



# **XMC 8.5 Workshop**

## Workflow Manager introduction

Markus Nikulski  
Sr. Corporate System Engineer

October 2020

# Python Script vs Workflow

## single Python Script

- all in one script
  - easy searching
  - easy local data sharing
- Script exist as file
- Script output exist as file

## Workflow

- chain multiple small Python scripts
- data exchange between scripts blocks
- e-mail support
- event / syslog support
- HTTP(s) GET/PUT support
- Shell support (BASH)
- Dashboard
- Advanced execution (NBI/Alarm/NAC)
- requires XMC advance license

<https://emc.extremenetworks.com/>

[https://emc.extremenetworks.com/content/oneview/docs/tasks/docs/c\\_workflows.html](https://emc.extremenetworks.com/content/oneview/docs/tasks/docs/c_workflows.html)



# Workflow

The screenshot displays the Workflow Dashboard interface. The main workspace shows a workflow diagram for "/Workflows/Examples/User Input". The workflow starts with a "Start" node, followed by an "init" activity, a diamond gateway, an "Activity Group" containing "up" and "down" activities, and an "End" node. A "Signal - 31" event triggers a parallel gateway that splits into two paths: one leading to a "good" activity (highlighted with a dashed green border) and another to a "bad" activity. Both paths merge at another parallel gateway, which then leads to a final "End" node.

On the left, the "Workflow Dashboard" sidebar includes a "User Workflows" list with a "Group" callout, a "Workflow" callout, and a "parallel conditional" callout. The "Palette" contains various activities and gateways, with callouts for "parallel", "Timer", and "Event Syslog".

On the right, the "Details" panel shows the configuration for the selected activity. The "Script Source" section has "Embed Script" selected. The "Script Configuration" section shows "Script Type" as "Python" and "Script Content" as "print 'good'". A "Parameters context dependent" callout points to the "Script Configuration" section. The "Execution Settings" section has "Terminate workflow on failure" checked.

At the bottom, the status bar shows "[ NetSight Administrator/root ] Last Updated: 2019/05/15 18:18:30 Uptime: 0 Days 00:33:20".



# create a simple Workflow

Workflow Dashboard | Scheduled Tasks | Saved Tasks | Scripts | **Workflows** | Search | Help | Menu

User Workflows -

- Workflows
  - Markus
    - test

No data to display

System Workflows +

Settings | Refresh

Details

**General**

Name:  
test

Description:

Save

---

# Workflow Variables

---

# Workflow variables

**build-in variables**  
devices / ports  
trigger user dialog

**Scope**  
Workflow = global  
Activity = per script block

**Type**  
is always a string!  
Has to be casted

Name	Default Value	Variable Ref...	Scope	Type	Referenced
workflowTimeout			Workflow	Number	false
GlobalStatus	init		Workflow	String	false
JSONDATA			Workflow	Json	false
Direction			Workflow	String	true
Direction_INFO	NO_IDEA		Workflow	String	false
devices			Workflow	Json	true
ports			Workflow	Json	true
scriptAssignm...			Activity	Activity	true
scriptName			Activity	String	true
scriptType			Activity	String	true
scriptContent			Activity	String	true
status			Activity	String	true
output			Activity	String	true
failFast			Activity	Boolean	true

```
timeout = int( emc_vars['workflowTimeout'] )
```



# Activity variables

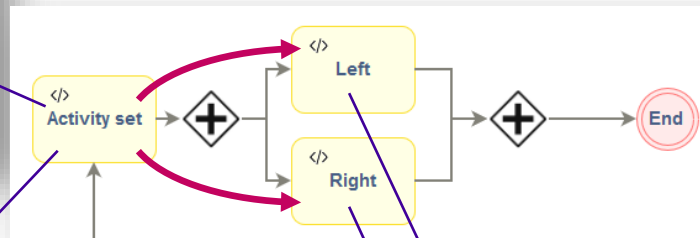
Details

General Variables Inputs Outputs

Name: Activity set  
ID: 4  
Custom ID: ID4

Display Name	Variable Reference
Status	status
Output	output
Left	Left
Right	Right

```
Variables  
Direction=DOWN  
Direction_INFO=NO_IDEA  
GlobalStatus=start  
ID4_Left=1.2.3.4  
ID4_Right=4.3.2.1  
ID4_status=SUCCESS  
JSONDATA=  
LeftStatus=1.2.3.4-LEFT
```



```
emc_results.put("Left", '1.2.3.4')  
emc_results.put("Right", '4.3.2.1')
```

```
print "Direction = " + emc_vars["ID4_Left"]
```

```
print "Direction = " + emc_vars["ID4_Right"]
```

Make them unique per script block, but this is more complex to use



# Workflow variables

Details

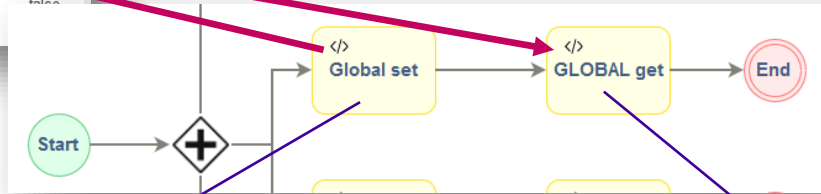
General **Variables** Inputs Outputs Menus Network OS

Add Edit Delete Global Variables...

Name	Default ...	Variabl...	Scope	Type	Referen
workflowTimeout			Workflow	Number	false
GlobalStatus	init		Workflow	String	false
JSONDATA			Workflow	String	false
Direction			Workflow	String	true
Direction_INFO	NO_IDEA		Workflow	String	false

Variables

```
Direction=DOWN
Direction_INFO=NO_IDEA
GlobalStatus=start
JSONDATA=
USE_IPV6=true
date=05/17/2019 09:59:53 AM
devices=
domain=
failFast=false
hostName=192.168.162.1
javaScript=
```



```
print "STATUS = " + emc_vars['GlobalStatus']
emc_results.put("GlobalStatus", 'start')
```

```
print "STATUS = " + emc_vars['GlobalStatus']
```

More easy to use, but can be in conflict with multiple read/write at the same time





# Workflow variables that are use to carry data structures

Details

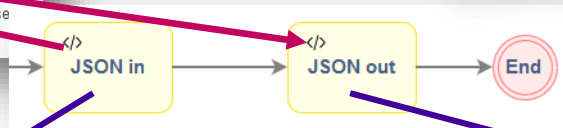
General Variables Inputs Outputs Menus Network OS

+ Add Edit Delete Global Variables...

Name	Default ...	Variabl...	Scope	Type	Referen
workflowTimeout			Workflow	Number	false
GlobalStatus	init		Workflow	String	false
JSONDATA			Workflow	Json	false
Direction			Workflow	String	true
Direction_INFO	NO_IDEA		Workflow	String	false

Variables

```
Direction=DOWN
Direction_INFO=NO_IDEA
GlobalStatus=init
JSONDATA={"Weight": 81, "Heigh": 163, "List": ["first", "second"], "Age": 49}
USE_IPV6=true
date=05/17/2019 09:59:53 AM
devices=
domain=
```



```
import json

data = ( {'Age':49, 'Heigh':163, 'Weight':81} )

data['List'] = ['first', 'second']

emc_results.put('JSONDATA', json.dumps( data ) )
```

```
import json

data = json.loads( emc_vars['JSONDATA'] )

print 'The age: ' + str( data['Age'] )

data['List'].append('new')
```

# Workflow variables that are use to carry data structures



serialization



string

deserialization



```
data['User'] = {}  
data['User']['Age'] = 30  
data['User']['Height'] = 185
```

```
print data['User']['Height']
```

```
import json  
text = json.dumps( data )
```

```
fh = open( 'data.json', 'w')  
fh.write( text )  
fh.close()
```

```
{"User": { "Age": 30, "Height": 193 } }
```

```
{  
  User: {  
    Age: 30,  
    Height: 193  
  }  
}
```

```
fh = open( 'data.json', 'r')  
text = fh.read()  
fh.close()
```

```
import json  
data = json.loads( text )
```

```
print data['User']['Height']
```



# Workflow variables used for user input

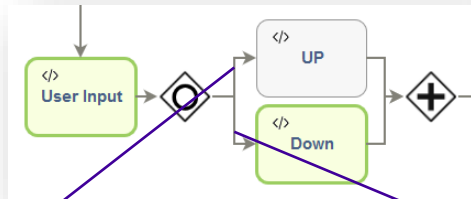
The image shows a workflow editor interface with a 'Details' pane on the right. The 'Inputs' tab is selected, showing a 'Manage Inputs...' button. A 'Run Workflow - Data handover' dialog is open, displaying 'Workflow Inputs' with a 'Timeout Properties' section (Timeout: 4 hr(s)) and a 'Custom Inputs' section. The 'Custom Inputs' section has a prompt 'What is your direction:' and a dropdown menu with 'UP' and 'DOWN' options. A 'Manage Inputs' dialog is also open, showing a table of workflow inputs.

Display Name	Display Type	Valid Values	Variable Referenc...	Required	Prompt User
Timeout	Timer		workflowTimeout	true	false
What is your dire...	ComboBox	UP,DOWN	Direction	true	true

emc\_vars['Direction']



# Workflow variables used conditional parallel gateways



```
Variables
Direction=DOWN
Direction_INFO=all good
GlobalStatus=init
JSONDATA=
USE_IPV6=true
date=05/17/2019 09:59:53 AM
devices=
domain=
```

Details

General **Condition**

Configuration

Expression Type:  
Evaluate Variables

Variable:  
Direction

Operator:  
Equals to

Value:  
UP

Details

General **Condition**

Configuration

Expression Type:  
Evaluate Variables

Variable:  
Direction

Operator:  
Equals to

Value:  
DOWN



# Workflow variables used for e-mail & event/syslog messages

## E-Mail

The screenshot shows the configuration for an E-Mail workflow. The main canvas displays a workflow diagram with an 'Event' trigger, an 'E-Mail' action, and an 'End' node. The 'Details' panel on the right is set to the 'Inputs' tab and contains the following configuration:

- General**: Manage Inputs...
- Email Configuration**
  - To: user@example.com
  - Email List: SupportTeam
  - Subject: User input is `${Direction_INFO}`
  - Body: The DIRECTION is `${Direction}` or Status is `${Direction_INFO}`
- Execution Settings**
  - Terminate workflow on failure.

## Event / Syslog

The screenshot shows the configuration for an Event / Syslog workflow. The main canvas displays a workflow diagram with an 'Event' trigger, an 'E-Mail' action, and an 'End' node. The 'Details' panel on the right is set to the 'Inputs' tab and contains the following configuration:

- General**: Manage Inputs...
- Signal Configuration**
  - Signal Type: Event
  - Event Title: DATA Handover
  - Event Category: Workflows
  - Event Source: User input `${Direction_INFO}`
  - Event Subcomponent: The DIRECTION is `${Direction}`
  - Event Severity: INFO
  - Event Message: The DIRECTION is `${Direction}`, user input is `${Direction_INFO}`



---

# Variables - Message

---

# workflowMessage, activityMessage, deviceMessage

- `emc_results.put("workflowMessage", "Some custom workflow message")`
- `emc_results.put("activityMessage", "Some custom activity message")`
- `emc_results.put("deviceMessage", "Some custom device message")`

Workflow Dashboard Scheduled Tasks Saved Tasks Scripts Workflows **Message Test workflow (12743)** ✕

Summary

Status	Start Date/Time	Name	Version	Source	# Devices	Started By	End Date/Time	Message	Path
✓	4/10/2020 10:02:52 AM	Message Test workflow	2	Workflow Designer [z...	2	zpala	4/10/2020 10:02:53 AM	<u>Some custom workflow message</u>	/Workflows/Zdenek/Message Test workflow

Graph View **Table View**


Show Output Show Variables

Status	Name	Custom ID	Activity Type	Start Date/Time	End Date/Time	Message
SUCCESS	Script - 4	ID4	Script	4/10/2020 10:02:52 AM	4/10/2020 10:02:53 AM	<u>Some custom activity message</u>


Devices Grid

Show Output


Status	Device IP	Output Path	Start Date/Time	End Date/Time	Message
SUCCESS	10.253.0.16		4/10/2020 10:02...	4/10/2020 10:02...	Some custom device message
SUCCESS	<u>10.253.0.9</u>		4/10/2020 10:02...	4/10/2020 10:02...	<u>Some custom device message</u>



ACTIVE  
0



COMPLETED  
12,642



FAILED  
103

Show Details Live Historical

Status	Start Date/Time ↓	Name	Version	Source	# Devices	Started By	End Date/Time	Message	Path
✓	4/10/2020 10:03:24 AM	RFC3580 for ISW thr...	19	Notification Action [R...	1	NetSight Server	4/10/2020 10:03:26 AM	Configured VLAN: 220 on IP 20.0.202.12: on P...	/Workflows/Zdenek/RFC3580 for ISW through SNMP
✓	4/10/2020 10:03:13 AM	RFC3580 for ISW thr...	19	Notification Action [R...	1	NetSight Server	4/10/2020 10:03:15 AM	Configured VLAN: 1 on IP 20.0.202.12: on Port:...	/Workflows/Zdenek/RFC3580 for ISW through SNMP
✓	4/10/2020 10:02:52 AM	Message Test workflow	2	Workflow Designer [z...	2	zpala	4/10/2020 10:02:53 AM	<u>Some custom workflow message</u>	/Workflows/Zdenek/Message Test workflow
✓	4/10/2020 9:52:56 AM	RFC3580 for ISW thr...	19	Notification Action [R...	1	NetSight Server	4/10/2020 9:52:58 AM	Configured VLAN: 220 on IP 20.0.202.12: on P...	/Workflows/Zdenek/RFC3580 for ISW through SNMP



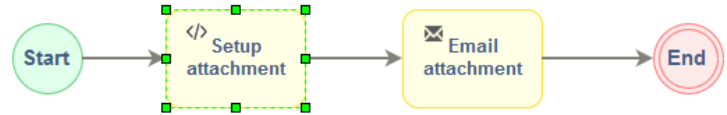
---

E-Mail with attachment

---



# Workflow e-mail with attachment (part I)



```
import os

server_log = os.path.join(emc_vars['jboss.server.log.dir'], 'server.log')

server_log = str(os.path.abspath( server_log ))

files = [server_log]

if os.path.exists(server_log):
    email_list = ",".join(files)
    print "emailAttachments:" + email_list
    emc_results.put("emailAttachments", email_list)
else:
    raise Exception("server.log not found: "+server_log)
```

- ← catch all file names
- ← get absolute path
- ← convert it to a list
- ← chain list to string
- ← assign to a external variable

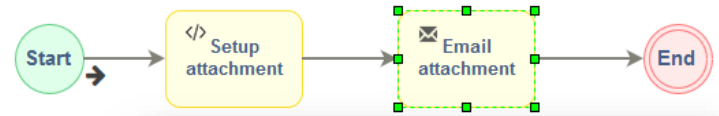
```
Output

Script Name: Email Attachments_Setup_attachment
Date and Time: 2019-11-08T07:15:12.778
XMC User: mnikulski
XMC User Domain:
IP:
emailAttachments: /usr/local/Extreme_Networks/NetSight/appdata/logs/server.log
```

Variable Reference	status
output	output
Email Attachments	emailAttachments



# Workflow e-mail with attachment (part II)



is the e-mail HTML content (not the attachment)

This is a simple workflow that demonstrates how to send an email with file attachments. For this example, the server.log was used.

<h2>How it works</h2>

The <b>Email Activity</b> supports a variable called <b>emailAttachments</b>.

This variable MUST be a string containing one or more comma-separated file paths.

For this to work, create a new activity/workflow variable called emailAttachments and add it to the "Email Activity". The variable scope can be workflow or activity. If the scope is "Activity" then it will need to be assigned to a variable from a previous activity using the "Variable Reference" column.

<br>

<br>

Sample python snippet:

<br>

<pre>

```
server_log = os.path.join(emc_vars['jboss.server.log.dir'], 'server.log')
files = [str(server_log)]
# create a comma separated list of file paths
email_list = ",".join(files)
emc_results.put("emailAttachments",email_list)
```

</pre>

The screenshot shows the configuration interface for the 'Email attachment' activity. The 'Email Configuration' section is expanded, showing the following fields:

- To:**
- Email List:**
- Subject:**
- Body:**   
<h2>How it works</h2>

The 'Variables' tab is selected, showing a table of variables:

Name	Default Value	Variable...	Scope	Type
workflow...			Workflow	String
workflow...			Workflow	String
emailUser	helpdesk@reading...		Workflow	String
scriptAss...			Activity	St...

A yellow arrow points from the 'Body' field in the configuration to the 'Sample python snippet' in the text above. A purple arrow points from the 'emailUser' variable in the table to the 'To:' field in the configuration.



# Workflow e-mail with attachment (result)

The screenshot shows a web-based email interface for a Helpdesk system. The top header displays the date and time: TUESDAY NOVEMBER 12, 2019, 02:19. The interface is divided into three main sections: a left sidebar for navigation, a central inbox list, and a right pane for the selected email.

**Left Sidebar:** Shows the user profile 'Helpdesk' with email 'helpdesk@reading.ctc.local'. Below are navigation options: Inbox, Drafts, Sent, Trash, and Junk.

**Inbox List:** A list of 11 messages. The top message is selected and highlighted in green:

From	Subject	Date	Size
xmc@reading.ctc.local	Workflow Email with attachments	Friday 02:19	1.1 MiB
gim@reading.ctc.local	GIM Test Email	13-Jun-19	1.7 KiB
xmc@reading.ctc.local	Here is the end-system history for engine...	09-May-19	132 KiB
xmc@reading.ctc.local	Here is the end-system history for Engin...	07-May-19	34.1 KiB
xmc@reading.ctc.local	Alarm: High CPU on EXOS - CTC demo Jan2...	29-Jan-19	114 KiB
xmc@reading.ctc.local	Alarm: PoC T-systems - High CPU utilizati...	29-Jan-19	2.4 KiB
xmc@reading.ctc.local	Alarm: PoC T-systems - High CPU utilizati...	11-Jan-19	54.7 KiB
xmc@reading.ctc.local	Automated Tech Support gathered for 10...	11-Jan-19	223 KiB
xmc@reading.ctc.local	NetSight Unknown Trigger Any	07-Jan-19	1.8 KiB
Helpdesk	"Personal Calendar" has been created	07-Jan-19	0 KiB
XMC		07-Jan-19	

**Email Content (Right Pane):** The selected email is titled 'Workflow Email with attachments' and is dated 'Friday, November 08, 2019 02:19 EST'. The sender is 'xmc@reading.ctc.local' and the recipient is 'helpdesk@reading.ctc.local'. The body text reads: 'This is a simple workflow that demonstrates how to send an email with file attachments. For this example, the server.log was used.' Below this is a section titled 'How it works' which explains the 'Email Activity' and its configuration. A 'Sample python snippet:' is provided, showing code that reads a file named 'server.log' and attaches it to an email. At the bottom of the email content, there is a preview of the 'server.log' attachment, showing it is 1.1 MiB in size. A yellow arrow points to this attachment preview.

```
server_log = os.path.join(emc_vars['jboss.server.log.dir'], 'server.log')
files = [str(server_log)]
# create a comma separated list of file paths
email_list = ",".join(files)
emc_results.put("emailAttachments", email_list)
```



---

# Workflow execution

---

# Workflow execution option

1. manual from the editor
2. device(s) or/and port context menu
3. scheduler
4. Alarm action
5. NAC notification
6. NBI call

Script

Workflow



# Workflow execution via Alarm

Details

< bles Inputs Outputs **Menus** Net >

Authorization Groups (Roles):  
NetSight Administrator

Category:  
System

Menus:  
Alarm

Groups:  
Select Groups... Rem

Edit Status Change Alarm Definition: Device Down

Severity: Critical

Enabled:

Criteria Actions Other Options

+ Add Edit... Remove Test

Actions

Run Task [Workflow - reconnect AP]
------------------------------------

Alarm Suppression

Enable Alarm Action Limit

Max Count: 5

Reset Interval: 1 Days

Save Cancel

```
if not emc_vars['family'] == 'ExtremeWireless WiNG':  
    return  
  
if 'alarmName' in emc_vars:  
    if not emc_vars['alarmName'] == 'Device Down':  
        return  
    else:  
        # do something
```



# Workflow execution via NAC

The screenshot displays the NetSight NAC configuration interface. The main view shows the configuration for a workflow named "Printer port set long timeout".

**Main Configuration:**

- Name:** Printer port set long timeout
- Notes:** change HP switch NAC timeout to long to keep printer in suspend mode onboard
- Type:** End-System
- Trigger:** State Change
- Conditions:** End-System Group: Printers
- Actions:** Access Control Events Workflow: NAC HP long timer(/Workflows/Customer use cases)
- Result:** State Change End-System matches Printers: Run Access Control Events workflow NAC HP long timer(/Workflows/Customer use cases) on the XMC server

**Details Modal:**

- Authorization Groups (Roles):** NetSight Administrator
- Category:** System
- Menus:** Access Control Events
- Groups:** Select Groups... Remove All Groups

The interface includes a sidebar with navigation options (Dashboard, Policy, Access Control, End-Systems, Reports) and a bottom status bar showing the user as [ NetSight Administrator/root ] and system uptime.



# Workflow execution via NAC

```
def validateStartCondition():  
  
    if( emc_vars["authType"].startswith("AUTH_MAC")           and  
        emc_vars["memberOfGroups"] == "Printer"              and  
        emc_vars["switchIP"]      == emc_vars["oldswitchIP"] and  
        emc_vars["switchPortId"]   == emc_vars["oldswitchPortId"] and  
        emc_vars["macAddress"]     == emc_vars["oldmacAddress"]  
    ):   
  
        return True  
    else:  
        return False
```

```
if validateStartCondition():  
  
    if emc_vars["state"] == "ACCEPT":  
        changeTimer('LONG')  
  
    elif emc_vars["state"] == "DISCONNECTED":  
        changeTimer('DEFAULT')
```





# Workflow query via NBI

```
GraphiQL [Prettify] [History] < Docs
```

```
1 {
2   workflows {
3     allWorkflows (path: "/Workflows/Examples/") {
4       name
5       category
6       path
7     }
8   }
9 }
```

```
{
  "data": {
    "workflows": {
      "allWorkflows": [
        {
          "name": "Data handover",
          "category": "System",
          "path": "/Workflows/Examples/Data handover"
        },
        {
          "name": "reconnect AP",
          "category": "System",
          "path": "/Workflows/Examples/reconnect AP"
        },
        {
          "name": "single Workflow",
          "category": "System",
          "path": "/Workflows/Examples/single Workflow"
        },
        {
          "name": "Update Device Notes - Group Membership",
          "category": "System",
          "path": "/Workflows/Examples/Update Device Notes - Group Membership"
        }
      ]
    }
  }
}
```

```
{
  workflows {
    allWorkflows (path: "/Workflows/Examples/") {
      name
      category
      path
    }
  }
}
```

```
{
  workflows {
    allWorkflows {
      name
      category
      path
    }
  }
}
```



# Workflow **execute** via NBI

with 8.3 is no direct data return exists

```
mutation {
  workflows {
    startWorkflow(
      input: {
        path: "/Workflows/Examples/Data handover",
        variables: {Direction: "DOWN"}
      } {
        status
        message
        executionId
      }
    }
  }
}
```

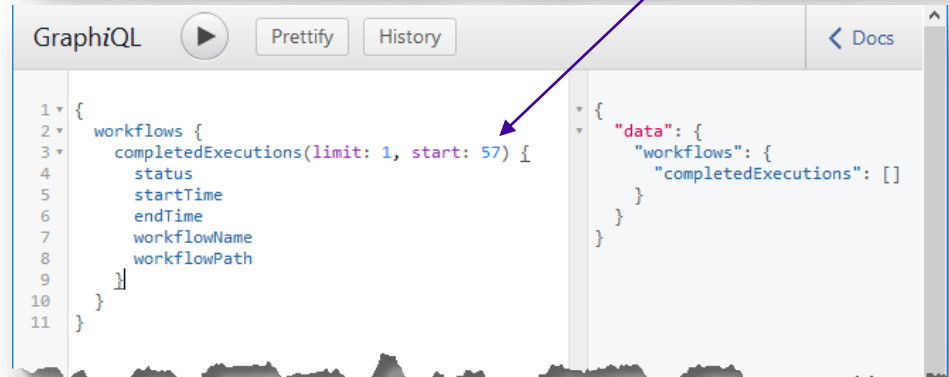
```
{
  workflows {
    completedExecutions(limit: 1, start: 57) {
      status
      startTime
      endTime
      workflowName
      workflowPath
    }
  }
}
```



The screenshot shows a GraphQL IDE interface with a query on the left and its JSON response on the right. The query is a mutation to start a workflow. The response shows the workflow was started successfully with a message and an execution ID of 57.

```
1 mutation {
2   workflows {
3     startWorkflow(
4       input: {
5         path: "/Workflows/Examples/Data handover",
6         variables: {Direction: "DOWN"}
7       } {
8         status
9         message
10        executionId
11      }
12    }
13  }
```

```
{
  "data": {
    "workflows": {
      "startWorkflow": {
        "status": "SUCCESS",
        "message": "",
        "executionId": 57
      }
    }
  }
}
```



The screenshot shows a GraphQL IDE interface with a query on the left and its JSON response on the right. The query asks for completed workflow executions starting from ID 57. The response shows an empty array for completed executions.

```
1 {
2   workflows {
3     completedExecutions(limit: 1, start: 57) {
4       status
5       startTime
6       endTime
7       workflowName
8       workflowPath
9     }
10  }
11 }
```

```
{
  "data": {
    "workflows": {
      "completedExecutions": []
    }
  }
}
```

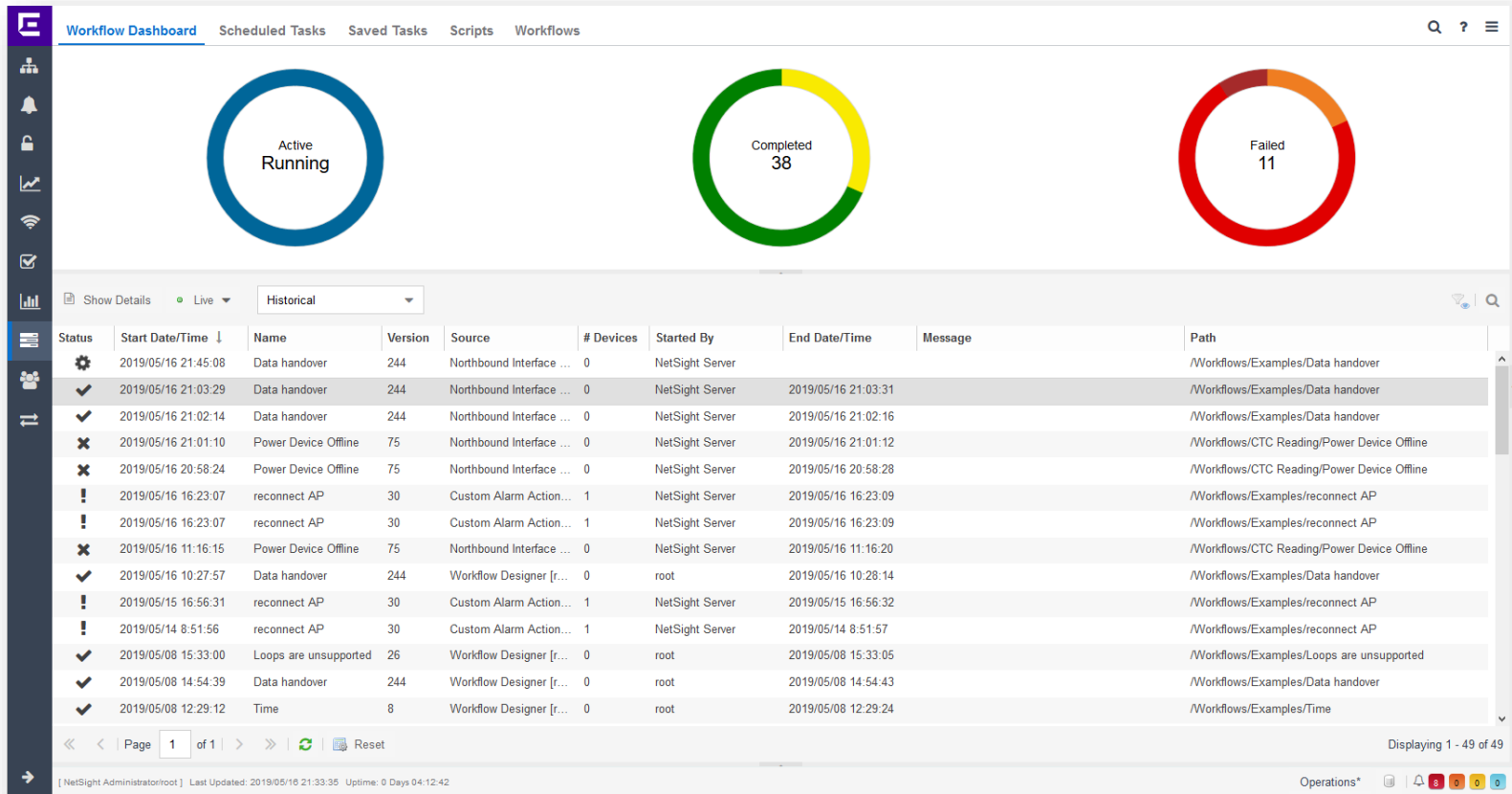


---

# Workflow Dashboard

---

# Workflow Dashboard



# Workflow Dashboard

The screenshot displays the Workflow Dashboard for a workflow named "Data handover (50)". The interface includes a navigation menu on the left, a top navigation bar with tabs for "Workflow Dashboard", "Scheduled Tasks", "Saved Tasks", "Scripts", "Workflows", and "Data handover (50)".

**Summary Table:**

Status	Start Date/Time	Name	Version	Source	# Devices	Started By	End Date/Time	Message	Path
✓	2019/05/16 21:45:08	Data handover	244	Northbound Interface ...	0	NetSight Server	2019/05/16 21:45:11		/Workflows/Examples/Data handover

**Graph View:** The workflow is shown in graph view, starting with a "Start" node, followed by a "User Input" node, a "Down" node (script icon), and an "Event" node (warning triangle icon), leading to an "End" node.

**Output Window:**

```
Script Name: Data handover_Global_set
Date and Time: 2019-05-16T21:45:08.806
XMC User: NetSight Server
XMC User Domain:
IP:
STATUS = init
```

**Devices Grid:** A table showing the execution status for devices.

Status	Device IP	Output Path	Start Date/Time	End Date/Time	Message
SUCCESS			2019/05/16 21:4...	2019/05/16 21:4...	

**Variables Window:**

```
Direction=DOWN
Direction_INFO=NO_IDEA
GlobalStatus=start
JSONDATA=
USE_IPV6=true
date=05/16/2019 09:45:07 PM
devices=
domain=
failFast=false
javax.script.engine=python
javax.script.engine_version=2.7.0
javax.script.language=python
javax.script.name=Python
jboss.bind.address=192.168.162.50
jboss.bind.address.management=127.0.0.1
jboss.http.port=8080
jboss.https.port=8443
jboss.server.log.dir=../appdata/logs
output=
scriptAssignment=import
scriptName=
scriptType=Python
serverHTTPSPort=8443
serverIP=192.168.162.50
..
```



---

## Workflow remarks

---

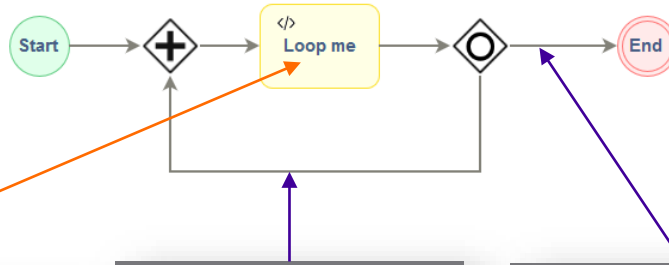
# Workflow loops (unsupported)

Details

General Variables Inputs Output

Add Edit Delete Global

Name ↑	Default V...	Variable ...	Scope
count			Workflow
sleep_timer	1		Workflow
start_count	3		Workflow
workflowC...			Workflow
workflowC...			Workflow



Edit Script

```
1 import time
2
3 if emc_vars["count"]:
4     in_count = int( emc_vars["count"] )
5 else:
6     in_count = int( emc_vars["start_count"] )
7
8 out_count = in_count - 1
9
10 print "Start count %s" % emc_vars["start_count"]
11 print "change count from %s to %s" % (str(in_count),str(out_count))
12
13 time.sleep( int( emc_vars["sleep_timer"] ) )
14 emc_results.put("count", str(out_count))
15
```

Save Cancel

Details

General Condition

Configuration

Expression Type: Evaluate Variables

Variable: count

Operator: Greater than

Value: 0

Details

General Condition

Configuration

Expression Type: Evaluate Variables

Variable: count

Operator: Equals to

Value: 0

Details

General Variables Inputs Output

Manage Inputs...

Timeout Properties

Timeout: 4 hr(s)

Infinity loops ends when the workflow timeout is reached



# Workflow loops (unsupported)

The screenshot shows the OneView workflow editor interface. The main canvas displays a workflow graph with the following components:

- A green circle labeled "Start".
- A diamond-shaped connector with a plus sign (+).
- A green rectangular task box labeled "Loop me" with a code icon (</>).
- A diamond-shaped connector with a circle inside (O).
- A red circle labeled "End".

The flow starts at "Start", goes to the plus connector, then to the "Loop me" task, then to the O connector, and finally to "End". A feedback arrow loops from the O connector back to the plus connector, indicating a loop structure. A blue box highlights the "Loop me" task and the O connector.

On the right side, there is a "Devices Grid" table:

Status	Device IP	Output Pat
SUCCESS		

Below the Devices Grid, there are buttons for "Show Output" and "Show Variables".

Variables

```
ID4_status=RUNNING
USE_IPV6=true
count=0
date=05/17/2019 08:43:27 AM
devices=
domain=
failFast=true
hostName=192.168.162.1
javax.script.engine=python
javax.script.engine_version=2.7.0
javax.script.language=python
javax.script.name=Python
jboss.bind.address=192.168.162.50
jboss.bind.address.management=127.0.0.1
jboss.http.port=8080
```

Output

```
Script Name: Loops are unsupported_Loop_me
Date and Time: 2019-05-17T08:43:32.353
XMC User: root
XMC User Domain:
IP:
Start count 3
change count from 1 to 0
```

Please be aware, that you will see only the last iteration result!





# Next Presentation

Use the [following link](#) to advance to the next PDF in the Workflow education presentation.





[WWW.EXTREMENETWORKS.COM](http://WWW.EXTREMENETWORKS.COM)

