FUTURE
AT HEART

Automation Manual

Manual Version 1.0.2

NTT DATA

        - Publish to Kafka:
        - For each loop:
- Expression language for transformation
- Examples
    - Example (A):
        - Configuration
    - Example (B)

NTT DATA

What is Symphony Automation?

Automation was built to orchestrate and automate internal an external task, with the simplicity and facility to create flows, the quick use provide the user the quickness and ease to create robust orchestration and automation processes.

Advantages:

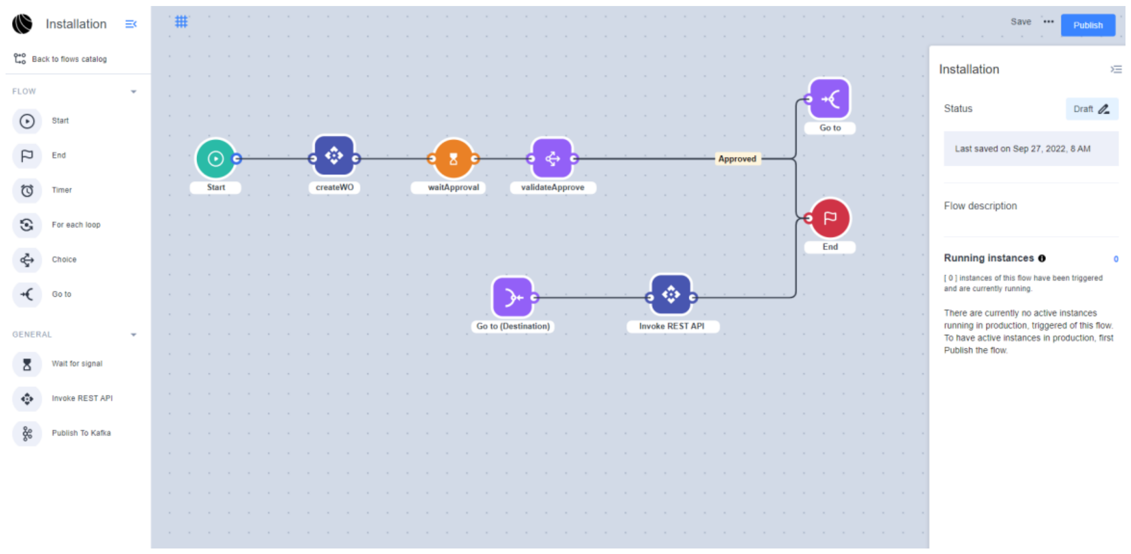
- It allows a person to define a flow with almost no knowledge of code.
- Therefore, the definition times of the automatisms are reduced.
- Help on issues of testing the flows.

High Level Challenges:

- Import and export flow templates.
- Consume kafka topics.

Objective:

Symphony's Automation Module, which is in process of definition and design to be included in the platform, will allow the network operator to automate and orchestrate internal and external tasks, interacting with the rest of the platform. This is a key need of the operators, so building a capable, flexible and complete automation tool will bring a great value for the platform.



Installation

Status: Draft

Last saved on Sep 27, 2022, 8 AM

Flow description

Running instances 0

[0] instances of this flow have been triggered and are currently running.

There are currently no active instances running in production, triggered of this flow. To have active instances in production, first Publish the flow.

Entities:

- **Flow** :

  A published flow that can be triggered by its trigger blocks or start by human interaction or API. It contains a list of blocks. It can also contain a draft flow that is newer version of the flow

- **Flow Draft**:

  Similar to flow but used when flow is still during work and inactive. Another key difference is that many validations that apply on flow graph in Flow don't apply on Flow Draft

- **Flow Template**:

  Copy of the the Flow that is attached to the running flow instance and used for the instance. This makes sure that new versions of Flow don't interrupt running instances

- **Flow Instance**:

  Running execution of the flow, holds a a list of block instance (currently running and already completed) and can be used to check the status of the running flow

- **Block**:

  Block is the execution unit in the flow. It can be inside Flow, Flow Draft or Flow Template. Blocks can be of 3 main categories: Administrative blocks (Start, End, GoTo, Decision, Fork), action blocks and trigger blocks. action blocks and trigger blocks types are set by enum values that corresponds to their implementation in code

Automation Blocks:

Start Block

Initialize the flows, this block marks the starting point of the flow.

End Block

Marks the end of the execution of an automation.

Timer Block

This block allows you to set a waiting time in the execution of the flow.

Choice Block

The choice block receives an input and depending on that determines a path of execution of the flow.

Go To Block

This block allow the user to direct the step to the respective step

Wait for signal Block

This block receives an input that will be a signal that at the moment will be tied to some behavior of Symphony, where upon receiving said signal it proceeds or executes some change according to the configuration.

Invoke Rest Api Block

This block receives URL to perform some request GET, POST, DELETE or PATCH, with its result it would work in some order of the flow.

Publish to kafka Block

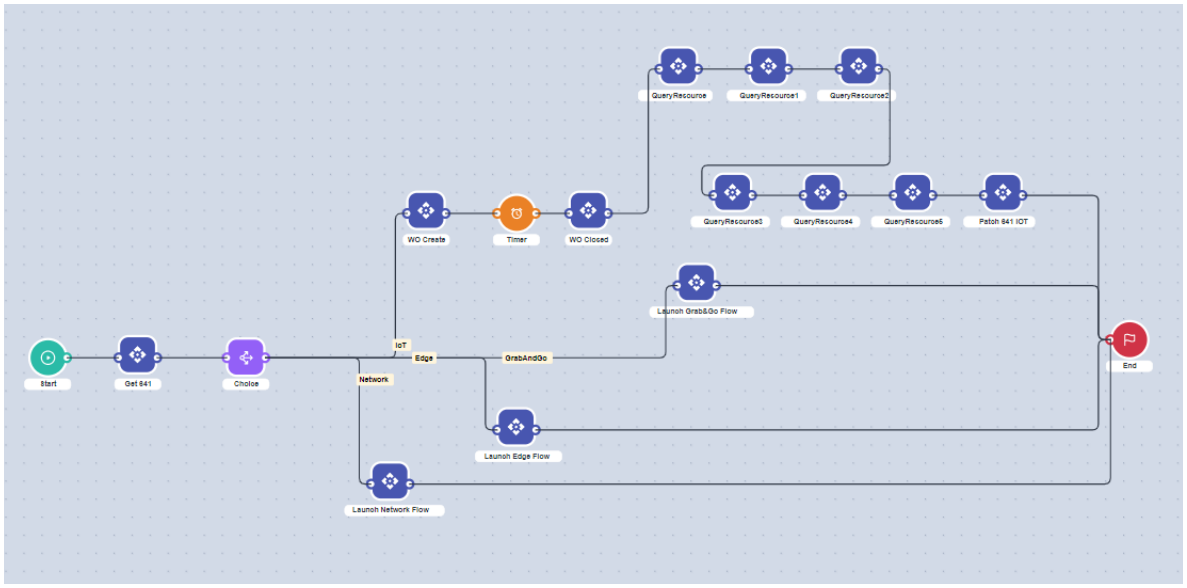
This block Based on the kafka url/uri, we can post messages.

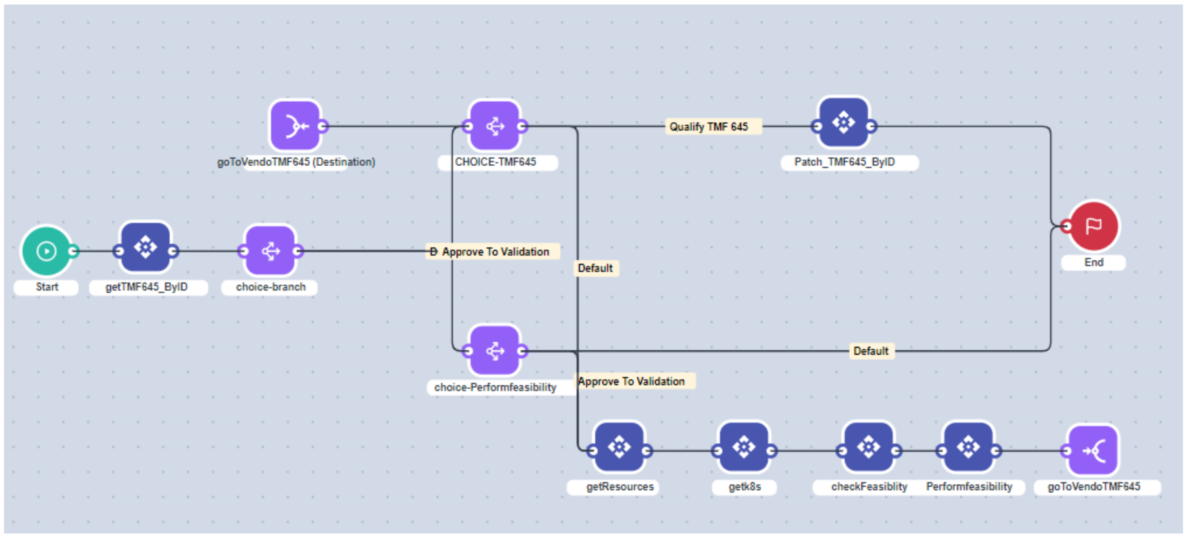
For each loop

This block allows the option to create loops.

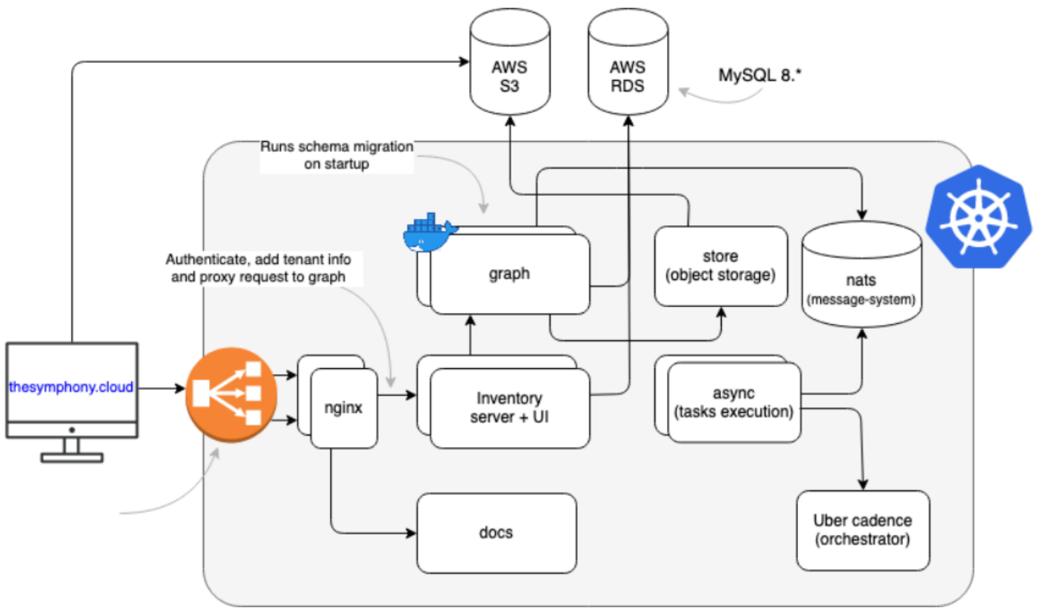
Use Cases:

- **Docomo**
  - In **Docomo** the Automation Engine allows the creation of workflow templates through the GUI simplifying the definition of workflows that can be instantiated to perform different functions over the Docomo network. Several flows can be defined in the automation engine to enable the interaction between the different systems of the SMO (e.g. NFMF, Non-RT RIC, NSSMF, EM) to accomplish functionalities related to the NF management.
- **Catalyst 2022**
  - We use Symphony automation in the Catalyst 2022 to execute multi-domain orchestration flows, from the feasibility analysis, where the Automation interacts with the infrastructure and inventory APIs. It then executes the activation flows of the IoT domain where it generates work orders, then networks where it uses TMF640 and 3GPP APIs, and finally the infrastructure and application domain where it deploys using GitOps.


Start
Get 641
Choice
IoT
Edge
Network
GrabAndGo
WO Create
Timer
WO Closed
QueryResource
QueryResource1
QueryResource2
QueryResource3
QueryResource4
QueryResource5
Patch 641 IOT
Launch Grab&Go Flow
Launch Edge Flow
Launch Network Flow
End


goToVendoTMF645 (Destination)
CHOICE-TMF645
Qualify TMF 645
Patch_TMF645_ByID
Start
getTMF645_ByID
choice-branch
Approve To Validation
Default
End
choice-Performfeasibility
Default
Approve To Validation
getResources
getk8s
checkFeasiblity
Performfeasibility
goToVendoTMF645

Production Environment



Automation Manual

Development

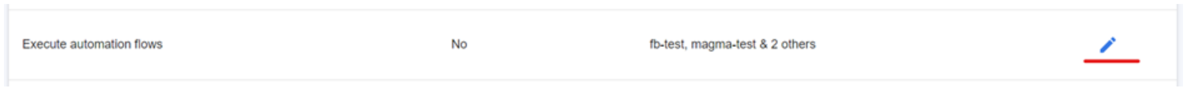
NTT DaTa

User Guide

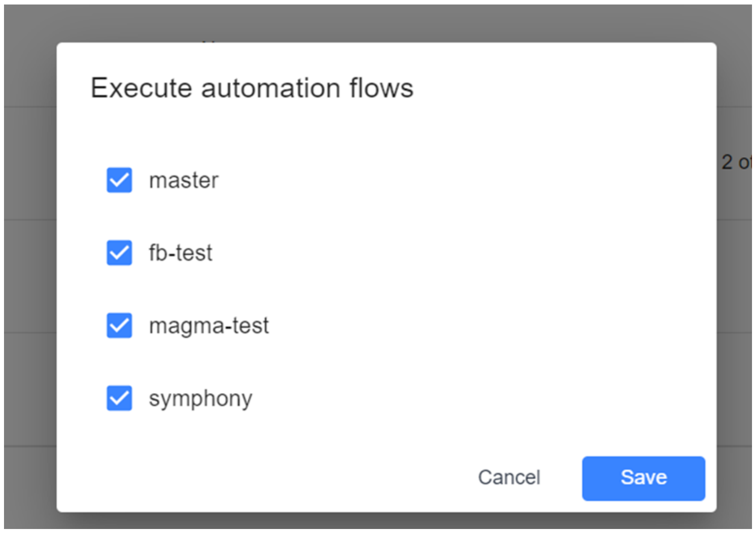
===

Pre requisite:

First validate that the Feature Execute automation flows are enabled like that:



Execute automation flows No fb-test, magma-test & 2 others

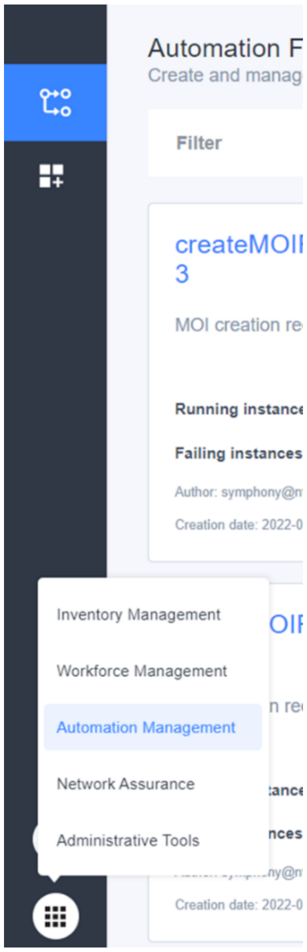


Execute automation flows

- ✓ master
- ✓ fb-test
- ✓ magma-test
- ✓ symphony

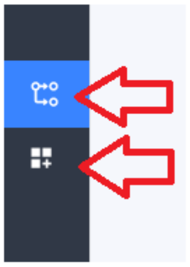
Locate Automation:

In Symphony select the respective option of AUTOMATION MANAGEMENT

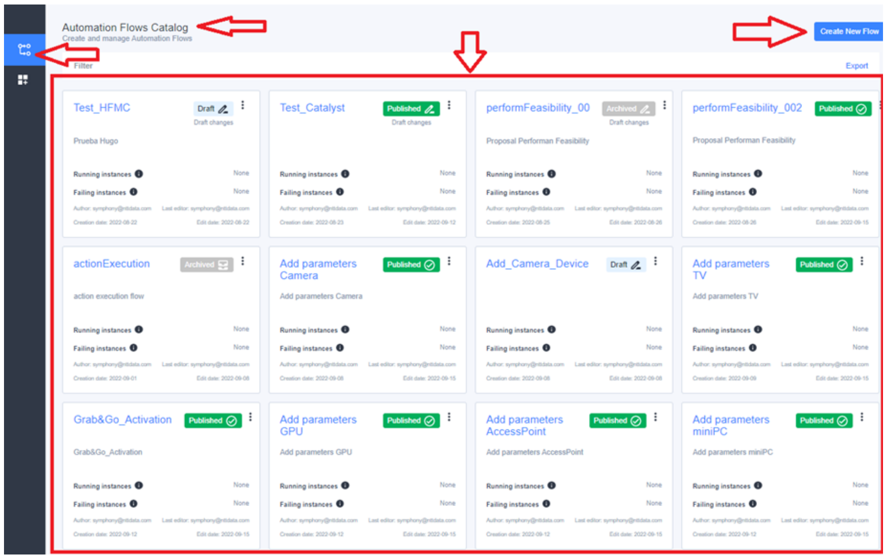


When you select the respective option, you enter to Automation Dashboard. In this, are two options

Automation flows and operation:



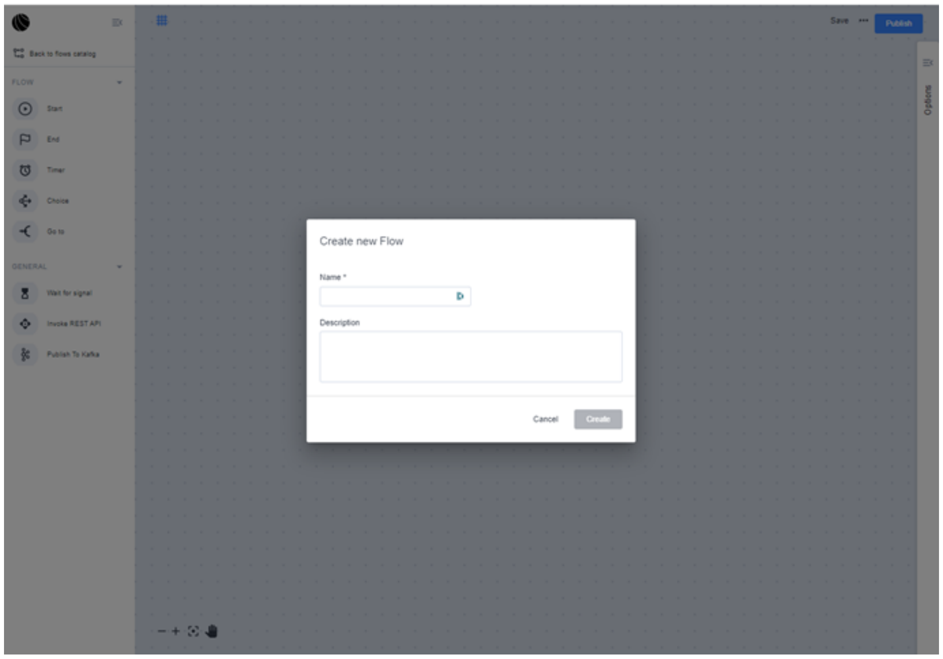
1. Into Automation flows:

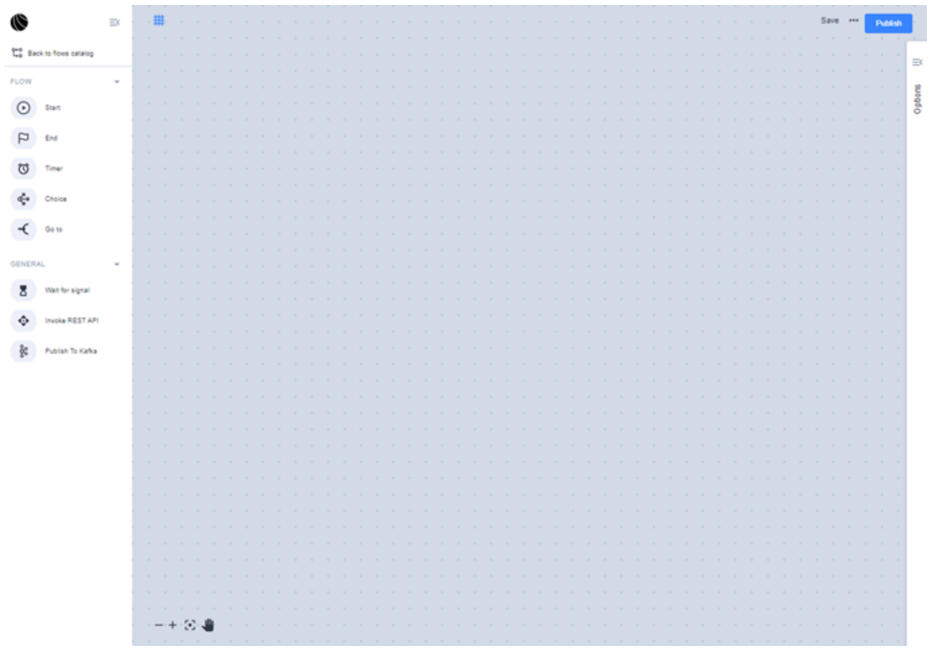


When you select the automation flow option:

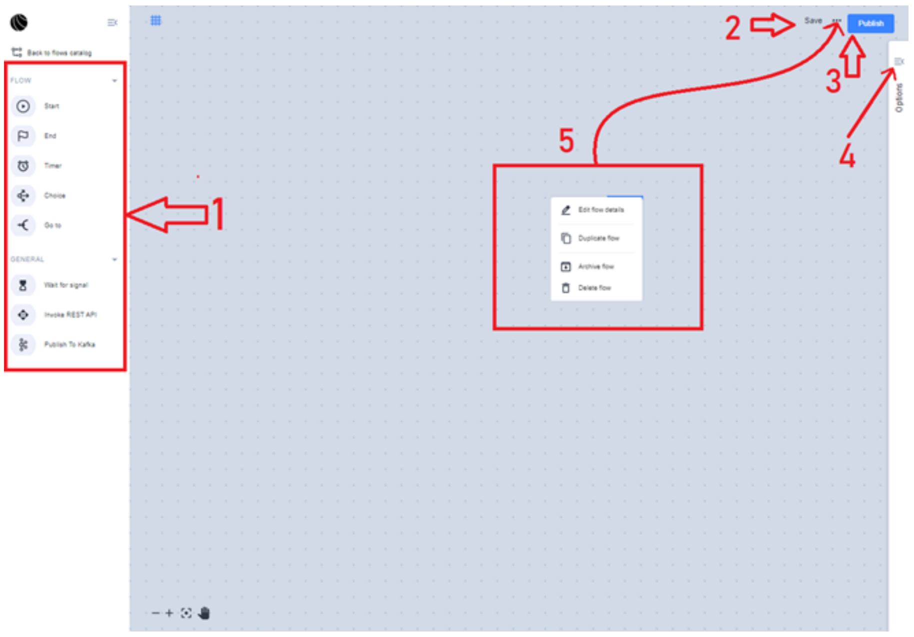
- It will turn blue for the selection option
- In the top you will see the respective name
- In the right top, there is a button for create new flow
- The selected rectangle you will see all the available flows that previously the user/s created.

1.1. Create New Flow:





When you are into the create new flow, the first step is to insert the name and description for the new flow, after that, you will now interact with dashboard flows.



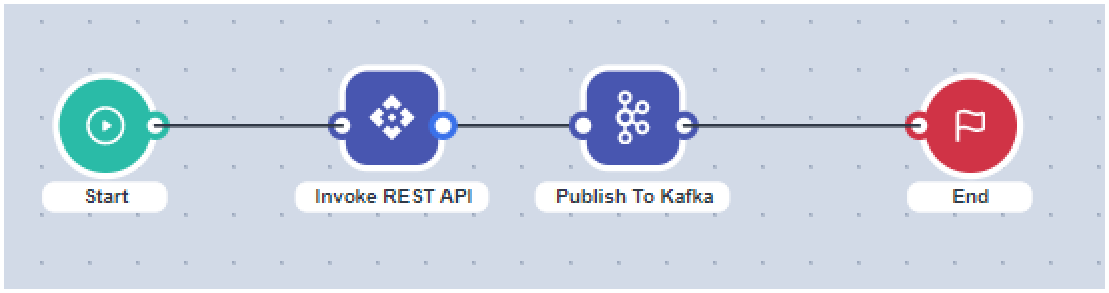
1. You can select the respective blocks to work.
2. Save: option to keep the actual draft.
3. Publish: when the flows are ready, with the publish you make it available for a respective launch.

4. Options Panel: show information about the draft or blocks.
5. Provide 4 options:

* Edit flow drafts: Name or description about the flow.
* Duplicate flow: make a duplicate to the current flow.
* Archive flow: Make the flow in archive status.
* Delete flow: Erase the current block.

1.2. BLOCKS:

Is important to all flows understand the interact always are Start block next blocks and finally End block.



A description of the function of each block and the items that can be set in the Options window are as follows.

BlockName	Function	Configuration	Input	Output	Error Handling	Remarks
Start	The starting point of the automation workflow	- Name				Only one can be placedon the canvas
End	End of the Automated workflow	- Name				
Timer	-Set time elapses -Wait for automation workflow until a specified date and time	Name Behavior ExpressionLanguage			- RetryPolicy	Set the value in a language: unix epoch or ISO8601 If the value is set in a language, it must be in unix epoch or ISO8601 format.
Choice	Branching automation workflow according to defined rules	- Name - Namerule - Add rule			- RetryPolicy	Multiple rules can be set
Go to	Move automation workflow that has reached Go to (Origin) to Go to (Destination)	Name Type			- RetryPolicy	Origin and Destinationcan be changed in Type
Wait for signal	Wait for automated workflow until a signal matching the definition arrives	Name SignalModule Signal Type Custom Filter Block flowuntil reception		- Transform Output - Transform State - Add original input tooutput	- RetryPolicy	

InvokeREST API	Invoke REST API endpoint	Name URL Method URL ConnectionTimeout Headers Body Content Auth Type	- Transform Input - Transform State	- Transform Output - Transform State - Add original input tooutput	- RetryPolicy	
PublishTo Kafka	Publish a message to Kafka	Name Brokers Topic MessageType	- Transform Input - Transform State	- Transform Output - Transform State - Add original input tooutput		

1. Start:

Is mandatory to Start all flows with this block.
Only the Configuration tab exists in the Start block.
The items that can be set on the Configuration tab of the Start block are as follows.

Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block

---

2. End:

Is mandatory to close the flow.
Only the Configuration tab exists in the Start block.
The items that can be set on the Configuration tab of the End block are as follows.

Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block

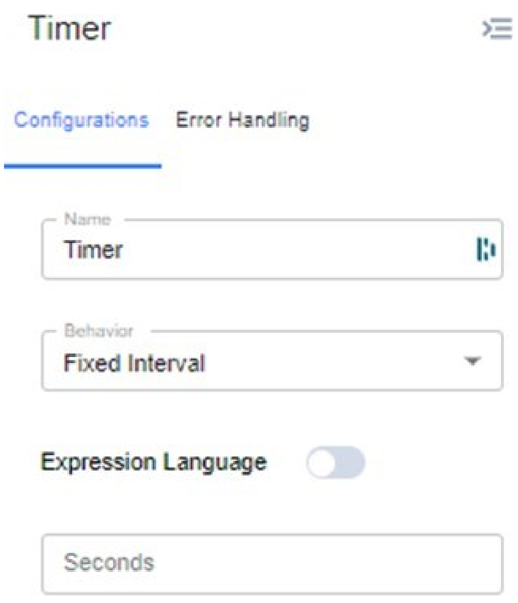
---

3. Timer:

This block provides the option to establish an interval for a concrete interaction Into the modify options. The Timer block has a Configuration tab and an Error Handling tab.

The items that can be set on the Configuration tab of the Timer block are as follows.

Configuration:


Timer

Configurations | Error Handling

Name: Timer

Behavior: Fixed Interval

Expression Language

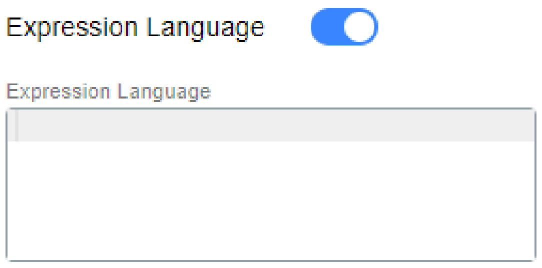
Seconds

Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block
Behavior	Pull-down	Choose fromthe following Fixed Interval Specificdate time	Select the conditions under which the automation workflow will resume Specified time elapsed Specific date and time Both fixed interval and specific date and time allow to use literal values, or to usevariables/expression language ad defined in section 1.3.
Expression Language	ON/OFF Button	Toggle ON/OFF	When ON, a description field appears
Expression Language	Text input	Unlimited	Input only if Expression Language button is ON Enter time/day in Expression Language expression language and unix epoch orISO8601 format
Seconds	Either of the following Enter anumber Spin button	Half-width number	Set the number of seconds to wait in the Input Timer blockif Behavior is Fixed Interval and Expression Language is OFF Set the number of seconds to wait in the Input Timer block if Behavior is FixedInterval and Expression Language is OFF
Time slot start	Calendar/Clock dialog	No input allowed	Set the date and time to restart the automation workflow with the input calendarand clock if Behavior is Specific date and time and Expression Language is OFF

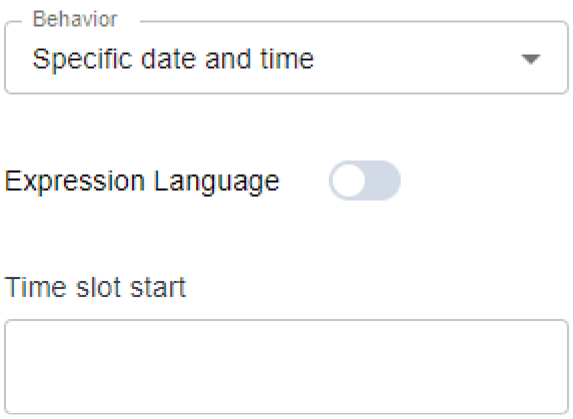
Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

Item Name	Input Method	Input/Select Format	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of the following Enter anumber Spin button	Half-width number	Set the interval for retry processing
Units	Pull-down	Select from the following Seconds Minutes Hours	Set Retry Interval Units
Max Attemps	One of the following numeric input spin button	single-byte numbers	Set the maximum number of retry processing attempts
Backoff rate	One of the following Enter anumber Spin button	single-byte number	Set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

- If Expression Language is ON, the Expression Language field is displayed instead of the Seconds and Time slot start fields.



- If Behavior is Specific date and time and Expression Language is OFF, the Time slot start field is displayed instead of the Seconds field.

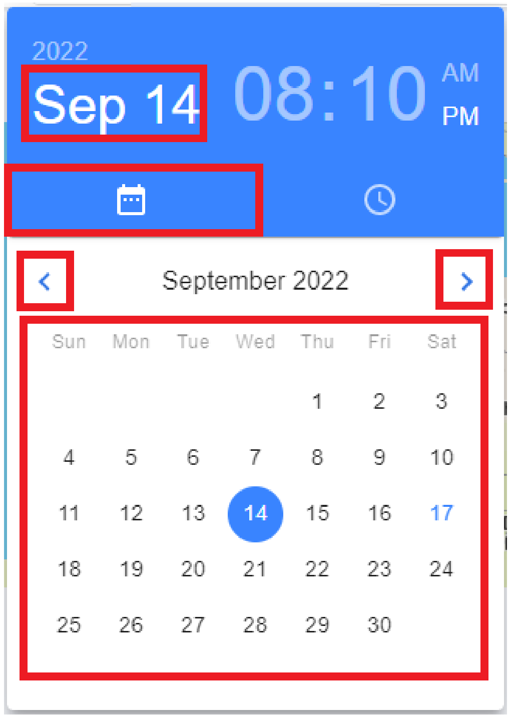

Behavior: Specific date and time

Expression Language

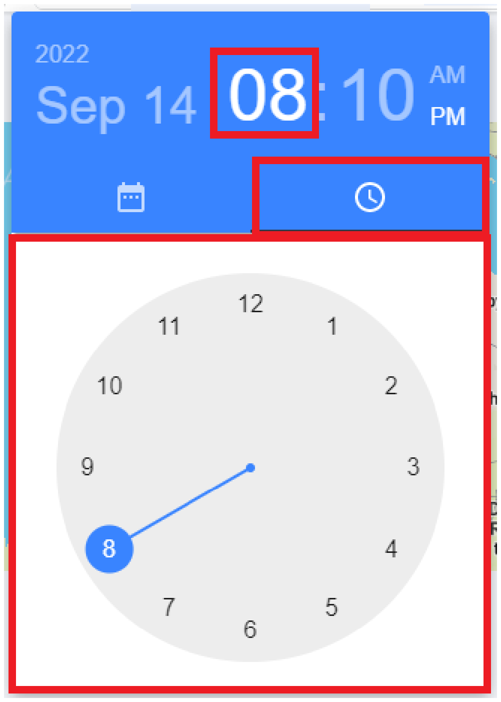
Time slot start

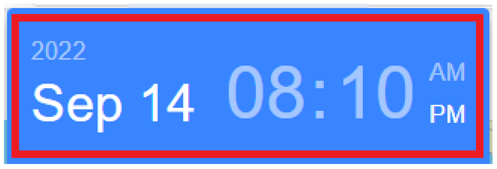
Clicking on the entry field will bring up a calendar/clock dialog that allows you to set the date and time. In the dialog, you can switch between the calendar and clock tabs to set the date and time.

You can also switch between each item and AM/PM by clicking on each item in the date/time display at the top of the dialog.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	



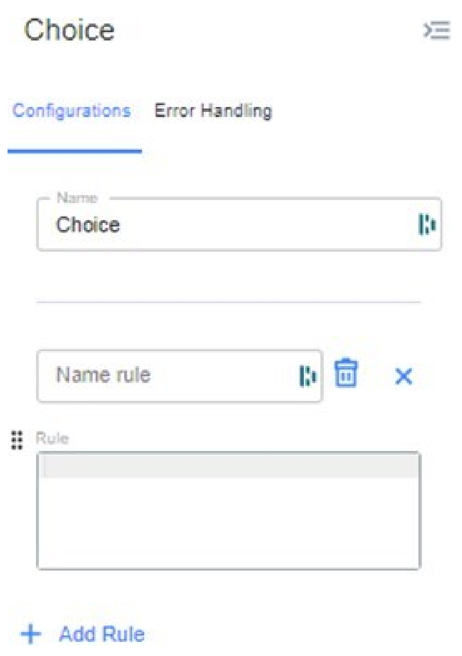


4. Choice:

The Choice block branches the automation workflow according to the rules you define. The Choice block has a Configurations tab and an Error Handling tab. The following items can be set on the Configurations tab of the Choice block.

This block stablishes and logical way into the flow with a respective define rule, in the options of this block we found:

Configuration:



Item Name	Input Method	Input/Select Format	Description
Name	Textinput	No restrictions	Set the name of the block
Name rule	Enter text	No restrictions	Set the name of the rule Click the trash button to delete the rule Click the pencil button to make the Rule field appear
Rule	Text input	No restrictions	Fill in the conditions for branching in Expression Language expression language The order of rules is switched by dragging the button with six dots, where the branching decision is made preferentially from the rule located at the top of the window
Add Rule	No value	No input allowed	Click to open a new Name rule field

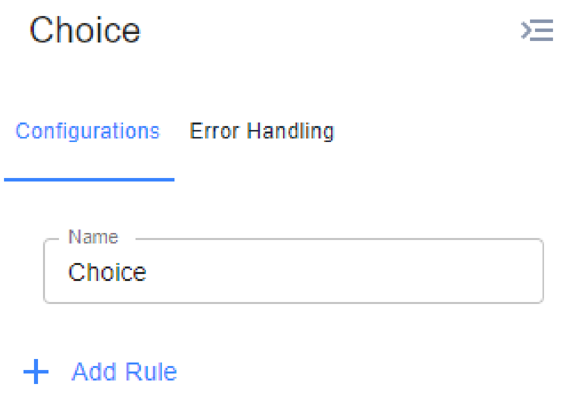
Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

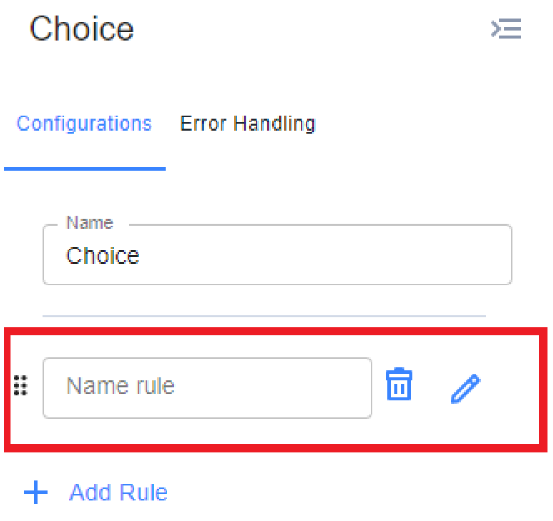
Item Name	Input Method	Input/Select Format	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of the following Enter anumber Spin button	Half-width number	Set the interval for retry processing
Units	Pull-down	Select from the following Seconds	Set Retry Interval Units

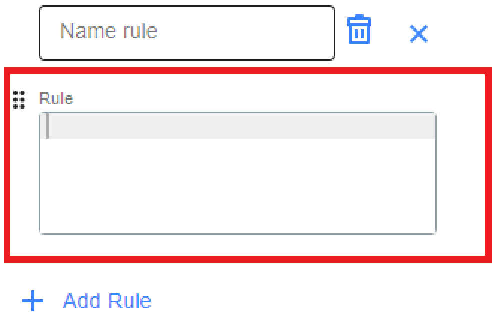
		Minutes Hours	
Max Attemps	one of the following numeric input spin button	single-byte numbers	set the maximum number of retry processing attempts
Backoff rate	one of the following Enter anumber Spin button	single-byte number	set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

- Click Add rule to display the Name rule field. On the right side of the Name rule column, a trash button and a pencil button are displayed. Clicking the trash button deletes the Name rule column.

- Click the pencil button to display the Rule field and enter a rule.

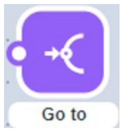


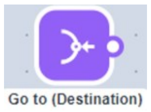




---

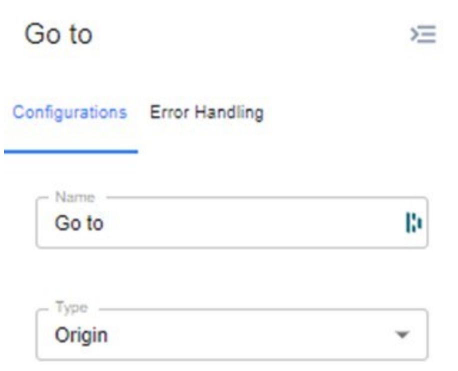
5. Go to:





This block a communication to move into the flow. There are two types of go to blocks: Go to (Origin) and Go to (Destination).

Configuration:



Name: Go to

Type: Origin

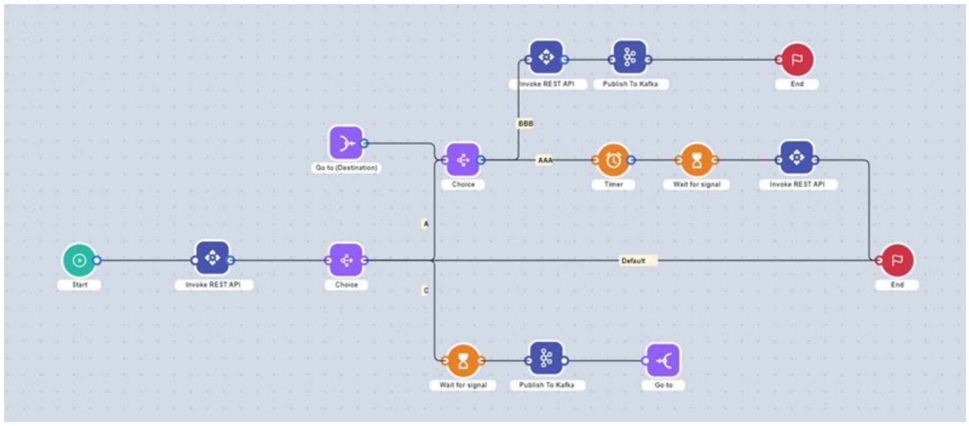
Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block
Type	Pull-down	Choose from the following Origin Destination	Select Origin to change the block to Go to(Origin) Select Destination to change the block to Go to(Destination)

- Automation workflows that reach the Go to (Origin) block are moved to a Go to (Destination) block that exists elsewhere on the canvas. It is used when the automation workflow is long, for example, and can improve overall visibility by separating the automation workflow.
- If there are multiple combinations of Go to blocks, the automated workflow that reaches the Go to (Origin) block will be moved to the Go to (Destination) block with the same Name column.

Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

Item Name	Input Method	Input/Select Format	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of the following Enter anumber Spin button	Half-width number	Set the interval for retry processing
Units	Pull-down	Select from the following Seconds Minutes Hours	Set Retry Interval Units
Max Attemps	one of the following numeric input spin button	single-byte numbers	Set the maximum number of retry processing attempts
Backoff rate	one of the following Enter anumber Spin button	single-byte number	Set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

Example

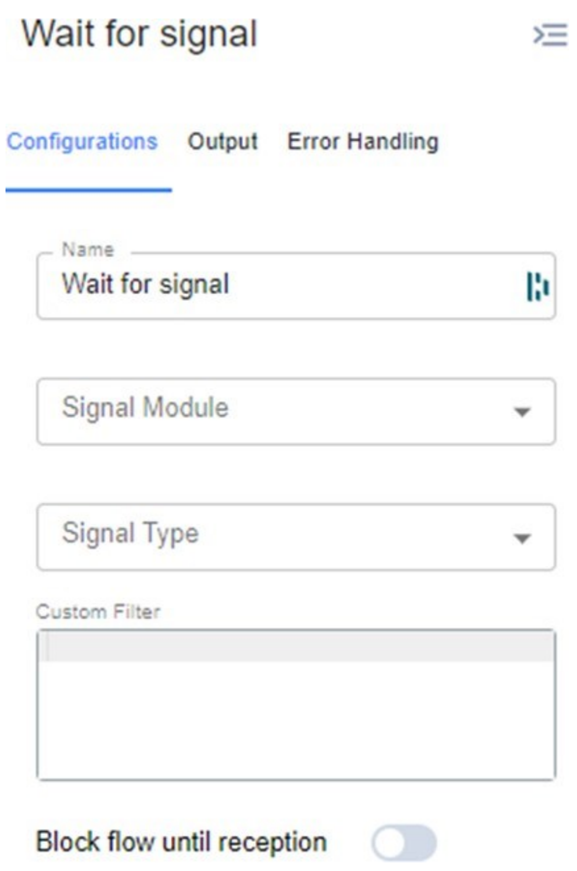


6. Wait for Signal:

This block brings the option to wait the respective signal from Symphony. The Wait for signal block contains the Configurations tab, the Output tab, and the Error Handling tab. The following items can be set on the Configurations tab of the Wait for signal block.

Configuration:

- Name


Wait for signal

Configurations Output Error Handling

Name: Wait for signal

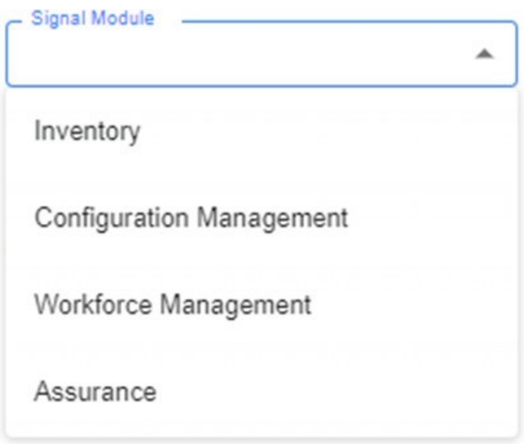
Signal Module

Signal Type

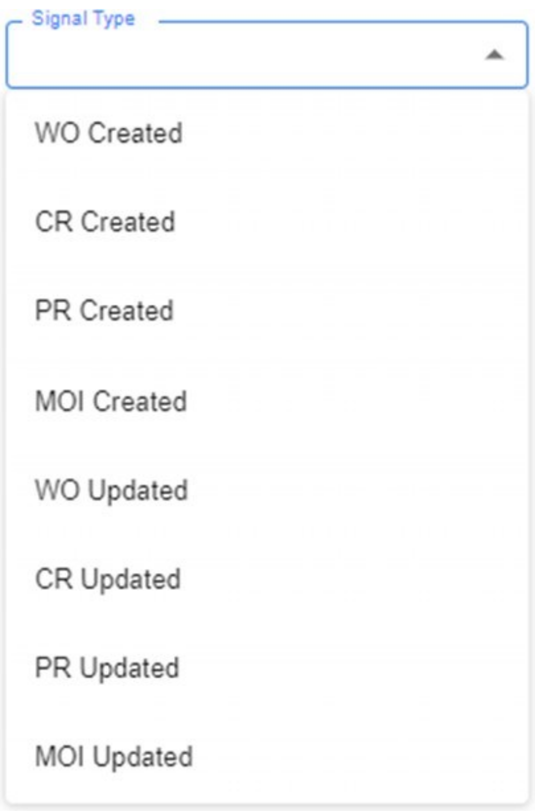
Custom Filter

Block flow until reception

- Signal Module



- Signal Type

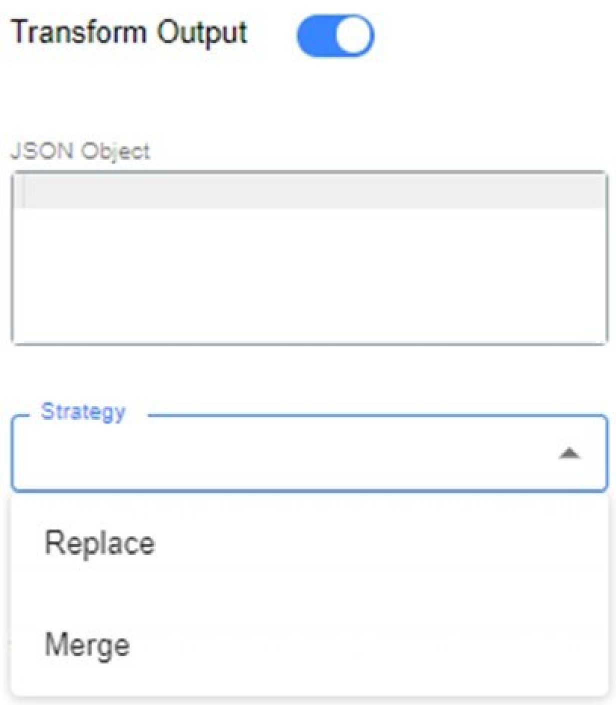


Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block
Signal Module	Pull-down	Choose from the following Inventory	Select the module for the specified signal

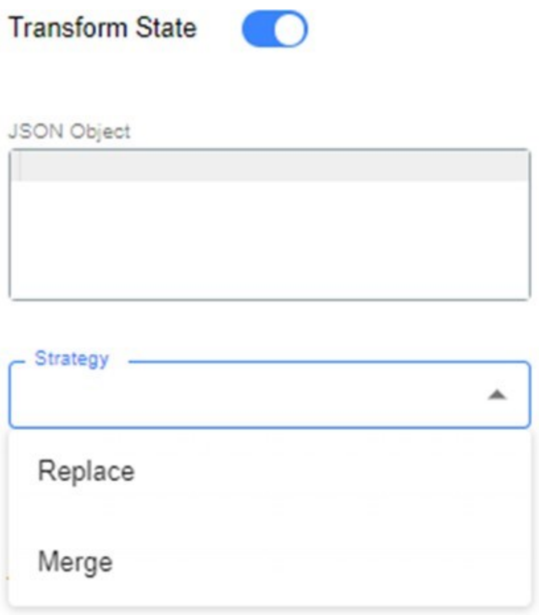
		Configuration Management Workforce Management Assurance	
Signal Type	Pull-down	Select from the following WO Created CR Created PR Created MOI Created WO Updated CR Updated PR Updated MOI Updated	Select the format of the signal to be specified WO stands for Work Order CR stands for Change Request PR standsfor Project Request
Custom Filter	Character Input	Unrestricted	Additional conditions for signal in Expression Language expressionlanguage
Block flow until reception	ON/OFF button	Toggle ON/OFF	Select whether to block automation workflow until the specifiedsignal is received

Output: This option provides 3 configurations:

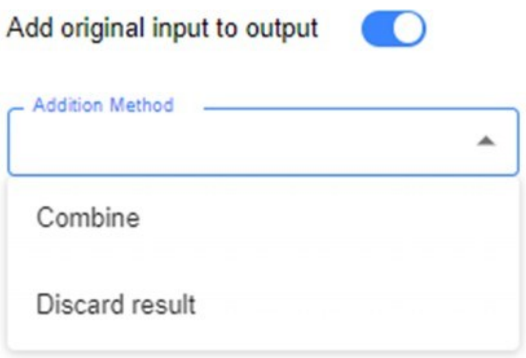
- Transform Output



- Transform State



- Add original input to output



Item Name	Input Method	Input/Select Format	Description
Transform Output	ON/OFF button	Toggle ON/OFF	Set output transform rule ON/OFF
Transform State	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
Add original Input to output	ON/OFF button	Toggle ON/OFF	Set option ON/OFF to add
JSON Object	Character input	Unrestricted	Transform Output, Transform State is ON only, input occurrence transformation rules in JSON format
Strategy	pull-down	select from the following Replace Merge	only appears if Transform Output, Transform State is ON If Replace, discard the output /state before transformation If Merge, append the output/state beforetransformation to the result

Addition Method	Pull-down	Select from the following Combine Discard result	Appears only if Add original Input to output is ON. If Discard result is ON, the input before conversion is discarded.

Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

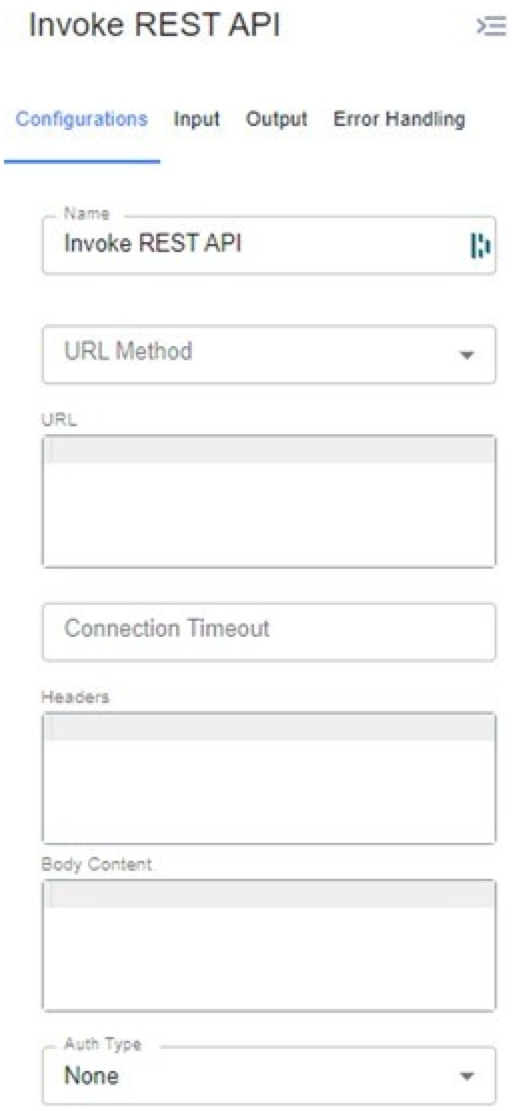
Item Name	Input Method	Input/Select Format	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of the following Enter anumber Spin button	Half-width number	Set the interval for retry processing
Units	Pull-down	Select from the following Seconds Minutes Hours	Set Retry Interval Units
Max Attemps	one of the following numeric input spin button	single-byte numbers	Set the maximum number of retry processing attempts
Backoff rate	one of the following Enter anumber Spin button	single-byte number	set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

7. Invoke Rest API:

This block brings the option to establish communication via Rest API, in this block. The Invoke REST API block has a Configurations tab, an Inputtab, an Output tab, and an Error Handling tab. The following items can be configured on the Configurations tab of the Invoke REST API block.

Configuration:

- If Auth Type is None



Invoke REST API

Configurations Input Output Error Handling

Name: Invoke REST API

URL Method

URL

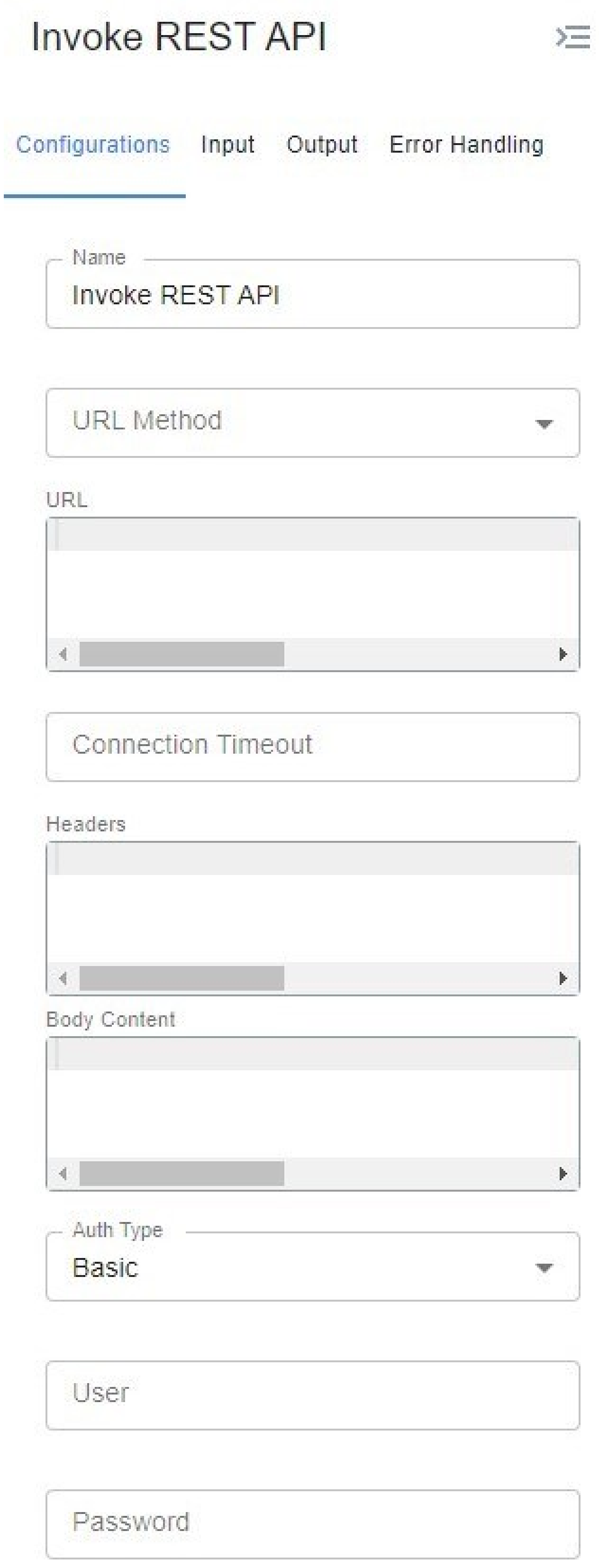
Connection Timeout

Headers

Body Content

Auth Type: None

- If Auth Type is Basic



Invoke REST API

Name: Invoke REST API

URL Method

URL

Connection Timeout

Headers

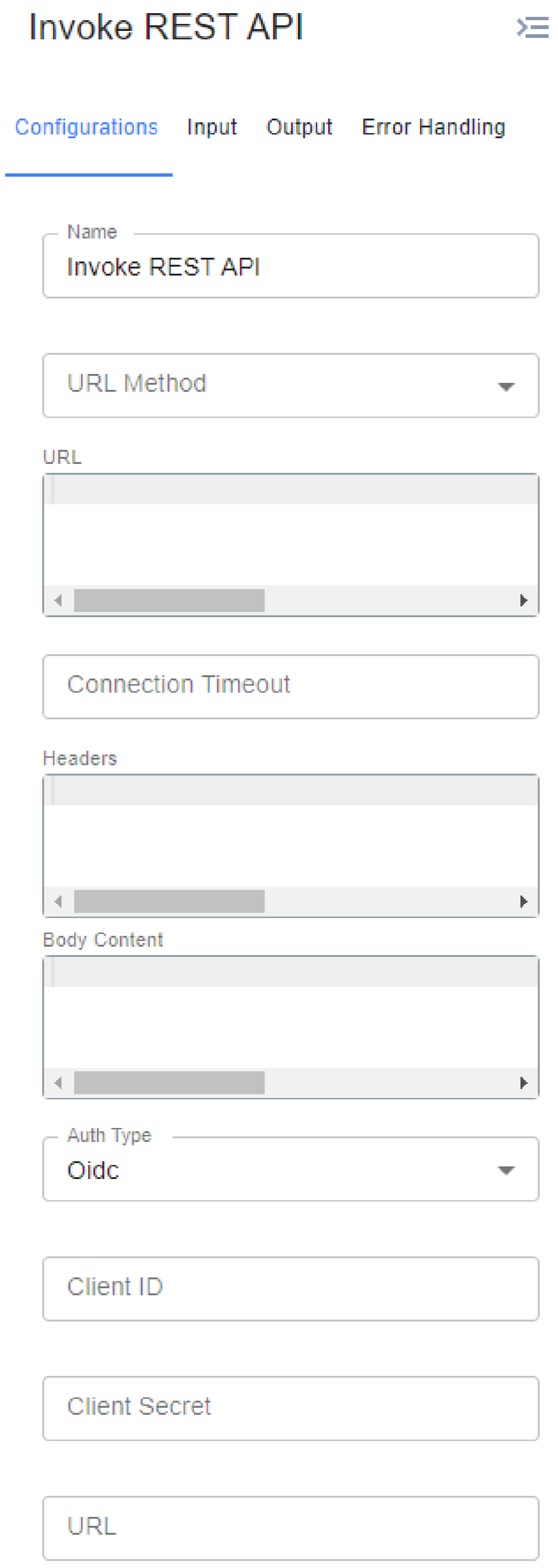
Body Content

Auth Type: Basic

User

Password

- If Auth Type is Oidc



Invoke REST API

Configurations | Input | Output | Error Handling

Name: Invoke REST API

URL Method

URL

Connection Timeout

Headers

Body Content

Auth Type: Oidc

Client ID

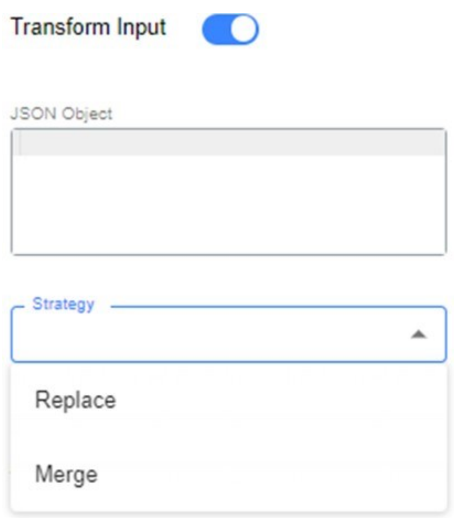
Client Secret

URL

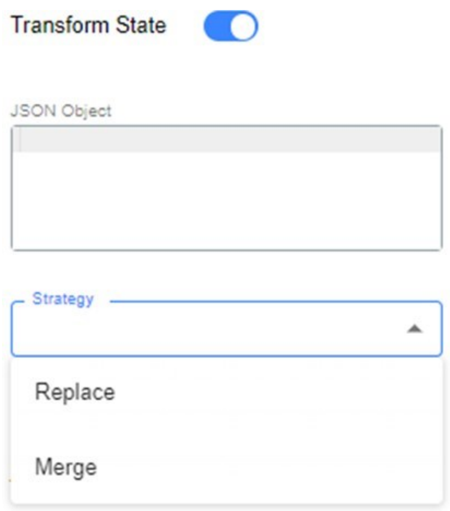
Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block
URL Method	pull-down	choose from the following GET POST PUT DELETE PATCH	select the URL method to call
URL	text input	no restrictions	set URL to call
Connection Timeout	Either of thefollowing Enter a number Spin button	Half-width number	Set the number of seconds an API call will time out
Headers	Character input	Unrestricted	Set API headers to call
Body Content	Character Input	Unrestricted	Set the body of the API to call
Auth Type	pull-down	select from thefollowing None Basic Oidc	set the authentication method for the API to call
User	Character input	Unlimited	Appearance only if Auth Type is Basic Enter a valid user name for Basic authentication
Password	Character input	Unlimited	Appearance only if Auth Type is Basic Enter a valid password for Basic Authentication
Client ID	Character input	Unrestricted	Appears only if Auth Type is Oidc Enter a valid client ID for OpenID Connectauthentication
Client Secret	Character Entry	Unrestricted	Appears only if Auth Type is Oidc Enter a valid client secret for OpenID Connectauthentication
URL	Character input	Unlimited	Appears only if Auth Type is Oidc Enter a valid URL for OpenID Connect authentication

Input: This option provides 2 configurations:

- Transform Input



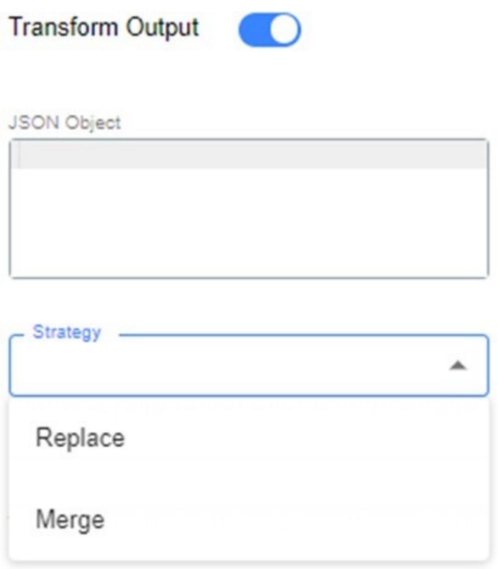
- Transform State



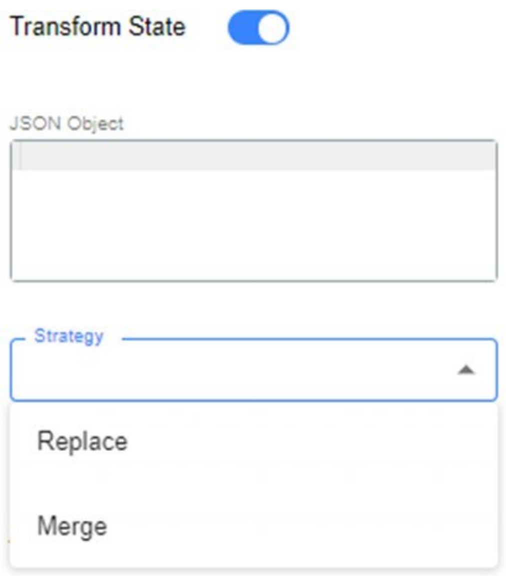
Item Name	Input Method	Input/Select Format	Description
TransformInput	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for input transformation rules
TransformState	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
JSON Object	Character Input	Unrestricted	Transform Input, Transform State must be ON to input occurrence, transformation rules in JSON format
Strategy	pull- down	select fromthe following Replace Merge	appears only if Transform Input, Transform State is ON If Replace, discard the originalinput /state If Merge, append the original input/state to the result

Output: This option provides 3 configurations:

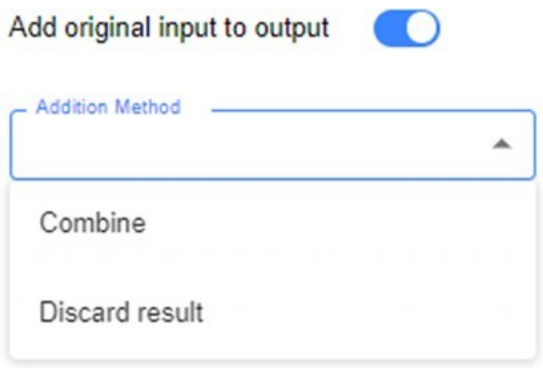
- Transform Output



- Transform State



- Add original input to output



Item Name	Input Method	Input/Select Format	Description
Transform Output	ON/OFF button	Toggle ON/OFF	Set output transform rule ON/OFF
Transform State	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
Add original Input to output	ON/OFF button	Toggle ON/OFF	Set option ON/OFF to add
JSON Object	Character input	Unrestricted	Transform Output, Transform State is ON only, input occurrence transformation rules in JSON format
Strategy	pull-down	select from the following Replace Merge	only appears if Transform Output, Transform State is ON If Replace, discard the output /state before transformation If Merge, append the output/state beforetransformation to the result
Addition Method	Pull-down	Select from the following Combine Discard result	Appears only if Add original Input to output is ON. If Discard result is ON, the input before conversion is discarded.

Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

Item Name	Input Method	Input/Select Format	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of the following Enter anumber Spin button	Half-width number	Set the interval for retry processing
Units	Pull-down	Select from the following	Set Retry Interval Units

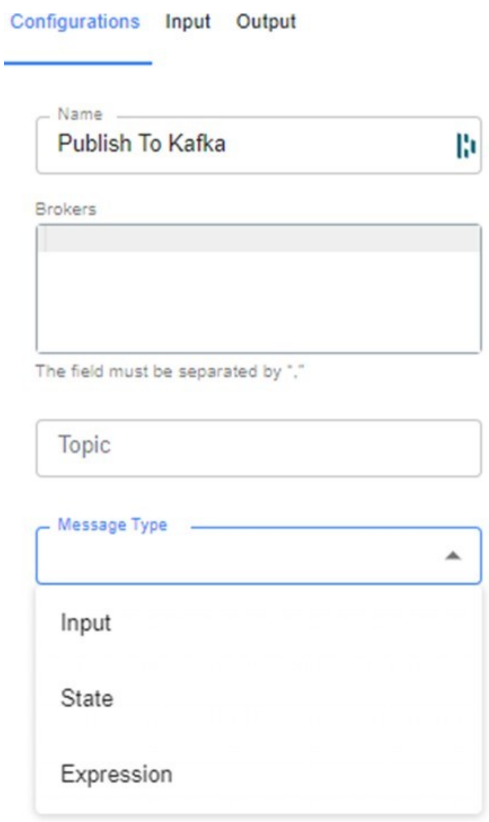
		Seconds Minutes Hours	
Max Attemps	one of the following numeric input spin button	single-byte numbers	set the maximum number of retry processing attempts
Backoff rate	one of the following Enter anumber Spin button	single-byte number	set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

---

8. Publish to Kafka:

This block establishes a connection with a kafka to publish messages to a given queue. The Publish To Kafka block has a Configuration tab, anInput tab, and an Output tab. The following items can be configured on the Configurations tab of the Publish To Kafka block.

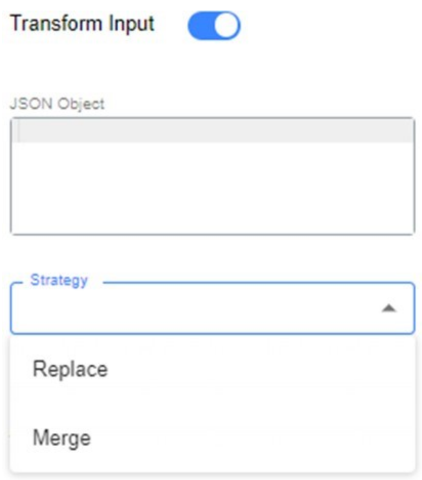
Configuration:



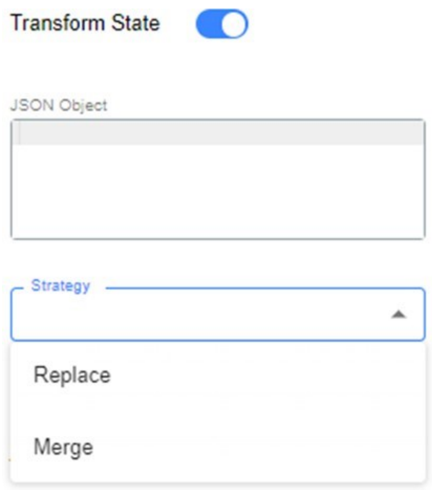
Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block
Brokers	Characterinput	Unlimited	Set Broker for target Kafka
Topic	Text input	Unlimited	Set the target Kafka Topic
Message Type	Pull-down	Select from the following Input State Expression	Select the type of data to publish
Message	Text input	Unrestricted	Occurs only when Message Type is Expression
Enter a message to be published toKafka			

Input: This option provides 2 configurations:

- Transform Input



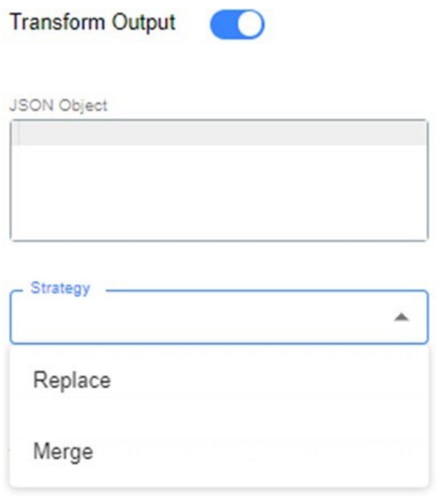
- Transform State



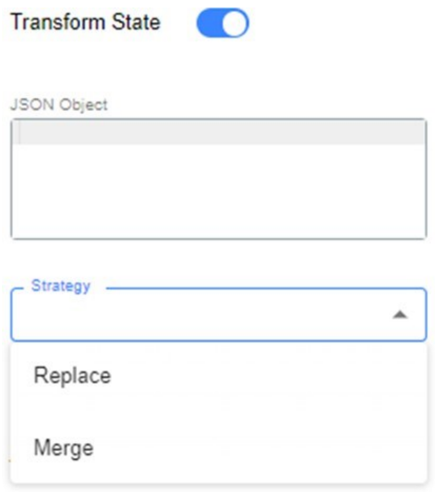
Item Name	Input Method	Input/Select Format	Description
TransformInput	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for input transformation rules
TransformState	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
JSON Object	CharacterInput	Unrestricted	Transform Input, Transform State must be ON to input occurrence, transformation rules in JSON format
Strategy	pull- down	select fromthe following Replace Merge	appears only if Transform Input, Transform State is ON If Replace, discard the originalinput /state If Merge, append the original input/state to the result

Output: This option provides 3 configurations:

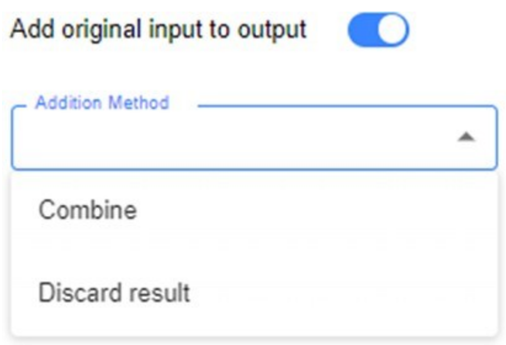
- Transform Output



- Transform State



- Add original input to output



Item Name	Input Method	Input/Select Format	Description
Transform Output	ON/OFF button	Toggle ON/OFF	Set output transform rule ON/OFF
Transform State	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
Add original Input to output	ON/OFF button	Toggle ON/OFF	Set option ON/OFF to add
JSON Object	Character input	Unrestricted	Transform Output, Transform State is ON only, input occurrence transformation rules in JSON format
Strategy	pull-down	select from the following Replace Merge	only appears if Transform Output, Transform State is ON If Replace, discard the output /state before transformation If Merge, append the output/state beforetransformation to the result
Addition Method	Pull-down	Select from the following Combine Discard result	Appears only if Add original Input to output is ON. If Discard result is ON, the input before conversion is discarded.

Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

Item Name	Input Method	Input/SelectFormat	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of thefollowing Enter anumber Spin button	Half-width number	Set the interval for retry processing
Units	Pull-down	Select from the following Seconds Minutes Hours	Set Retry Interval Units
Max Attemps	one of thefollowing numeric input	single-bytenumbers	

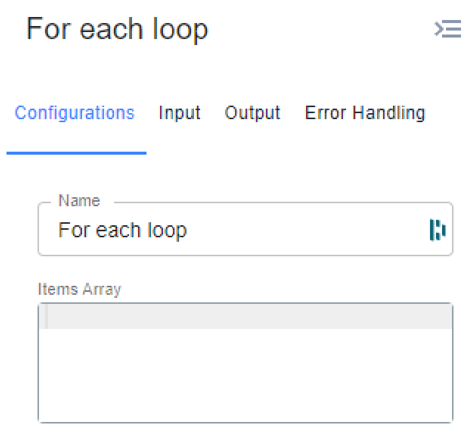
	spin button		set the maximum number of retry processing attempts
Backoffrate	following - Enter anumber	single-byte number	set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

---

9. For each loop:

This block establishes a flow loop. The number of executions can be set by entering it in the Items Array field. The For each loop block has a Configurations tab, an Input tab, an Output tab, and an Error Handling tab. The following items can be set in the Configurations tab of the Foreach loop block.

Configuration:



Name: For each loop

Items Array:

Item Name	Input Method	Input/Select Format	Description
Name	Text input	No restrictions	Set the name of the block
Items Array	Text input	Unlimited	Set number of times to run

It will iterate twice, since the list have two objects, and in the first iteration the following input will be forwarded: {"key1": "value1", "key2":"value2"}, and in the second iteration the input sent to the first internal block in the for each will be {"key1": "value3", "key2": "value4"}.

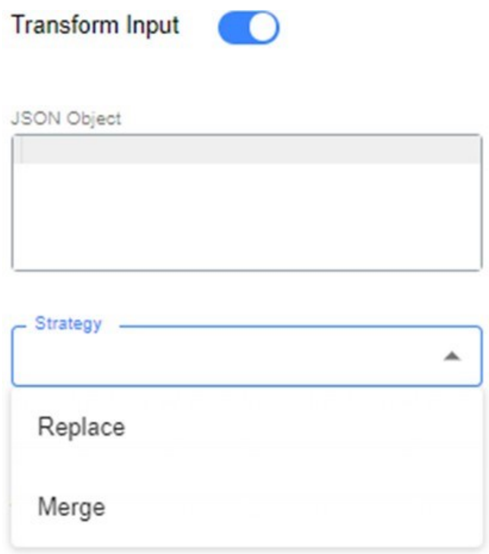
The other way this Items Array field can be configured, is by receiving a JSON containing different keys and object values. In this case the blockinternal and automatically transforms the input sending in each iteration a structure containing the key name on a field the object value in a different field. For example, if the input is: {"CU-02": {"key1":

"value1"}, "DU-01": {"key2": "value2}}, the block will iterate twice, and in the first iteration it will send as input {"key": "CU-02", "value": {"key1":"value1"}}. In the second iteration, it will send as input {"key": "DU-01", "value":

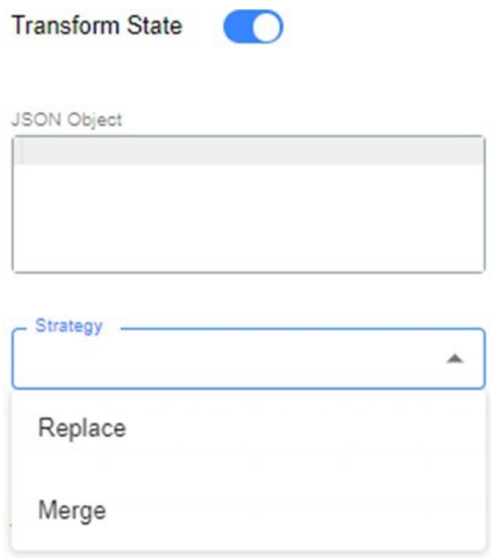
{"key2": "value2}}.

Input: This option provides 2 configurations:

- Transform Input



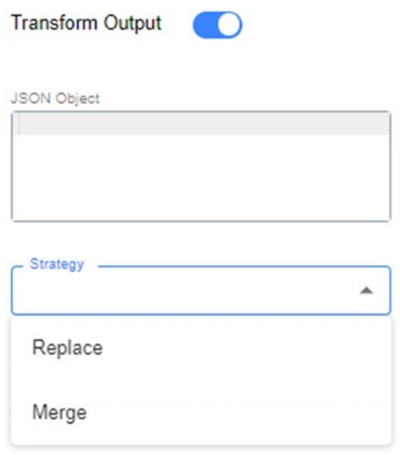
- Transform State



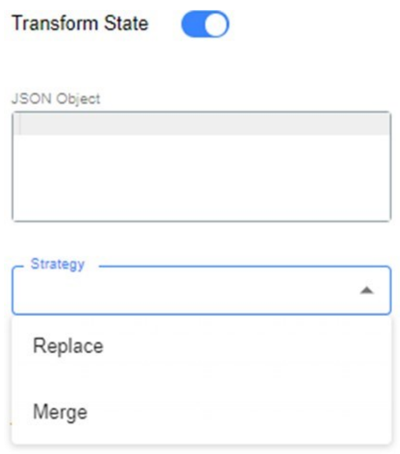
Item Name	Input Method	Input/Select Format	Description
TransformInput	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for input transformation rules
TransformState	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
JSON Object	CharacterInput	Unrestricted	Transform Input, Transform State must be ON to input occurrence, transformation rules in JSON format
Strategy	pull- down	select fromthe following Replace Merge	appears only if Transform Input, Transform State is ON If Replace, discard the originalinput /state If Merge, append the original input/state to the result

Output: This option provides 3 configurations:

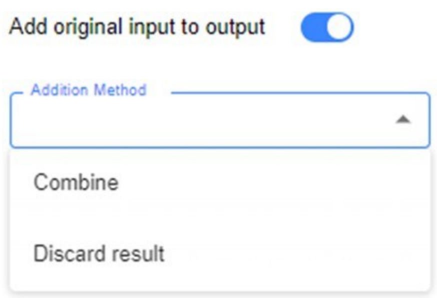
- Transform Output



- Transform State



- Add original input to output



Item Name	Input Method	Input/Select Format	Description
Transform Output	ON/OFF button	Toggle ON/OFF	Set output transform rule ON/OFF
Transform State	ON/OFF Button	Toggle ON/OFF	Set ON/OFF for state transformation rules
Add original Input to output	ON/OFF button	Toggle ON/OFF	Set option ON/OFF to add
JSON Object	Character input	Unrestricted	Transform Output, Transform State is ON only, input occurrence transformation rules in JSON format
Strategy	pull-down	select from the following Replace Merge	only appears if Transform Output, Transform State is ON If Replace, discard the output /state before transformation If Merge, append the output/state beforetransformation to the result
Addition Method	Pull-down	Select from the following Combine Discard result	Appears only if Add original Input to output is ON. If Discard result is ON, the input before conversion is discarded.

Error Handling: Its function is to have retry policies when the logic of the block fails or does not complete.

Item Name	Input Method	Input/Select Format	Description
RetryPolicy	ON/OFF button	No input allowed	Set retry policy ON/OFF
Retry Interval	One of the following Enter anumber Spin button	Half-width number	Set the interval for retry processing

Units	Pull-down	Select from the following Seconds Minutes Hours	Set Retry Interval Units
Max Attemps	one of the following numeric input spin button	single-byte numbers	set the maximum number of retry processing attempts
Backoffrat	One of the following: - Enter a number Spin button	single-byte number	set the backoff rate (the rate at which the retry processing intervalincreases with each attempt)

1.3. Expression language for transformation:

Expression language can be used to transform input, output, input state, output state or even to use in block logic fields. The next table summarizes the functions available for data transformation.

input-state	function syntax	output
{"negative_value": -35.7}	{"abs1": math.abs(input.negative_value)}	{"abs1": 35.7} <br/> <br/>
literal input	{"abs2": math.abs(-15)}	{"abs2": 15}
{"value2": 0.4}	{"acos": math.acos(input.value2}	{"acos": 1.1592794807274085}
{"value2": 0.4}	{"acosh": math.acosh(state.value2 }	{"acosh":3.3092083606287246}
{"value2": 0.4}	{"asin": math.asin(input.value2)}	{"asin":0.41151684606748806}
{"value2": 0.4}	{"asinh": math.asinh(state.value2)}	{"asinh":3.311872343563387}
{"value2": 0.4}	{"atan": math.atan(input.value2)}	{"atan":0.3805063771123649}
{"value2": 0.4}	{"atanh": math.atanh(input.value2)}	{"atanh":0.423648930193602}
{"value2": 0.4}	{"cbrt": math.cbrt(state.value2)}	{"cbrt":2.3928025107131377 }
{"value2": 0.4}	{"ceil": math.ceil(state.value2)}	{"ceil":14}
{"value2": 0.4}	{"cos": math.cos(input.value2)}	{"cos":0.921060994002885}
{"value2": 0.4}	{"cosh": math.cosh(state.value2)}	{"cosh":445455.5829901414}
{"code": "Input code"}	{"ends_with1": input.code.endsWith('In')}	{"ends_with1": false} <br/> <br/>
{"code": "Input code"}	{"ends_with2": input.code.endsWith('de')}	{"ends_with2": true} <br/> <br/>
{"value2": 0.4}	{"exp": math.exp(state.value2)}	{"exp":890911.1659791603}
{"value2": 0.4}	{"exp2": math.exp2(state.value2)}	{"exp2":13307.943261900557}
{"value2": 0.4}	{"expm1": math.expm1(state.value2)}	{"expm1":890910.1659791603}
{"value2": 0.4}	{"floor": math.floor(state.value2)}	{"floor":13}
{"value2": 0.4}	{"hypot": math.hypot(input.value2,state.value2)}	{"hypot": 13.705838172107534} <br/> <br/>

{"code": "Input code"}	{"index_of1": input.code.indexOf('de')}	{"index_of1":8}
{"code": "Input code"}	{"index_of2": input.code.indexOf('xe')}	{"index_of2":-1}
{"value": 90.0}	{"input_value": input.value + 10.0}	{"input_value":100}
{"code": "Input code"}	{"join1": strings.join([input.code.trim(), state.code.trim()], ' - ')}	{"join1":"Input code - Inpute code"}
literal input	{"join2": strings.join(['a', 'b', 'c', 'd', 'e'], ', ')}	{"join2":"a, b, c, d, e"} <br/> <br/>
{"value2": 0.4}	{"log": math.log(state.value2)}	{"log":2.617395832834079}
{"value2": 0.4}	{"log10": math.log10(state.value2)}	{"log10":1.1367205671564067}
{"value2": 0.4}	{"log1p": math.log1p(state.value2)}	{"log1p":2.6878474937846906}
{"value2": 0.4}	{"log2": math.log2(state.value2)}	{"log2":3.776103988073164}
{"code": "Input code"}	{"lower": input.code.toLower()}	{"lower":"input code"}
{"value2": 0.4}	{"max": math.max(input.value2, state.value2)}	{"max": 0.4} <br/> <br/>
{"value2": 0.4}	{"min": math.min(input.value2, state.value2)}	{"min":0.4} <br/> <br/>
{"value2": 0.4}	{"mod": math.mod(input.value2, state.value2)}	{"mod":0.4} <br/> <br/>
{"value2": 0.4}	{"pow": math.pow(state.value2, 5)}	{"pow":482617.24456999986}
literal input	{"pow10": math.pow10(5)}	{"pow10":100000}
{"value2": 0.4}	{"remainder": math.remainder(input.value2, state.value2)}	{"remainder":0.4} <br/> <br/> <br/>
{"code": "Input code"}	{"replace": input.code.replaceAll('d', 'x')}	{"replace":"Input coxe"} <br/> <br/>
{"value2": 0.6}	{"round": math.round(state.value2)}	{"round": 1}
{"value2": 0.4}	{"sin": math.sin(input.value2)}	{"sin":0.3894183423086505}
{"value2": 0.4}	{"sinh": math.sinh(state.value2)}	{"sinh": 445455.58298901893}
{"code": "Input code"}	{"size": input.code.size()}	{"size":10}
{"code": "Input code"}	{"split1": input.code.split(' ')} <br/> <br/> <br/> <br/>	{"split1":[<br/>"Input,<br/>"code"<br/>]}
literal input	{"split2": 'Transformation test'.split('t')} <br/> <br/> <br/> <br/> <br/> <br/>	{"split2": [<br/>"Transforma",<br/>"ion ",<br/>"es",<br/>""<br/>]}
{"value2": 0.4}	{"sqrt": math.sqrt(state.value2)}	{"sqrt":3.7013511046643495 }
{"code": "Input code"}	{"starts_with1": input.code.startsWith('In')}	{"starts_with":true} <br/> <br/>
{"code": "Input code"}	{"starts_with2": input.code.starstWith('Ok')}	{"starts_with2":false} <br/> <br/>
{"code": "State code" }	{"state_code": state.code}	{"state_code":"State code"}
literal value	{"static": "STATIC"}	{"static":"STATIC"}
{"code": "State code"}	{"substring1": state.code.substring(2)}	{"substring1":"ate code"}
{"code": "State code"}	{"substring2": state.code.substring(3,7)}	{"substring2":"te c"} <br/> <br/>
{"value2": 0.4}	{"tan": math.tan(input.value2)}	{"tan":0.4227932187381618}
{"value2": 0.4}	{"tanh: math.tanh(input.value2)}	{"tanh:0.42364893019360184}
{"code": "Input code"}	{"trim1": input.code.trim()}	{"trim1":"Input code"}
literal input	{"trim2": ' Trim a literal '.trim()}	{"trim2":"Trim a literal"}

{"value2": 13.4}	{"trunc": math.trunc(state.value2)}	{"trunc":13}
{"code": "Input code"}	{"upper": input.code.toUpper()}	{"upper":"INPUT CODE"}
{"counters": [{"counterName": "a", "counterValue": 1},{"counterName": "b", "counterValue": 2}]}	{"mapCounters": input.counters.map(i,i.counterName: i.counterValue)}	{"mapCounters": ["a": 1, "b": 2]} <br/> <br/> <br/>
{"mapCounters": ["a": 1, "b": 2]}	{"counters": mapagg(input.mapCounters)	{"counters": {"a": 1, "b": 2} <br/> <br/>
{"counter": ""}	{"nullCounter": replaceEmptyByNull(input.counter)}	{"nullCounter": null} <br/> <br/>
{"counters": [{"a": null, "b": 1, "c": null},{"a": 2, "b": null, "c": null}]}	{"counterWithValue": input.counters.getFirst(i,[a,b,c])}	{"counterWithValue": [1,2]} <br/> <br/>
{"parameters": {"ManagedNFService[0].attributes.administrativeState": "LOCKED", "ManagedNFService[0].id": "0"}}	{"body": unflatten(input.parameters)} <br/> <br/> <br/> <br/>	{"body": {"ManagedNFService" : [{"attributes" : {"administrativeState" : "LOCKED"},"id" : "0"}]}}
{"numericValue": 1}	{"stringValue": toString(input.numericValue)}	{"stringValue": "1"} <br/> <br/>
{"stringValue": "1"}	{"numericValue": toNumber(input.stringValue)}	{"numericValue": 1} <br/> <br/>
{"a": 1, "b": 2}	deleteKey(input,"a")	{"b": 2}
{"changeRequestParams": {"ManagedElement[0].GNBCUUPFFunction[0].attributes.NetworkId": "1","ManagedElement[0].NRCellCU[0].attributes.NetworkId": "2"},"currentCMVersion": {"ManagedElement[0].GNBCUUPFFunction[0].id": "0","ManagedElement[0].GNBCUUPFFunction[0].attributes.NetworkId": "3","ManagedElement[0].NRCellCU[0].id": "0", "ManagedElement[0].NRCellCU[0].attributes.NetworkId": "4","ManagedElement[0].NRCellCU[1].id": "1", "ManagedElement[0].NRCellCU[1].attributes.NetworkId": "3"}}	{"data": nested_modify_3gpp(input)} <br/> <br/> <br/> <br/> <br/> <br/> <br/> <br/>	{"data": {"/ManagedElement=DU-01/GNBCUUPFFunction=0": [{"urlSuffix": "/ManagedElement=DU-01/GNBCUUPFFunction=0","body": {"GNBCUUPFFunction": [{"attributes": {"NetworkId": "1"},"id": "0"}]}}],"/ManagedElement=DU-01/NRCellCU=0": [{"urlSuffix": "/ManagedElement=DU-01/NRCellCU=0","body": {"NRCellCU": [{"attributes": {"NetworkId": "2"},"id": "0"}]}}]}}

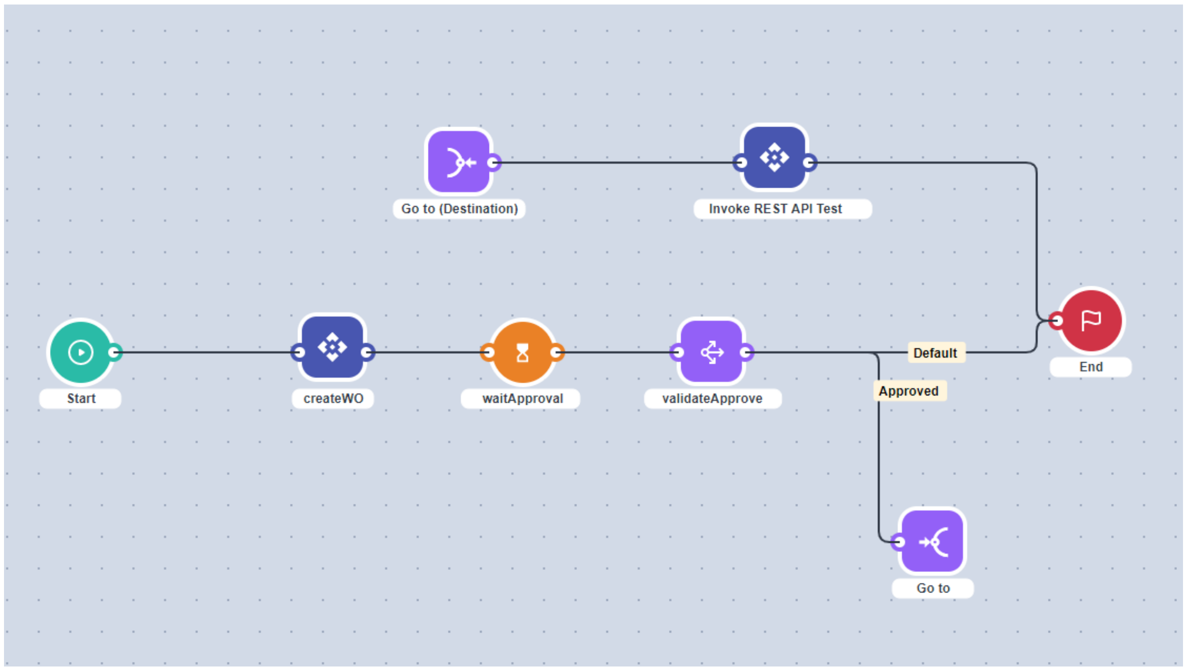
Additionally, operators can be used to build expressions. Operators are summarized in the next table:

Operator Category	Operator symbol
multiplicative	* / %
additive	+ -
comparison	< > <= >=
equality	== !=
logical AND	&&
logical OR	\\|\\|
logical NOT	!

1.4. Examples:

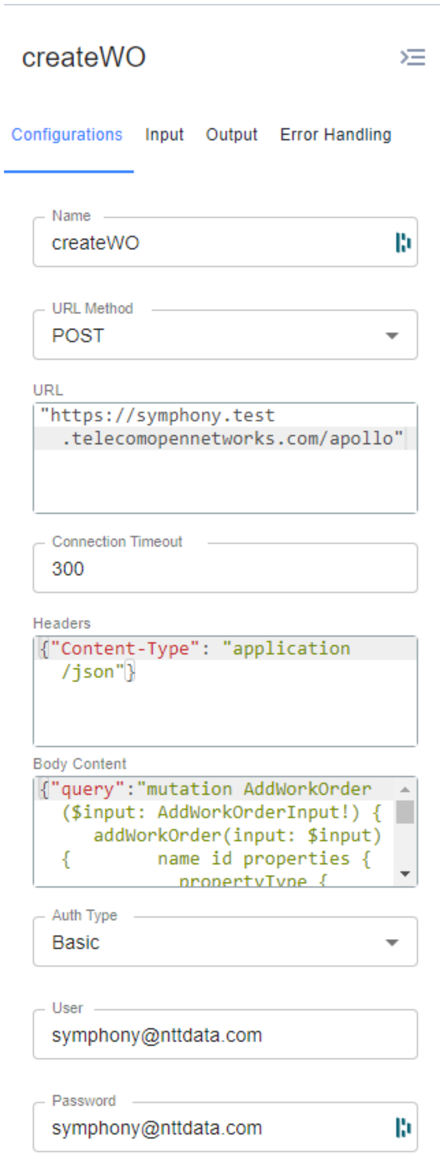
Example (A):

This example shows a creation of a Work order in Symphony, when the process is completed, a Signal arrive to the block “Wait fort Signal”, when the signal is okay, the flow continue to a “Choice” block, where it validates the work order creation, with two scenarios True or False, if true, the sequence go to eh “Go To” Block, and it continue to a finel “Invoke Rest API” block, it apply a patch and finally go to the End block.



Configuration:

For the block “Invoke Rest API” with the name ***createWO***:



createWO

Configurations Input Output Error Handling

Name: createWO

URL Method: POST

URL
```
"https://symphony.test
  .telecomopennetworks.com/apollo"
```

Connection Timeout: 300

Headers
```
{"Content-Type": "application
  /json"}
```

Body Content
```
{"query":"mutation AddWorkOrder
  ($input: AddWorkOrderInput!) {
     addWorkOrder(input: $input)
  {        name id properties {
       propertyType {
```

Auth Type: Basic

User: symphony@nttdata.com

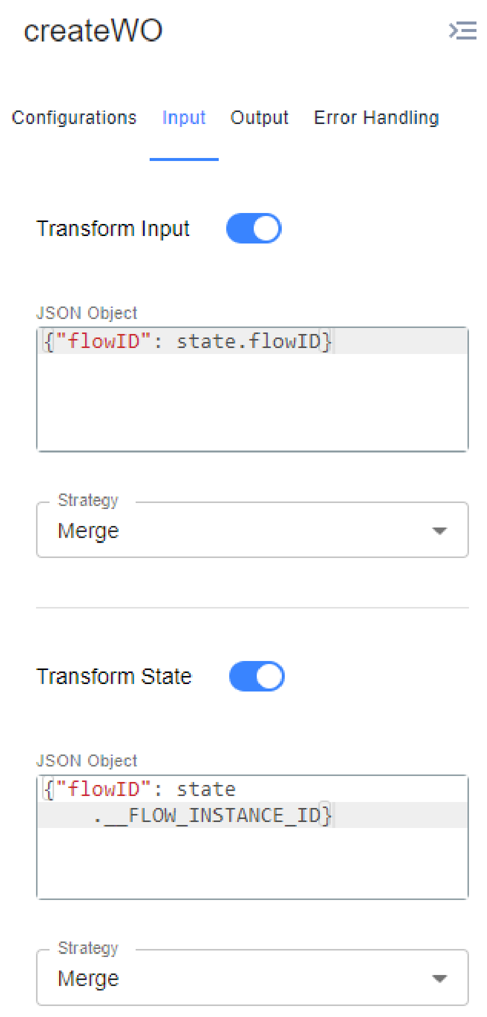
Password: symphony@nttdata.com

For the specific **Body Content**:

It important to know the IDs for:

- WORK_ORDER_TEMPLATE_ID
- ID_PROPERTY_TYPE_TEMPLATE_ID

```
{
  "query": "mutation AddWorkOrder($input: AddWorkOrderInput!) {addWorkOrder(input: $input
) {name id properties {propertyType {name}propertyTypeValue {name}}}}",
  "variables": {
    "input": {
      "name": input.changeId.prepend("WO Order Test "),
      "description": "This WO is oriented to a Test.",
      "assigneeId": input.approver,
      "ownerId": input.approver,
      "workOrderTypeId": "WORK_ORDER_TEMPLATE_ID",
      "status": "PLANNED",
      "priority": "NONE",
      "flowInstanceId": input.flowID,
      "checkListCategories": [],
      "properties": [{
          "booleanValue": false,
          "stringValue": "null",
          "propertyTypeID": "ID_PROPERTY_TYPE_TEMPLATE_ID"
        }
      ]
    }
  }
}
```


createWO

Transform Input: on

JSON Object

```
{"flowID": state.flowID}
```

Strategy: Merge

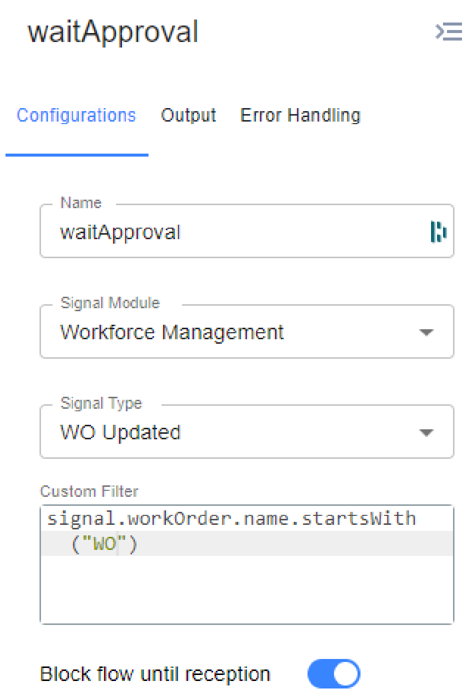
Transform State: on

JSON Object

```
{"flowID": state
    .__FLOW_INSTANCE_ID}
```

Strategy: Merge

For the block “Wait for signal” with the name ***waitApproval***:


waitApproval

Name: waitApproval

Signal Module: Workforce Management

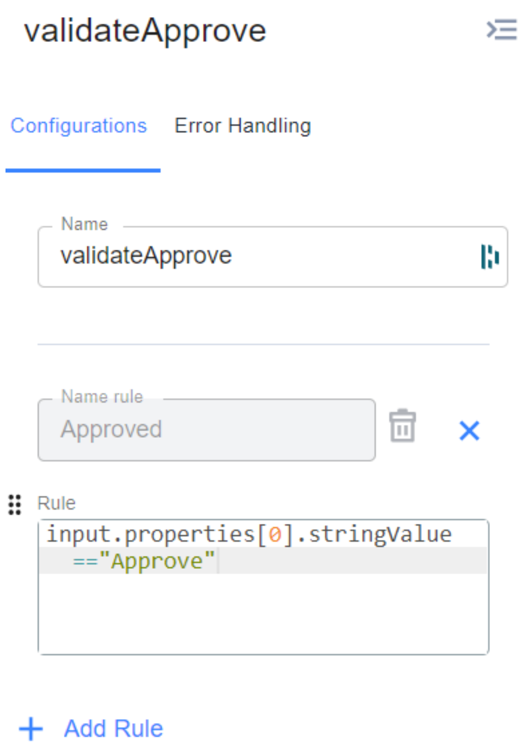
Signal Type: WO Updated

Custom Filter

```
signal.workOrder.name.startsWith
  ("WO")
```

Block flow until reception

For the block “Choice” with the name ***validateApprove***:


validateApprove

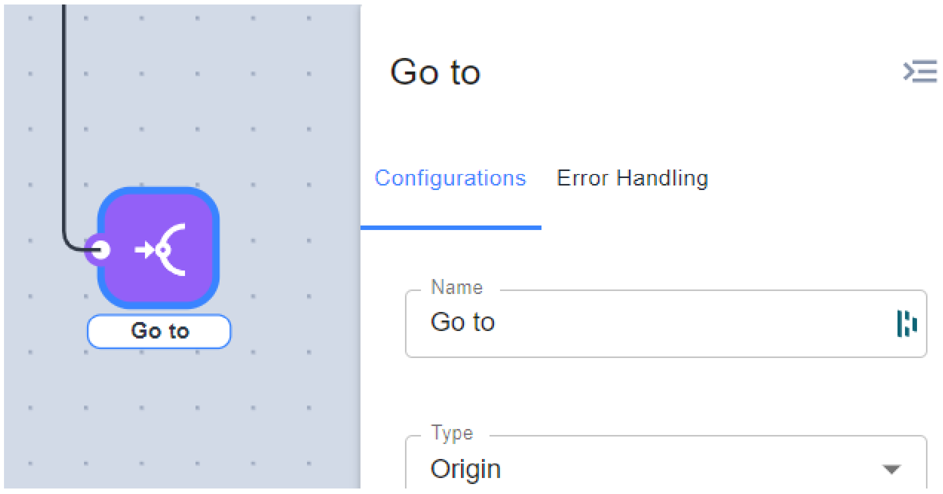
Name: validateApprove

Name rule: Approved

Rule

```
input.properties[0].stringValue
  =="Approve"
```

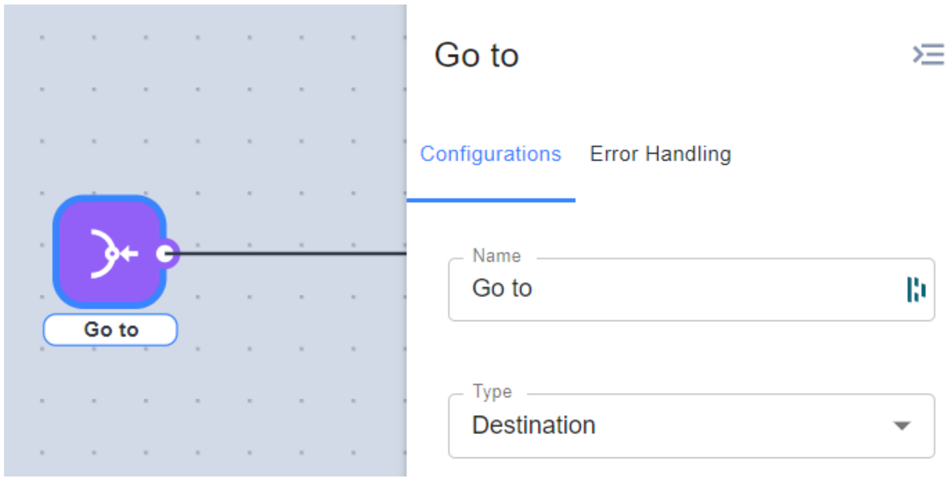
For the block “Go to” for the **origin**:



Name: Go to

Type: Origin

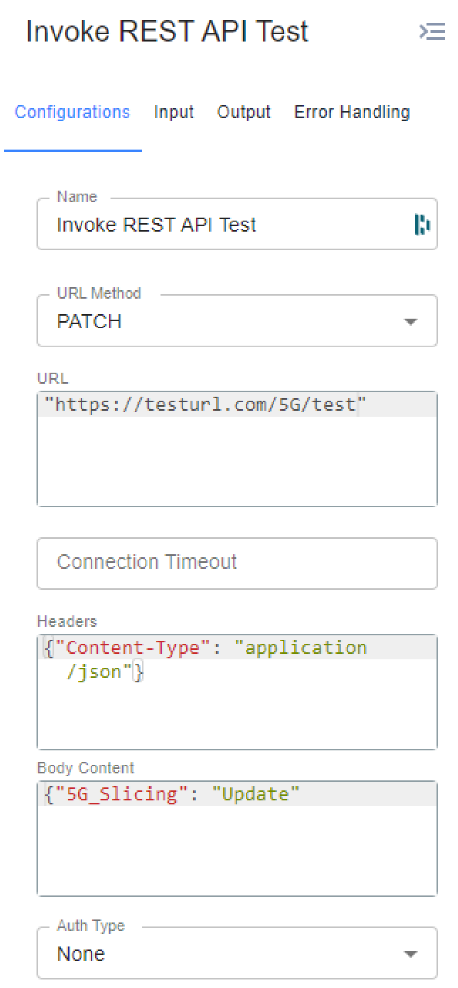
For the block “Go to” for the **Destination**:



Name: Go to

Type: Destination

For the block “Invoke Rest API” with the name ***Invoke REST API Test***:


Invoke REST API Test

Configurations Input Output Error Handling

Name: Invoke REST API Test

URL Method: PATCH

URL
```
"https://testurl.com/5G/test"
```

Connection Timeout

Headers
```
{"Content-Type": "application
  /json"}
```

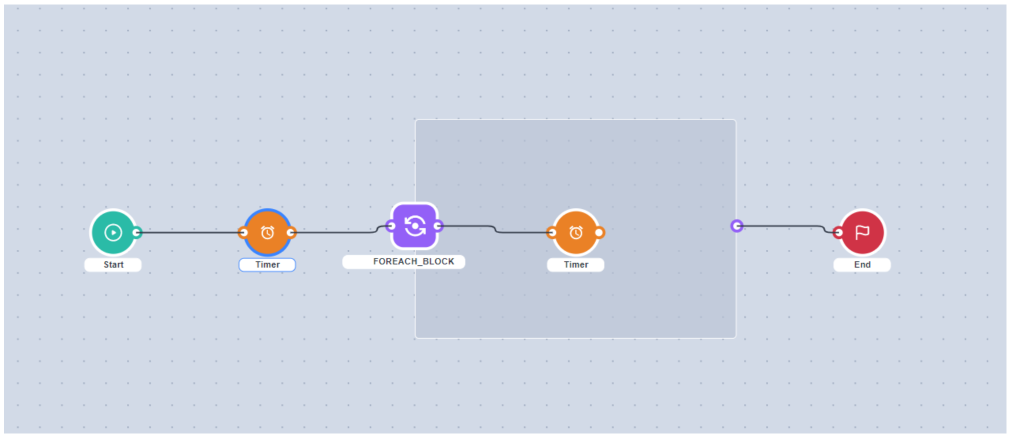
Body Content
```
{"5G_Slicing": "Update"
```

Auth Type: None

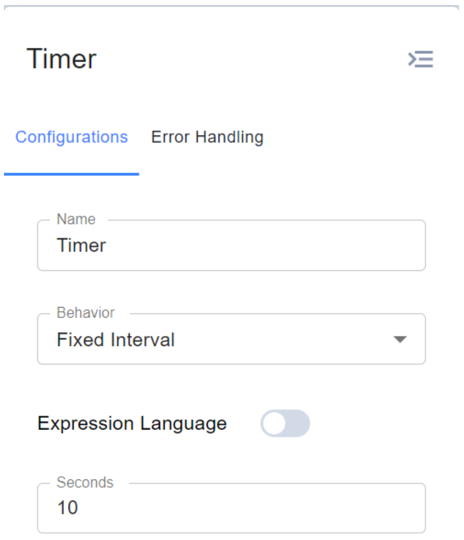
Example (B):

The next example shows a simple use of the For Each Block. When the flow is instantiated, the instantiation includes an array as parameter, with **values** as the parameter name:[value1, value2, value3].

The next picture shows the diagram of the flow:



The idea is that the first timer will hold de flow for 10 seconds, according with the block configuration:


Timer

Configurations Error Handling

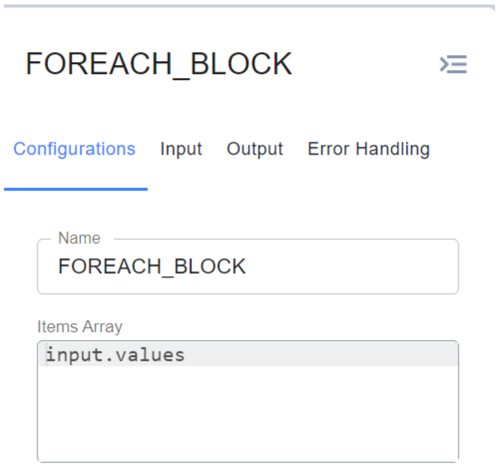
Name: Timer

Behavior: Fixed Interval

Expression Language

Seconds: 10

Next, the for each block has defined as its items Array : input.values, that corresponds with the array that was sent as parameters when flow is instantiated.


FOREACH_BLOCK

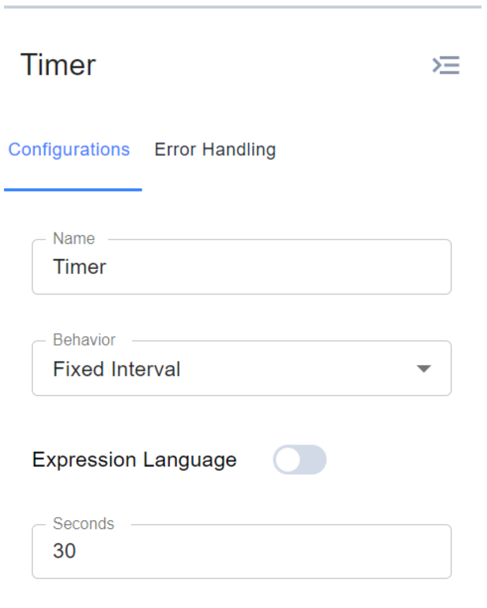
Configurations Input Output Error Handling

Name: FOREACH_BLOCK

Items Array

```
input.values
```

With this configuration, the for each will iterate 3 times and will send each of the list values as input parameter for the second timer, so the second timer will hold the flow for 30 seconds each round, for a total 90 seconds, according with the block config shown in the next image:



Name: Timer

Behavior: Fixed Interval

Expression Language: off

Seconds: 30

NTT DATA
ENVIRONMENT PROJECT
DATA ENCODING
FUTURE
AT HEART