NOKIA

NSP
Network Services Platform
Release 19.3

Workflow Manager Application Help

3HE-15179-AAAA-TQZZA

Issue 1

March 2019

Legal notice

Nokia is a registered trademark of Nokia Corporation. Other products and company names mentioned herein may be trademarks or tradenames of their respective owners.

The information presented is subject to change without notice. No responsibility is assumed for inaccuracies contained herein.

© 2019 Nokia.

Contents

1 Workflow Manager **5**
1.1 What is Workflow Manager? **5**
1.2 Using the WFM **10**
1.3 How do I use workflows? **13**
1.4 How do I use actions? **14**

1 Workflow Manager

1.1 What is Workflow Manager?

1.1.1

The WFM allows you to create and execute workflows using Mistral's v2 DSL YAML. You can create automated procedures and closed loop automation using Nokia NSP or 3rd party APIs.

Some example use cases for the WFM:

- Node software upgrades
- Service activation tests
- Service fulfillment invoking pre and post deployment workflows
- Customizable policy logic for other applications
- Mass migration of services from one tunnel type to another
- One place scheduling for all NSP REST APIs and node CLI
- Centralized orchestration for Nokia NSP and 3rd party vendor APIs

The WFM uses Mistral DSL v2 which is based on YAML. YAML defines expressions in workflow and action definitions. The formatting of the YAML must comply with Mistral DSL v2 specifications. For more information about Mistral DSL v2, the workflow hierarchy and associated attributes, refer to the Mistral DSL v2 specification online.

As part of workflow creation and execution, you will interact with these objects:

- Workflows
- Input
- Output
- Tasks
- Actions

1.1.2 Workflows

A workflow represents the process to do a job, for example, configure a service. Workflow attributes are described in detail in the Mistral DSL v2 specification. The following is a sample of a workflow:

Figure 1-1 Sample workflow

```
version: '2.0'



validateWFMAPI:
  description: 'Validate WFM APIs'
  type: direct
```

```
  input:
    - token_auth: 'Token_info'
    - rest_gateway_host: '198.51.100.0'
    - wfm_port: '8543'
    - apis:
        - '/v1/workflow'
        - '/v1/workflow?limit=1'

  output:
    status: <% $.status %>


  tasks:
    validateAllApis:
      wait-before: 30
      with-items: api in <% $.apis%>
      action: std.http
      input:
        url: 'https://<% $.rest_gateway_host %>:<% $.wfm_port%>/wfm/api<%
$.api %>'
        method: GET
        verify: False
        headers:
          Content-Type: 'application/json'
          Authorization: <% $.token_auth%>
      publish:
        data: <% task(valdiateAllApis).result.content.response %>
      on-complete:
        - validateWFAllAPI
        - validateWFLimitAPI
      on-error:
        - FailedApiCall

    validateWFAllAPI:
      publish:
        length1: <% $.data[0].totalRows %>
      on-complete:
        - successDone: <% $.length1 > 2 %>
        - FailedValidation: <% $.length1 < 2 %>
        - branchSuccess

    validateWFLimitAPI:
      publish:
        length2: <% $.data[1].totalRows %>
      on-complete:
        - successDone: <% $.length2 = 1 %>
        - FailedValidation: <% $.length2 != 1 %>
        - pause
```

```
      - branchSuccess

  branchSuccess:
    join: all
    publish:
      status: "branched after pause"

  successDone:
    publish:
      status: "DONE"

  FailedValidation:
    publish:
      status: "Validation Failed"

  FailedApiCall:
    publish:
      status: "Error on Api <% $.api %>"
```

1.1.3 Input

Input is the input parameters of a workflow or task, if required. The value of each parameter is a JSON-compliant type (number, string, etc), a dictionary, or a list. It can also be an expression to retrieve a value from a task context, or any of the mentioned types containing inline expressions (for example, string “<% $.movie_name %> is a cool movie!”).

Example In the sample workflow, the token authorization credentials, REST gateway host, WFM port, and APIs are inputs for the workflow. The `validateAllApis` task uses the host, port, and API information to create a URL as an inline expression.

1.1.4 Output

Output is a data structure containing expressions that define workflow output. The output attribute is optional.

Example The output of the sample workflow is the status of the validation performed by the workflow. The expression that defines the output is created by the `publish` statements in the last four tasks of the workflow. Depending on the path performed, one of these tasks will be executed and the status will be defined.

1.1.5 Tasks

A task defines a specific logical step in a workflow. The actions performed by the task can take input data as needed, and produce output data, as defined by the action.

A single task is a logical step that can perform an action, for example, performing a REST request using std.http or CLI request using std.ssh. A task can call other workflows. A task can also be iterated over a list of items performing bulk operations.

Example The `validateAllApis` task in the sample workflow uses the std.http action to perform a REST request.

Tasks can also be chained together based on the following criteria. Tasks that are not chained are executed at the same time.

1. on-success (runs only if the task ran successfully)

   - task 1

   - task 2

2. on-error (runs only if the task failed)

   - task 3

   - task 4

   Example In the sample workflow, the `FailedApiCall` task only runs if `validateAllApis` fails.

3. on-complete (always runs)

   - task 5

   - task 6

   - task 7

   Example In the sample workflow, the `validateWFAllAPI` task runs a set of tasks on completion.

On-success and on-error chaining run subsequent tasks at the same time. On-complete chaining runs subsequent tasks in sequence.

Example In the sample workflow, the `validateWFAllAPI` task runs the `branchSuccess` task after the conditional `successDone` and `FailedValidation` tasks have completed.

Based on the type of action used in a task, a task requires mandatory inputs as defined by the action. For example, the std.http action requires the following inputs:

```
someTask:
        action: std.http
        input:
          url: 'https://<% $.rest_gateway_host %>:<% $.nsd_port%>/ports/'

        method: GET
        verify: False
        headers:
          Content-Type: 'application/json'
          Authorization: 'Bearer <% $.token %>'
        publish:
          portResult: <% task().result.content.response.data %>
```

A list of common attributes is available in the Mistral DSL v2 specification. Mistral allows task policies, conditional branching, and evaluating expression using Jinja2 or YAQL.

Example In the`validateWFAllAPI` task in the sample workflow, the `successDone` and `FailedValidation` tasks are conditional: only one of these tasks will run depending on the value of the length variable.

1.1.6 Actions

An action has a name and input parameters, and outputs a result. A task can execute an action. It can also use the result output by an action. WFM is packaged with a set of system actions such as std.http and std.ssh. System actions cannot be deleted or modified.

You can create ad-hoc actions from system actions. An ad-hoc action is created as a wrapper around a system action, allowing you to set default values and modify the desired output of the action. Ad-hoc actions cannot be used as base actions for another ad-hoc action.

Example of a system action

The system action std.email sends an email message via SMTP. Required parameters are the addresses of recipients and sender, and the SMTP server credentials. Optional parameters are the subject and body text of the email.

When this action is included in a task, the parameters must be provided as part of the task.

Example In the sample workflow, the `validateAllApis` task executes the system action `std.http` to access an IP address using HTTP.

Example of an ad-hoc action

The YAML for an ad-hoc action includes the base system action, the parameters for performing the action, and any input and output you need for the ad-hoc action.

For example, the standard email action requires multiple parameters. Its use can be simplified by creating an ad-hoc action. The following ad-hoc action, error_email, is wrapped around std.email action to provide the std.email input parameters. The only input required by the ad-hoc action is an execution id.

```
version: '2.0'


error_email:
  input:
   - execution_id
  base:std.email
  base-input:
   to_addrs: [admin@mywebsite.org]
   subject: 'Something went wrong with your workflow :('
   body: |
      Please take a look at Dashboard to find out what's wrong with your
workflow execution
       <% $.execution_id %>.
       -- Sincerely, NSP Team
    from_addr:'nsp@nokia.com'
   smtp_server: 'smtp.nokia.com'
   smtp_password: 'SECRET'
```

The ad-hoc action can be incorporated into a workflow:

```
nsp_workflow:
```

```
tasks:
  ...
  send_error_email:
    action: error_email execution id=<% execution().id %>
```

1.2 Using the WFM

The main dashboard shows an overview of all workflow executions. You can view the status of all running workflow executions, waiting on input action executions, failed workflow executions, and paused workflow executions, and drill down to the respective filtered lists based on the execution status. You can also view the upcoming scheduled workflows, most recently failed executions, and recent workflows.

From the main menu, you can navigate to the workflows view or drill down to other components in the workflow hierarchy.

Each view provides a list of the selected objects. From a view, you can search for, create, edit, or delete an object. Point to a list item to see a menu of object-specific options at the right of the page.

From the Dashboard menu, you can navigate to the following views:

- Workflows
  The Workflows view shows the global list of workflows created or imported to the WFM.
- Workflow Executions
  A workflow execution is an instance of an executed workflow. The Workflow Executions view shows the global list of workflow executions in the WFM.
- Schedules
  The Schedules view shows the global list of active schedules in the WFM.
- Action Executions
  The Action Executions view shows the global list of executions of actions performed by workflows. Only executions in a running or failed state are shown.
- Actions
  The Actions view lists both system and ad-hoc actions.

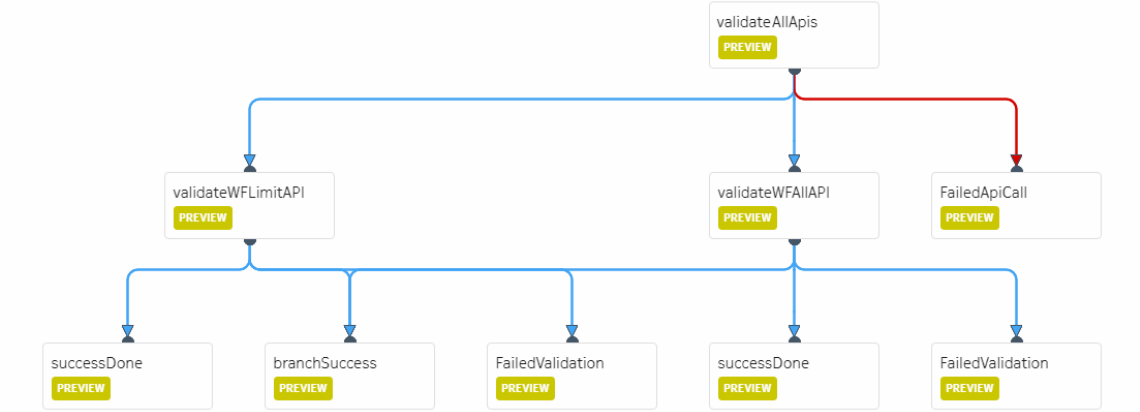
1.2.1 Workflows view

The Workflows view displays a list of the workflows that have been created or imported to the WFM. Double-click on a workflow to display the Info page for the workflow. In the left Status panel, you can view the status of running, failed, paused or success workflow executions.

You can use the Info menu to navigate to views for the selected workflow: Executions, Definition, Schedules, and Flow.

- Executions - the filtered list of executions for the workflow. You can create an execution of the workflow from this view.
- Definition - show the YAML definition
- Schedules - the filtered list of configured schedules for the workflow. You can create a schedule for the workflow from this view.
- Flow - view a graphical representation of the workflow

Example The following is a flow view of the sample workflow.



1.2.2 Workflow Executions view

When a workflow is executed, an execution is created with a unique execution ID. The execution is added to the list on the Workflow Executions view. The Workflow Executions view provides a global list of workflows that have been executed using the WFM, including their inputs, outputs and state. You can create an execution of any workflow from this view.

The following operations are supported for a workflow execution:

- Modify State - you can modify the state of a running execution to paused, error, or cancelled

  Note: The workflow task that is currently running is not cancelled and runs to completion. No new tasks for the workflow are scheduled.
- Resume - you can resume a paused execution
- Re-run - you can start a new workflow execution with the same inputs. The execution you selected re-run from is not affected. For example, if you re-run a paused execution, the execution will remain paused and a new execution with the same inputs will be created and started.
- Delete
- Update

Double-click on an execution to display the Info page for the execution. In the left Status panel, you can view the status of running, waiting, failed, paused, or success tasks.

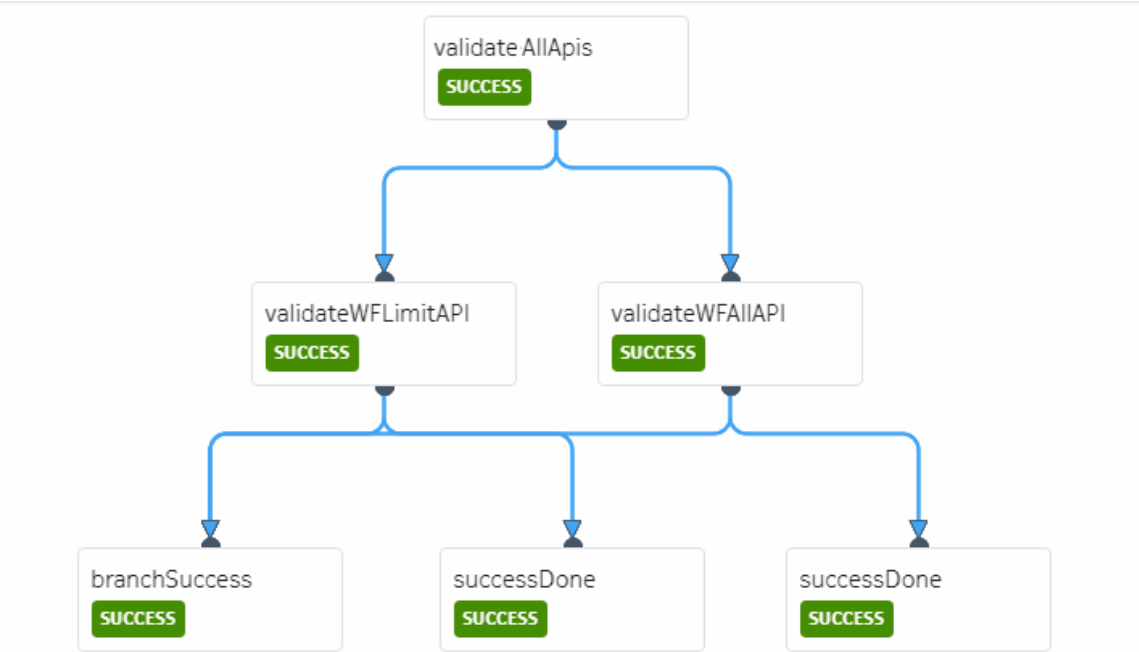
You can use the Info menu to navigate to views for the execution: Tasks, Definition, Input/Output, and Flow.

- Tasks- the filtered list of tasks performed when the execution ran.

  Example The `FailedApiCall` task in the sample workflow only executes if `validateAllApis` fails, therefore if the `validateAllApis` task succeeded, `FailedApiCall` will not appear in the Tasks view.
- Definition - show the YAML code that was executed, including input parameters entered.
- Input/Output - show the input provided when the execution was created, and the output generated.

- Flow - view a graphical representation of the execution, including success or failure of workflow tasks. The flow view shows the path that was performed, not the full workflow.

  Example The following is a flow view of an execution of the sample workflow.



1.2.3 Schedules view

Select Details for a schedule to view the input and parameters for the scheduled executions. You can create a schedule for any workflow from this view.

1.2.4 Action Executions view

As a workflow proceeds, actions are executed to perform the workflow tasks. The Action Executions view shows the action executions that are currently in a running or failed state.

You can create or modify an action execution. You can perform an invoke callback operation to push the status of the action execution to running or failed.

The action executions view only shows action executions that are initiated by workflows. If you create an action execution, a dialog box will pop up with the results. Creating an action execution can be useful for testing ad-hoc actions.

1.2.5 Actions view

The Actions view shows the actions saved to the WFM application. Both system actions and ad-hoc actions are listed. You can create, edit, or delete an ad-hoc action.

1.3 How do I use workflows?

1.3.1

The deployment of a workflow is successful only if the validation is successful. If the validation fails, the validation failure message provides information to facilitate a correct workflow.

A workflow can be imported or exported. You can import and export workflows as text files. An imported workflow must be validated before it can be saved to the WFM.

Create or modify a workflow

1

From the WFM application, choose Workflows from the main menu. A list of existing workflows appears.

2

Click CREATE WORKFLOW or choose an existing workflow and click Update. The Create|Update Workflow window and YAML editor open.

3

Specify a name and description for the workflow.

4

Define the workflow tasks and click VALIDATE & UPDATE FLOW to check your workflow.

5

Click Flow to view a graphical representation of the workflow.

6

Click CREATE|UPDATE to save your workflow.

The workflow appears in the list.

Create or modify a workflow execution

7

Choose one of the following options:

a. To create a workflow execution, select an existing workflow and click Execute. The CREATE EXECUTION window opens.

b. To modify a workflow execution, open an existing workflow and choose Info→Executions. A list of executions appears. Select an execution and from the More menu, choose Update. The Update Execution window opens.

8

Provide the required information and inputs.

You can select Kafka Notification to receive events for the workflow execution states and action execution states. An event is triggered when there is a state change on execution or actions.

9

Click CREATE|UPDATE to start|update your workflow execution.

Create a schedule

10

From the WFM application, select an existing workflow and click the Schedule icon at the right of the page. The Create Schedule window opens.

11

In the CONFIGURATION panel, provide a schedule name and input.

12

Click SCHEDULE and choose the schedule details for the workflow.

13

Click SCHEDULE to save your schedule.

End of steps

1.4 How do I use actions?

1.4.1

An action can only be deployed if it validates.

Create or modify an action

1

From the WFM application, choose Actions from the main menu. A list of existing actions appears.

2

Click CREATE ACTION or select an existing action and click Update. The Create|Update Action window and YAML editor open.

3

Define the action and click Validate to check your work.

4

Click CREATE|UPDATE to save your work.

Create or modify an action execution

5

From the WFM application, choose Action Executions from the main menu. A list of existing action executions appears.

6

Click CREATE ACTION EXECUTION or select an existing action execution and click Update. The Create|Update Action Execution window and YAML editor open.

7

Provide the required information and inputs.

8

Click CREATE|UPDATE to save your work.

An action execution list item is only created if an action is executed by a workflow. When you create an action execution, the results appear in a pop-up message but the action execution is not added to the list. You can create an action execution to test if an action will validate.

End of steps