResult =  itemJob.getFirstBuild().result;
        def buildTime = itemJob.getFirstBuild().getTime()
        def buildDuration = itemJob.getFirstBuild().getDuration()
        def strBuildDuration = itemJob.getFirstBuild().durationString
        strPrint = "DETAILS | ${firstBuild} | Time | ${buildTime} | Result | $
{jobResult} | Duration | ${strBuildDuration} | Duration(ms) | ${buildDuration}
"
        println "FRESH ITEMS | ${iItemCount++} | ${topI.getName()} | ${iJobCou
nt++} | ${itemJob.getName()} | ${strPrint}"
        System.out.println "FRESH ITEMS | ${iItemCount} | ${topI.getName()} |
${iJobCount} | ${itemJob.getName()} | ${strPrint}"
      }
      else
        iOlderThan60Mins++
    }
    catch(NullPointerException npexp)
    {
      iNoBuilds++
        //println "${iItemCount++} | ${topI.getName()} | ${iJobCount++} | ${it
emJob.getName()} | NULL-POINTER EXCEPTION | ${npexp.message}"
    }
    catch(Exception exp)
    {
      println "${iItemCount++} | ${topI.getName()} | ${iJobCount++} | ${itemJo
```

```
b.getName()} | GENERIC EXCEPTION | ${exp.message}"
    }
  }
}

println "FRESH ITEMS TOTALS | TOTAL ITEMS | ${iTotalItems} | TOTAL BUILDS | ${
iTotalBuilds} | OLD BUILDS | ${iOlderThan60Mins} | BUILDING | ${iIsBuilding} |
 NO BUILDS | ${iNoBuilds} | LAST HR BUILDS | ${iLastHrBuilds} | START | ${star
tDate}"

//---------------------------------------------------------------------------
def endDate = new Date()
def formatEndDate = sdf.format(endDate)
println "End time: ${formatEndDate}"

System.out.println "END | ${formatEndDate} | GRAND FRESH ITEMS TOTALS | TOTAL
ITEMS | ${iTotalItems} | TOTAL BUILDS | ${iTotalBuilds} | OLD BUILDS | ${iOlde
rThan60Mins} | BUILDING | ${iIsBuilding} | NO BUILDS | ${iNoBuilds} | LAST HR
BUILDS | ${iLastHrBuilds} | START | ${startDate}"

return null
//==========================
// ----------------- TopAllItemsCounts ---------------
#!groovy
import java.text.SimpleDateFormat
import com.cloudbees.hudson.plugins.folder.*
import com.cloudbees.hudson.plugins.modeling.impl.folder.FolderTemplate
import org.jenkinsci.plugins.workflow.job.*;
import org.jenkinsci.plugins.workflow.multibranch.*;
import com.cloudbees.hudson.plugins.modeling.impl.jobTemplate.*;
import hudson.maven.MavenModuleSet;
import hudson.model.FreeStyleProject;
import hudson.maven.MavenModule;
import jenkins.branch.OrganizationFolder;
import hudson.matrix.*; //MatrixConfiguration; //hudson.matrix.MatrixProject
import hudson.ivy.IvyModuleSet;
import com.cloudbees.hudson.plugins.modeling.impl.auxiliary.AuxModel;
import com.cloudbees.hudson.plugins.modeling.impl.builder.BuilderTemplate;
import com.cloudbees.hudson.plugins.modeling.impl.publisher.PublisherTemplate;

def iTotalItems = 0;
def iItemCount = 0;
def iCount = 0
def iFolderCount = 0
def iJobCount = 0 //WorkflowJob
def iMultiBranchProject = 0 //WorkflowMultiBranchProject
def iMavenModuleSet = 0;
def iMavenModule = 0;
```

```
def iFreeStyle = 0 //hudson.model.FreeStyleProject
def iJobTemplate = 0;
def iOrgFolder = 0;
def iMatrixConf = 0;
def iMatrixProj = 0;
def iFolderTemp = 0;
def iIvyModuleSet = 0
def iAuxModel = 0
def iBuilderTemplate = 0
def iPublisherTemplate = 0

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"

//---------------------------------------------------------------------
//Displays all item counts for all top level items on the Jenkins Instance
//---------------------------------------------------------------------

//Master HostName and IP Address
println "Master executed on - ${java.net.InetAddress.getLocalHost()}"

println "---------------------------------------------------------------------
---"
println "Displays all item counts for all top level items on the Jenkins Insta
nce"
println "---------------------------------------------------------------------
---"

def jenkins = Jenkins.getInstance()
//Get Workspaces for all immediate items
def topItems = jenkins.getItems()

println "*************************************************************"
println "-------- Item Counts ---------------------------------------"
println "*************************************************************"
println "Top#, Name, Total Items, Folder, Workflow-Job, Maven-
ModuleSet, MultiBranch, FreeStyle, JobTemplate, MavenModule, Oragnization-
Folder, Matrix-Configuration, Matrix-Project, Folder-
Template, AuxModel, IvyModuleSet, BuilderTemplate, PublisherTemplate, UnKnown"
println "=====================================================================
========================================="
topItems.eachWithIndex
{
  topName, topCount ->
  def allItemJobs = topName.getAllItems()
  iItemCount = 0
  iCount = 0
```

```
iFolderCount = 0
iJobCount = 0 //WorkflowJob
iMultiBranchProject = 0 //WorkflowMultiBranchProject
iMavenModuleSet = 0;
iMavenModule = 0;
iFreeStyle = 0 //hudson.model.FreeStyleProject
iJobTemplate = 0;
iOrgFolder = 0;
iMatrixConf = 0;
iMatrixProj = 0;
iFolderTemp = 0;
iIvyModuleSet = 0;
iAuxModel = 0;
iBuilderTemplate = 0;
iPublisherTemplate = 0;

allItemJobs.eachWithIndex
{
  itemJob, itemCount ->
  try
  {
  iTotalItems++;
      iItemCount++
  if(itemJob instanceof Folder)
      iFolderCount++;
  else if(itemJob instanceof WorkflowJob)
      iJobCount++
  else if(itemJob instanceof MavenModuleSet)
      iMavenModuleSet++;
  else if(itemJob instanceof WorkflowMultiBranchProject)
      iMultiBranchProject++
  else if(itemJob instanceof FreeStyleProject)
      iFreeStyle++
  else if(itemJob instanceof JobTemplate)
      iJobTemplate++
  else if(itemJob instanceof MavenModule)
      iMavenModule++
  else if (itemJob instanceof OrganizationFolder)
      iOrgFolder++
  else if (itemJob instanceof MatrixConfiguration)
      iMatrixConf++
  else if (itemJob instanceof MatrixProject)
      iMatrixProj++
  else if (itemJob instanceof FolderTemplate)
      iFolderTemp++
  else if (itemJob instanceof IvyModuleSet)
      iIvyModuleSet++
  else if (itemJob instanceof AuxModel)
```

```
        iAuxModel++
    else if (itemJob instanceof BuilderTemplate)
        iBuilderTemplate++
    else if (itemJob instanceof PublisherTemplate)
        iPublisherTemplate++
    else
    {
        iCount++;
        println "\t\t${topCount} | ${itemCount} | Item-
Name : ${itemJob.getName()} | Class-Name : ${itemJob.getClass()}"
    }
    }
    catch(Exception exp)
    {
        println "Exception - ${exp.message}"
    }
  }

  println "${topCount}, ${topName.getName()}, ${iItemCount}, ${iFolderCount},
${iJobCount}, ${iMavenModuleSet}, ${iMultiBranchProject}, ${iFreeStyle}, ${iJo
bTemplate}, ${iMavenModule}, ${iOrgFolder}, ${iMatrixConf}, ${iMatrixProj}, ${
iFolderTemp}, ${iAuxModel}, ${iIvyModuleSet}, ${iBuilderTemplate}, ${iPublishe
rTemplate}, ${iCount}";

  //println "${topCount} | ${topName.getName()} | ${iItemCount} | ${iFolderCou
nt} | ${iJobCount} | ${iMavenModuleSet} | ${iMultiBranchProject} | ${iFreeStyl
e} | ${iJobTemplate} | ${iMavenModule} | ${iOrgFolder} | ${iMatrixConf} | ${iM
atrixProj} | ${iFolderTemp} | ${iAuxModel} | ${iIvyModuleSet} | ${iBuilderTemp
late} | ${iPublisherTemplate} | ${iCount}";

  //println "${topCount} | Top-Folder-Name : ${topName.getName()} | Item-
Count : ${iItemCount} | Folder-Count | ${iFolderCount} | Workflow-Job-
Count | ${iJobCount} | Maven-ModuleSet-
Count | ${iMavenModuleSet} | MultiBranch-
Count | ${iMultiBranchProject} | FreeStyle-
Count | ${iFreeStyle} | JobTemplate-Count | ${iJobTemplate} | MavenModule-
Count | ${iMavenModule} | Oragnization-Folder-Count | ${iOrgFolder} | Matrix-
Configuration-Count | ${iMatrixConf} | Matrix-Project-
Count | ${iMatrixProj} | Folder-Template-Count | ${iFolderTemp} | AuxModel-
Count | ${iAuxModel} | IvyModuleSet-
Count | ${iIvyModuleSet} | BuilderTemplate-
Count | ${iBuilderTemplate} | PublisherTemplate-
Count | ${iPublisherTemplate} | UnKnown | ${iCount}";
}

println "_____
_____"
println "Total items count on ${java.net.InetAddress.getLocalHost()} - ${iTota
```

```
lItems}"

//-------------------------------------------------------------------------
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"

use(groovy.time.TimeCategory)
{
    def duration = endDate - startDate
    print "Elapsed time: Days: ${duration.days}, Hours: ${duration.hours}, Min
utes: ${duration.minutes}, Seconds: ${duration.seconds}"
}
//=================================
// --------------- TotalItemsAndCounts ------------------
#!groovy - Rohit K. Bhatnagar
import java.text.SimpleDateFormat
def iCount = 0
def iItemCount = 0
def iFolderCount = 0
def iJobCount = 0 //WorkflowJob
def iMultiBranchProject = 0 //WorkflowMultiBranchProject
def iMavenModuleSet = 0;
def iMavenModule = 0;
def iFreeStyle = 0 //hudson.model.FreeStyleProject
def iJobTemplate = 0;
def iOrgFolder = 0;
def iMatrixConf = 0;
def iMatrixProj = 0;
def iFolderTemp = 0;
def iIvyModuleSet = 0
def iAuxModel = 0
def iBuilderTemplate = 0
def iPublisherTemplate = 0

def sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss.SSS")
def startDate = new Date()
println "Start time : ${sdf.format(startDate)}"

//-------------------------------------------------------------------------
//Displays total counts for various items defined in the Jenkins Instance
//-------------------------------------------------------------------------

//Master HostName and IP Address
println "Master executed on - ${java.net.InetAddress.getLocalHost()}"

println "-------------------------------------------------------------------------
--"
println "Displays total counts for various items defined in the Jenkins Instan
```

```
ce"
println "-------------------------------------------------------------------
--"

import com.cloudbees.hudson.plugins.folder.*
import com.cloudbees.hudson.plugins.modeling.impl.folder.FolderTemplate
import org.jenkinsci.plugins.workflow.job.*;
import org.jenkinsci.plugins.workflow.multibranch.*;
import com.cloudbees.hudson.plugins.modeling.impl.jobTemplate.*;
import hudson.maven.MavenModuleSet;
import hudson.model.FreeStyleProject;
import hudson.maven.MavenModule;
import jenkins.branch.OrganizationFolder;
import hudson.matrix.*; //MatrixConfiguration; //hudson.matrix.MatrixProject
import hudson.ivy.IvyModuleSet;
import com.cloudbees.hudson.plugins.modeling.impl.auxiliary.AuxModel;
import com.cloudbees.hudson.plugins.modeling.impl.builder.BuilderTemplate;
import com.cloudbees.hudson.plugins.modeling.impl.publisher.PublisherTemplate;
```

```
def jenkins = Jenkins.getInstance()
//Get Workspaces for all immediate items
//def topItems = jenkins.getItems(com.cloudbees.hudson.plugins.folder.Folder.c
lass)
//def topItems = jenkins.getAllItems(com.cloudbees.hudson.plugins.folder.Folde
r.class)
def topItems = jenkins.getAllItems()
topItems.each
{
  topI ->
  try
  {
    iItemCount++;
      if(topI instanceof Folder)
    {
      iFolderCount++;
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()} | Columns : ${topI.getColumns()} | Items : ${topI.ge
tItems()} | Jobs : ${topI.getAllJobs()}"
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()} | Items : ${topI.getItems()} | Jobs : ${topI.getAllJ
obs()}"
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
      else if(topI instanceof WorkflowJob)
    {
          iJobCount++
```

```
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if(topI instanceof MavenModuleSet)
    {
      iMavenModuleSet++;
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if(topI instanceof WorkflowMultiBranchProject)
    {
      iMultiBranchProject++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if(topI instanceof FreeStyleProject)
    {
      iFreeStyle++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if(topI instanceof JobTemplate)
    {
      iJobTemplate++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if(topI instanceof MavenModule)
    {
      iMavenModule++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof OrganizationFolder)
    {
      iOrgFolder++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof MatrixConfiguration)
    {
      iMatrixConf++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof MatrixProject)
    {
      iMatrixProj++
```

```
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof FolderTemplate)
    {
      iFolderTemp++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof IvyModuleSet)
    {
      iIvyModuleSet++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof AuxModel)
    {
      iAuxModel++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof BuilderTemplate)
    {
      iBuilderTemplate++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else if (topI instanceof PublisherTemplate)
    {
      iPublisherTemplate++
      //println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
    else
    {
      iCount++;
      println "${iItemCount} | Item-Name : ${topI.getName()} | Class-
Name : ${topI.getClass()}"
    }
  }
  catch(Exception exp)
  {
    println "Exception - ${exp.message}"
  }
  //println "${iItemCount} | Item-Name : ${topI.getName()} | Folder-
Count : ${iFolderCount} | Job-Count : ${iJobCount} | UnKnown : ${iCount}"
}
```

```
println "Total-Items-Count | ${iItemCount} | Folder-
Count | ${iFolderCount} | Workflow-Job-Count | ${iJobCount} | Maven-ModuleSet-
Count | ${iMavenModuleSet} | MultiBranch-
Count | ${iMultiBranchProject} | FreeStyle-
Count | ${iFreeStyle} | JobTemplate-Count | ${iJobTemplate} | MavenModule-
Count | ${iMavenModule} | Oragnization-Folder-Count | ${iOrgFolder} | Matrix-
Configuration-Count | ${iMatrixConf} | Matrix-Project-
Count | ${iMatrixProj} | Folder-Template-Count | ${iFolderTemp} | AuxModel-
Count | ${iAuxModel} | IvyModuleSet-
Count | ${iIvyModuleSet} | BuilderTemplate-
Count | ${iBuilderTemplate} | PublisherTemplate-
Count | ${iPublisherTemplate} | UnKnown | ${iCount}"

println "========="

println "Total-Items-Count | ${iItemCount}"
println "Folder-Count | ${iFolderCount}"
println "Workflow-Job-Count | ${iJobCount}"
println "Maven-ModuleSet-Count | ${iMavenModuleSet}"
println "MultiBranch-Count | ${iMultiBranchProject}"
println "FreeStyle-Count | ${iFreeStyle}"
println "JobTemplate-Count | ${iJobTemplate}"
println "MavenModule-Count | ${iMavenModule}"
println "Oragnization-Folder-Count | ${iOrgFolder}"
println "Matrix-Configuration-Count | ${iMatrixConf}"
println "Matrix-Project-Count | ${iMatrixProj}"
println "Folder-Template-Count | ${iFolderTemp}"
println "AuxModel-Count | ${iAuxModel}"
println "IvyModuleSet-Count | ${iIvyModuleSet}"
println "BuilderTemplate-Count | ${iBuilderTemplate}"
println "PublisherTemplate-Count | ${iPublisherTemplate}"
println "UnKnown | ${iCount}"
```

```
//-------------------------------------------------------------------------
def endDate = new Date()
println "End time: ${sdf.format(endDate)}"

use(groovy.time.TimeCategory)
{
    def duration = endDate - startDate
    print "Elapsed time: Days: ${duration.days}, Hours: ${duration.hours}, Min
utes: ${duration.minutes}, Seconds: ${duration.seconds}"
}

return null
//=========================
```

Rohit K Bhatnagar
Director
Software Development Engineering

Mastercard
2200 Mastercard Boulevard | 211-12E3
O'Fallon, MO 63368+7263 | DL - JT@mastercard.com
tel +1 (636) 722-6833 | mobile +1 (636) 515-6336