Topic updated on September 21, 2023

Defining a trigger

Following are the trigger components used while defining a trigger:

- The trigger's event type
- The trigger's local variables, static variables, macros and event variables
- The trigger's settings
- The trigger's actions, including the success and failure routes between actions.

When you define a trigger, you name the trigger, identify the event type (Data, Schedule, On-Demand and so on), define the event parameters, and then configure one or more actions. You will be able to Validate the trigger to check for correctness and completeness. When the trigger is saved, it is written to an internal database file on the node.

It is possible to edit, validate and save the trigger definition multiple times as the trigger's application logic is defined. Use the trigger report to generate a report of the trigger's execution to understand the trigger's execution progresses through its actions along with the actions success and failure routes.

Defining an example trigger

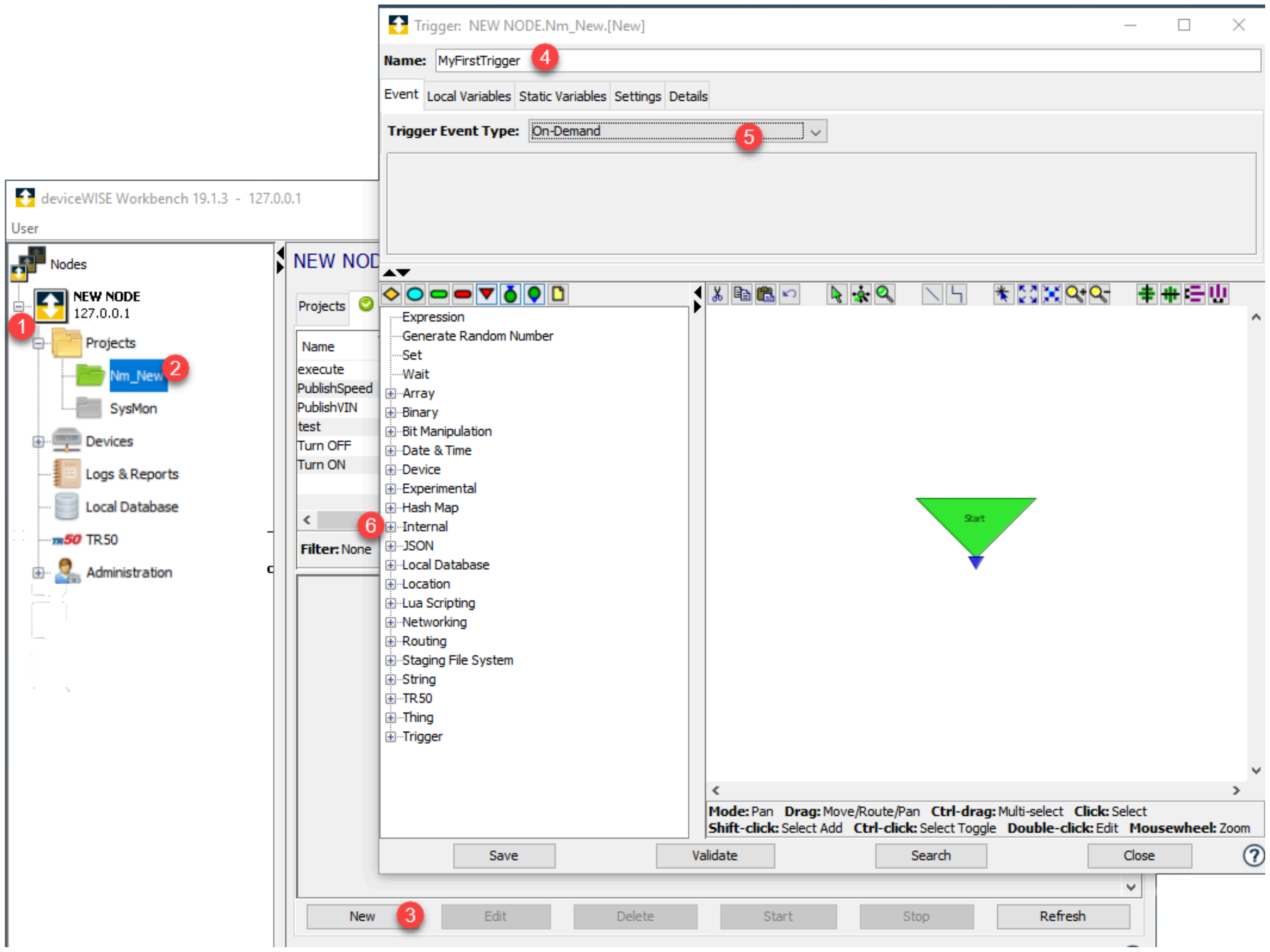
This example will quickly step you through the defining and execution of a sample trigger. The concept and reference information for each of the trigger components is in their specific sections, so all of the details and variations will not be covered in this quick example.

Adding actions to the Canvas

1. Select and expand the node where the trigger will be defined and executed.
2. Click the project where the trigger will reside.

   This example assumes that you will use the project defined in the Projects.
3. Click the **New** button at the bottom of the right hand pane of the project's tab to start the definition of a new trigger.

The New trigger window appears.



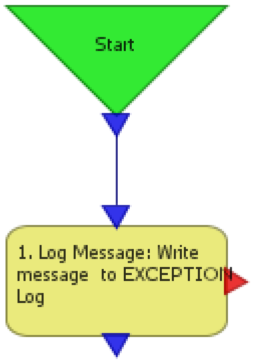
4. Enter *MyFirstTrigger* for the **Name**.

5. Select On-demand for the **Trigger Event Type**.

6. From the left hand pane list of actions, expand the **Internal** category

7. Click and place the **Log Message** action in the canvas and place it below the **Start** block

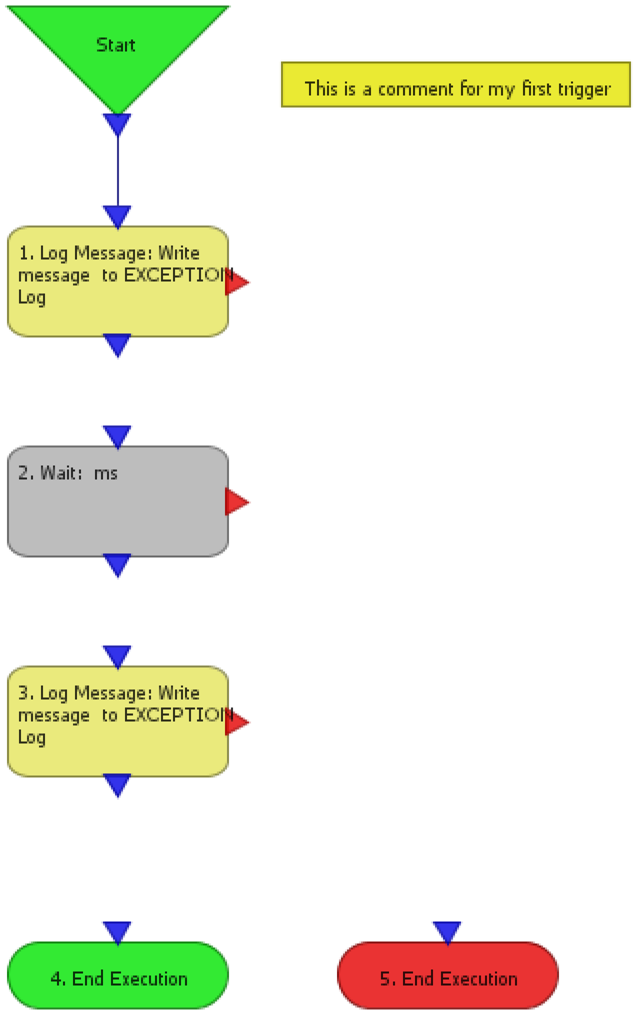
   Alternatively, you can drag (click and hold) the action and then position its location and drop (release the mouse button).



8. From the left hand pane, click**Wait** action and position it on the Canvas below the **Log Message** action.

9. From the left hand pane, click **Log Message** action and position it below the **Wait** action.
10. From the tool bar above the left hand pane, select the **End Execution (Success)** action and position it on the Canvas below the second **Log Message** action.
11. From the tool bar above the left hand pane, select the **End Execution (Failure)** action and position it on the Canvas to the right of the **End Execution (Success)** action.
12. From the tool bar above the left hand pane, select the **Comment** block and position it on the Canvas to the right of the **Start** block. Enter *This is a comment for my first trigger* into the comment block

The Canvas should look similar to this:


Start
This is a comment for my first trigger
1. Log Message: Write message to EXCEPTION Log
2. Wait: ms
3. Log Message: Write message to EXCEPTION Log
4. End Execution
5. End Execution

Entering the parameter details for each action

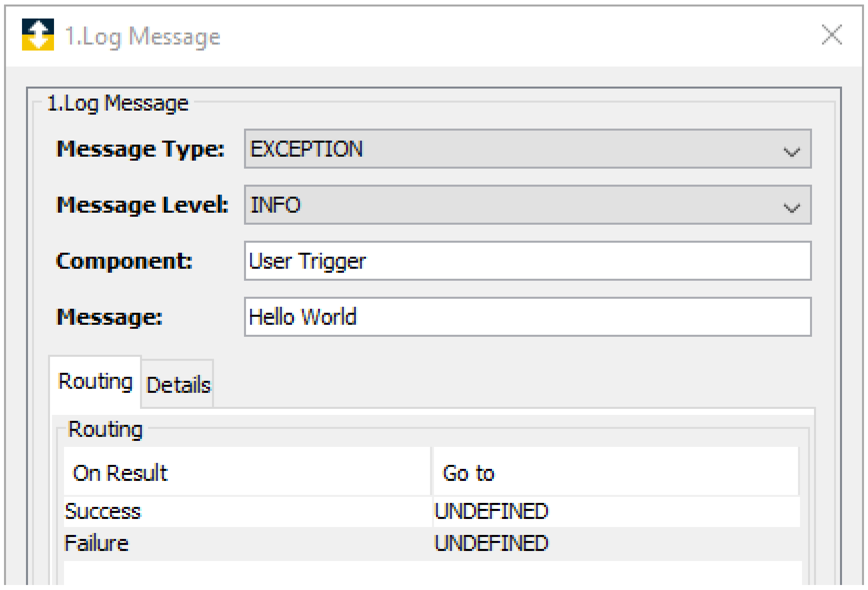
The two Log Message actions and the Wait action have parameters that need to be entered.
To enter the parameter information for an action, double click the action in the Canvas area.

For the actions, do the following:

1. Double click the first **Log Message** action.

   The action's parameters details are displayed in a window.

2. In the **Message** parameter, delete the $(Message) text and enter *Hello World*.



1.Log Message

1.Log Message

Message Type: EXCEPTION

Message Level: INFO

Component: User Trigger

Message: Hello World

Routing

On Result	Go to
Success	UNDEFINED
Failure	UNDEFINED

3. In the **Details** tab, enter a comment for this action.

4. Close the window by selecting the red close icon (X) in the upper right of the window.

5. Double click the **Wait** action.

6. In the Time to Wait(ms) parameter, enter *5000* for 5000 milliseconds (5 seconds).

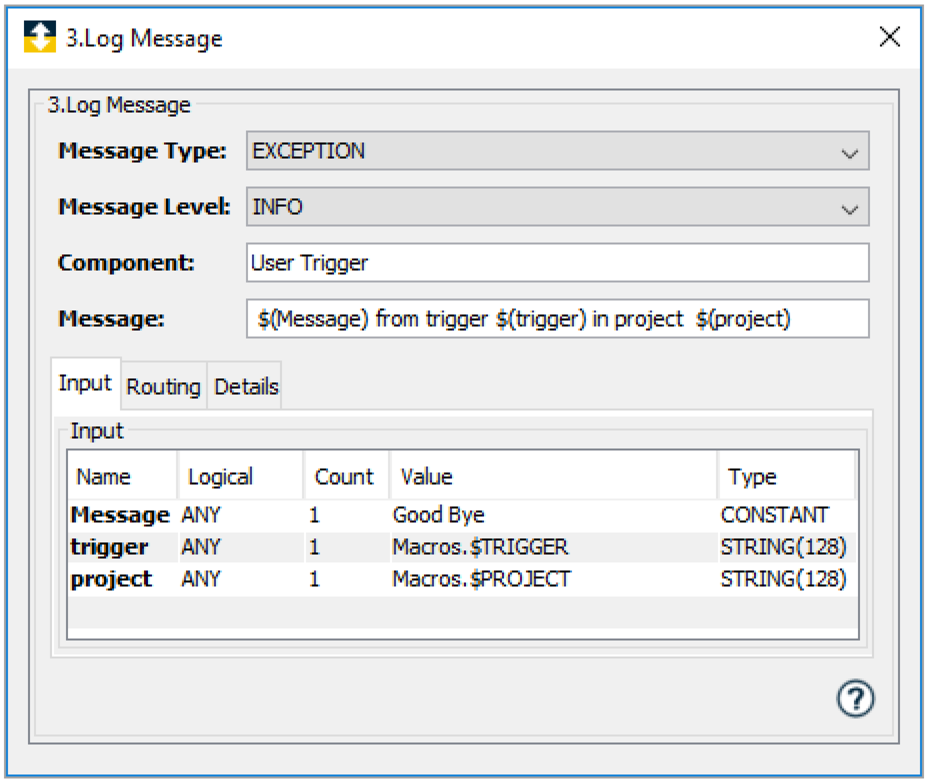
7. Close the window by selecting the red close icon (X).

8. Double click the second Log message action.

9. In the **Message** parameter, enter *$(Message) from trigger $(trigger) in project $(project)*.

   Each of the substitution parameters enclosed in a $( ) becomes an input parameter on the Input tab.

10. Enter variables for each of the input parameters as follows:


3.Log Message

3.Log Message

Message Type:	EXCEPTION
Message Level:	INFO
Component:	User Trigger
Message:	$(Message) from trigger $(trigger) in project $(project)

Input

Name	Logical	Count	Value	Type
Message	ANY	1	Good Bye	CONSTANT
trigger	ANY	1	Macros.$TRIGGER	STRING(128)
project	ANY	1	Macros.$PROJECT	STRING(128)

11. Close the window by selecting the red close icon (X).

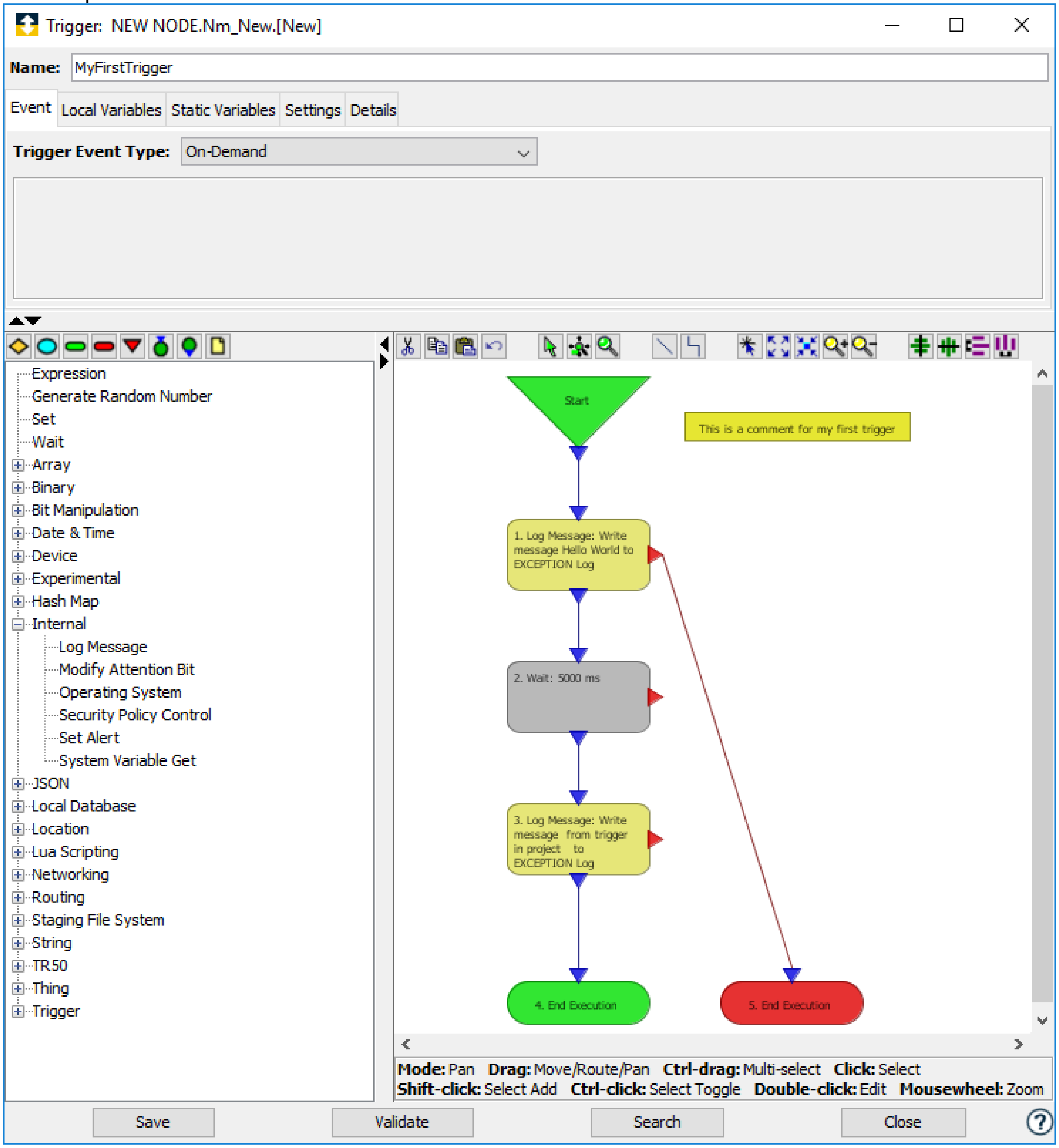
Specifying the routing for each action

The actions and blocks need routings for each of the input and output ports.

For the routings, do the following:

1. Click and hold the output port at the bottom of the **Start** block, then drag the mouse cursor to the input port at the top of the first **Log Message** action.
   When the input port turns from blue to yellow, release the mouse button.
   A route connection line should be drawn between the **Start** block and the first **Log Message** action.
2. Draw a connection line from the bottom of the first **Log Message** action to the top of the **Wait** action.
3. Draw a connection line from the bottom of the **Wait** action to the top of the second **Log Message** action.
4. Draw a connection line from the bottom of the second **Log Message** action to the top of the **End Execution (Success)** action.
5. Finally draw a connection line from the red side exit of the Wait action to the top of the **End Execution (Failure)** action.

The completed Canvas should look similar to this:



Trigger: NEW NODE.Nm_New.[New]

Name: MyFirstTrigger

Trigger Event Type: On-Demand

Validating and saving the trigger

1. Click **Validate** at the bottom of the right hand pane. The validation function will check each action for correctness and completeness.

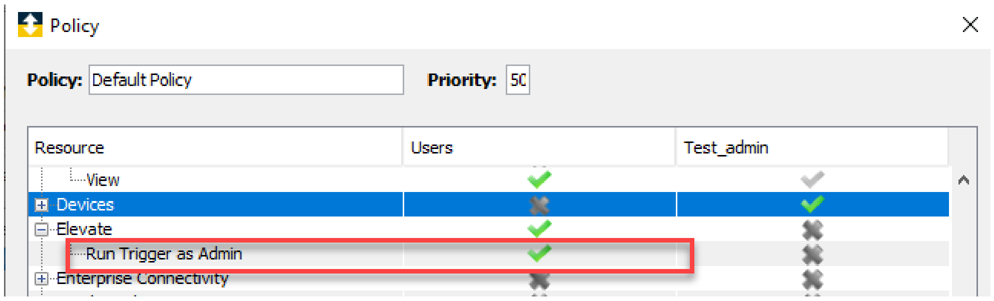
   If errors are found they will be displayed in a window for review and correction. Your trigger should validate successfully, if not review any errors and make the corrections.

2. Click **Save** at the bottom of the right hand pane to save the trigger definition and close the trigger Canvas Editor.

   The trigger will be listed in the project tab's list of triggers in a **Stopped** state

Starting and executing the trigger

1. Click the trigger you want to start.
2. Do one of the following:
   1. Click **Start** or right-click on the trigger and then select the **Start** option.
   2. To run the trigger as an admin - Right click on the trigger and select **Start as Admin** to run the trigger as admin.

      > **Note**
      >
      > In order to run a trigger as an admin, the admin needs to provide your profile necessary access. In the Policy, **Elevate > Run Trigger as Admin** should be enabled for you to start the trigger as an admin.
      >
      > 
      >
      > Policy
      >
      > Policy: Default Policy Priority: 5C
      >
      > | Resource | Users | Test_admin |
      > |---|---|---|
      > | View | ✓ | ✓ |
      > | Devices | ✖ | ✓ |
      > | Elevate | ✓ | ✖ |
      > | Run Trigger as Admin | ✓ | ✖ |
      > | Enterprise Connectivity | ✖ | ✖ |
      >
      > For more information on setting up the policies, see Policies tab.

3. The trigger's state should be **Started** and the trigger's status should be **Loaded**.

   If the the trigger's status is **Unloaded**, then the project needs to be started. To do this, right-click on the project's tab to display a pop-up menu and then select the **Start** option. The project should be **Started** and the trigger should be **Started** and **Loaded**.
4. Right-click on the trigger, and then select the **Fire Trigger** option.
5. The trigger (an On-Demand event type) will be executed.

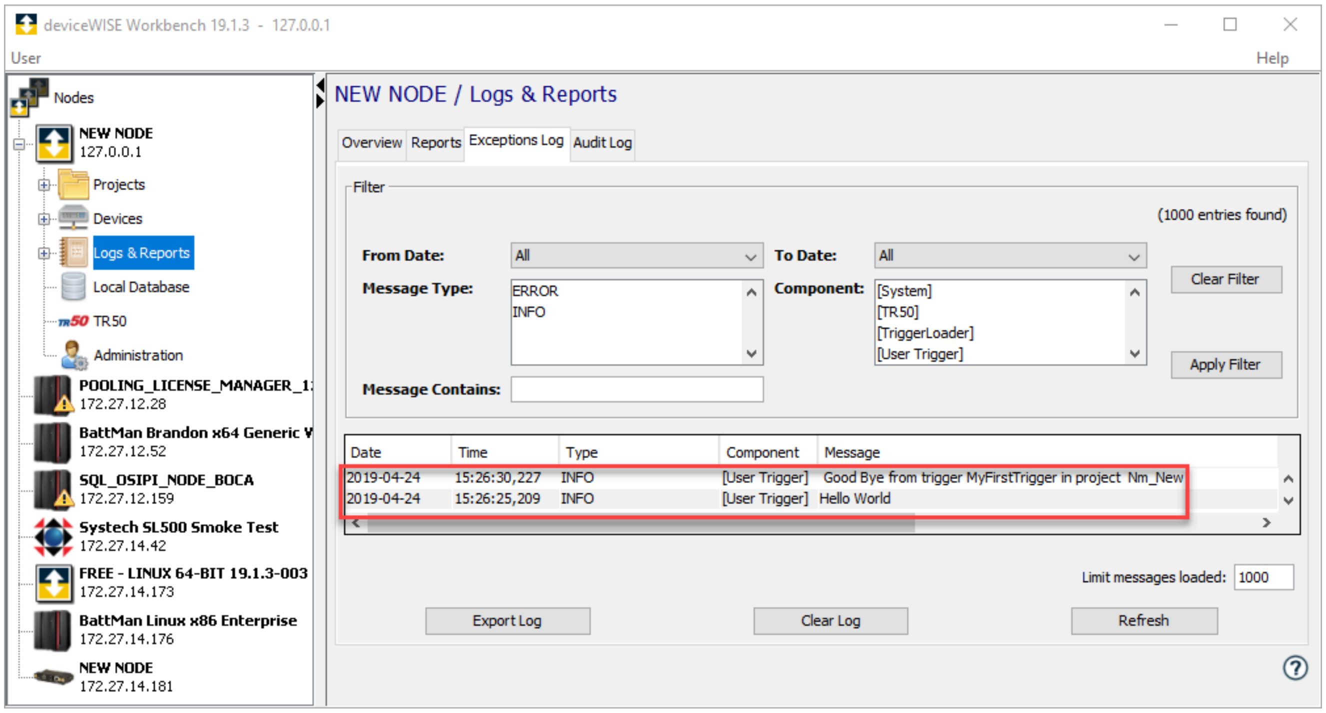
   You may see the **In Progress** count change to 1, and then you should see the **Successes** count change to 1.
6. Select the **Refresh** button a few times until you see the completion of the trigger's execution.
7. You will notice the updates to the **Last Triggered** and **Avg Time (ms)** values.

8. Since this trigger added messages to the Exceptions log, we will view it to see the messages.

9. In the left hand pane, select the **Logs & Reports** icon for this node.

10. In the right hand pane select the **Exceptions Log** tab.

    The Exceptions Log messages are displayed, including the two from this trigger:


deviceWISE Workbench 19.1.3 - 127.0.0.1
User
Help
Nodes
NEW NODE 127.0.0.1
Projects
Devices
Logs & Reports
Local Database
TR50
Administration
POOLING_LICENSE_MANAGER_1 172.27.12.28
BattMan Brandon x64 Generic V 172.27.12.52
SQL_OSIPI_NODE_BOCA 172.27.12.159
Systech SL500 Smoke Test 172.27.14.42
FREE - LINUX 64-BIT 19.1.3-003 172.27.14.173
BattMan Linux x86 Enterprise 172.27.14.176
NEW NODE 172.27.14.181
NEW NODE / Logs & Reports
Overview
Reports
Exceptions Log
Audit Log
Filter
Clear Filter
Apply Filter
Export Log
Clear Log
Refresh

(1000 entries found)

From Date: All **To Date:** All

Message Type: ERROR, INFO

Component: [System], [TR50], [TriggerLoader], [User Trigger]

Message Contains:

Date	Time	Type	Component	Message
2019-04-24	15:26:30,227	INFO	[User Trigger]	Good Bye from trigger MyFirstTrigger in project Nm_New
2019-04-24	15:26:25,209	INFO	[User Trigger]	Hello World

Limit messages loaded: 1000

11. The Audit Log can also be viewed to see the types of auditing messages that are logged by the system when events occur.

Tip

- It is possible to use the trigger List Editor instead of the trigger Canvas Editor.
- The example trigger's logic is simple, more complex triggers could include access to device variables, access to enterprise applications, interaction with the deviceWISE Cloud and much more.

That completes this example trigger. The details of each trigger component are described in the following sections.

What's Inside

Trigger event type

Trigger local variables, static variables, macros and event variables

---