

SIEMENS

Insights Hub

Visual Flow Creator

System Manual

04/2024

[Introduction to Visual Flow Creator](#) **1**

[Process Industrial IoT data in Visual Flow Creator](#) **2**

[User interface Visual Flow Creator](#) **3**

[User rights in Visual Flow Creator](#) **4**

[Visual Flow Creator basics](#) **5**

[Nodes in Industrial IoT](#) **6**

[Dashboard in Visual Flow Creator](#) **7**

[Optimizing Visual Flow Creator flows](#) **8**

[Subflows](#) **9**

[Custom nodes](#) **10**

[Example for execution time calculation in VFC](#) **11**


[Appendix](#) **12**


[Glossary](#) **13**


Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.

 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.

 CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1. Introduction to Visual Flow Creator	6
1.1. Introduction to Visual Flow Creator.....	6
2. Process Industrial IoT data in Visual Flow Creator	8
2.1. Process Industrial IoT data in Visual Flow Creator.....	8
3. User interface Visual Flow Creator	9
3.1. User interface of Visual Flow Creator.....	9
4. User rights in Visual Flow Creator	26
4.1. User rights.....	26
5. Visual Flow Creator basics	28
5.1. Using nodes.....	28
5.2. Create flows.....	39
6. Nodes in Industrial IoT	45
6.1. Nodes in Industrial IoT.....	45
6.2. Industrial IoT node library.....	46
6.3. Usages of function nodes.....	62
6.4. Usages of function nodes.....	62
6.5. Using context nodes.....	81
6.6. Using parquet node.....	84
6.7. Usages of analytics nodes.....	86
6.8. Using analytics nodes.....	88
6.9. Storage node library.....	90
6.10. Array node library.....	93
6.11. Simple Anomaly node library.....	97
6.12. Integrated Data Lake nodes.....	106
6.13. SDI nodes.....	110
6.14. http-in node.....	133

6.15. Pump-Simulation node.....	137
6.16. Usage of MQTT nodes.....	144
6.17. Link Nodes.....	150
6.18. Usage of Business Intelligence nodes.....	156
6.19. Simatic Notifier node.....	159
6.20. Predictive Learning (PRL) nodes.....	166
6.21. Work Order Nodes.....	170
6.22. MindConnect nodes.....	179
7. Dashboard in Visual Flow Creator.....	187
7.1. User Interface of Dashboard nodes output.....	187
7.2. Usages of standard dashboard nodes.....	188
7.3. Dashboard layouts.....	230
7.4. Using standard dashboard nodes.....	236
7.5. Application examples on node properties.....	238
7.6. Example for using Vega dashboard node.....	241
7.7. Usage of IIoT Dashboard nodes.....	247
7.8. Using IIoT Dashboard nodes.....	251
7.9. Implement IIoT Map node - Example.....	254
7.10. Integrating Insights Hub Monitor plugin - Example.....	257
7.11. Usage of Dashboard Viewer.....	259
7.12. Adding links to Dashboard Viewer.....	262
8. Optimizing Visual Flow Creator flows.....	265
8.1. Optimizing Visual Flow Creator flows.....	265
9. Subflows.....	266
9.1. Subflows.....	266
10. Custom nodes.....	272
10.1. Custom nodes.....	272

11. Example for execution time calculation in VFC..... 282

11.1. Example for execution time calculation in VFC..... 282

12. Appendix..... 284

12.1. Appendix..... 284

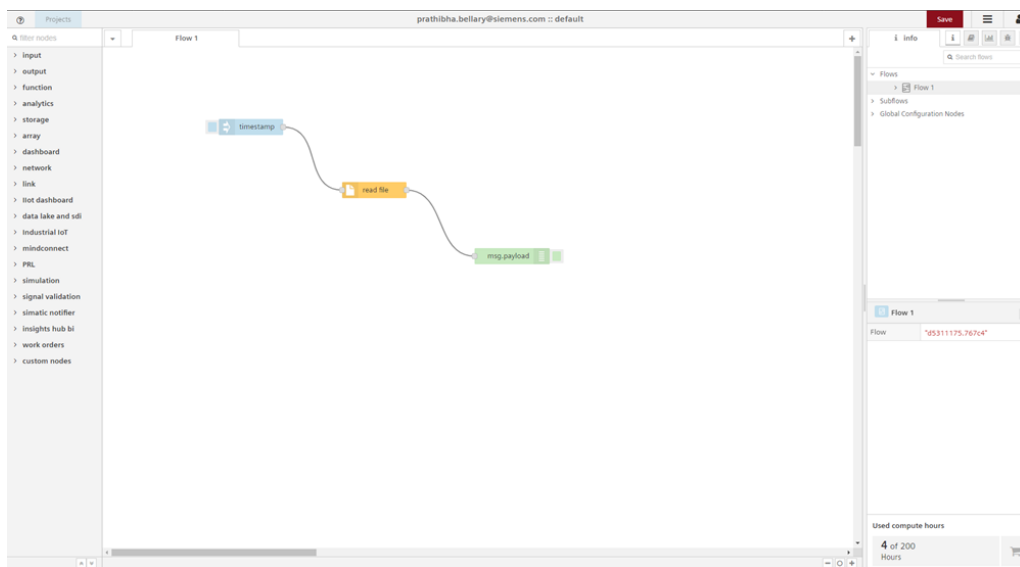
13. Glossary..... 287

13.1. Glossary..... 287

Introduction to Visual Flow Creator

1.1 Introduction to Visual Flow Creator

Visual Flow Creator is an application used to create a visual representation of data processing workflows.



Visual Flow Creator is a component in Industrial IoT. For more information, refer to [Additional information](#).

Visual Flow Creator is a software for visual data flow programming in IoT (Internet of Things). Nodes are predefined blocks of functionality. You can interlink and add nodes in the working area and combine them to produce a "flow". For more information, refer to [Create flows](#).

With Visual Flow Creator, IoT applications can be easily implemented in a short amount of time. The focus is on development of prototypes. The graphical user interface also makes it easier to get started with IoT programming. Knowledge of JavaScript is advantageous for the use of Visual Flow Creator.

Visual Flow Creator provides the following functionalities:

Easily access data by using Industrial IoT nodes

- Data flow processing
- Creating own evaluations with Industrial IoT data
- Preparing collected data
- Calculating KPIs
- Creating REST APIs

- Establishing a link to multiple apps
- Creating custom rules
- Triggering notifications at certain events
- Creating simple dashboards
- Inserting own JavaScript code in "functions" nodes. For more information, refer to [Function node library](#).



Data processing in Visual Flow Creator

- Users of Visual Flow Creator are responsible for checking all external resources and sources.
- Visual Flow Creator is suitable only to a limited extent for processing of continuous information.
- Error-free execution of flows is dependent on the Industrial IoT services.

Additional information

You can find additional information about VFC using the following links:

- Visual Flow Creator examples: [vfc-examples](#).
- Visual Flow Creator tutorial: [VFC Introduction](#).

You can find additional information about Node-RED using the following links:

- [noderedguide.com](#)
- [www.nodered.org](#)

Process Industrial IoT data in Visual Flow Creator²

2.1 Process Industrial IoT data in Visual Flow Creator

Available data from Industrial IoT

You have the option of using and processing the following data from Industrial IoT.

- Events
- Files
- Time series data
- Creating dashboards
- Integrated Data Lake

Specific Industrial IoT nodes are available for processing data. For additional information on Industrial IoT nodes, refer to [Using Industrial IoT nodes](#).

User interface Visual Flow Creator

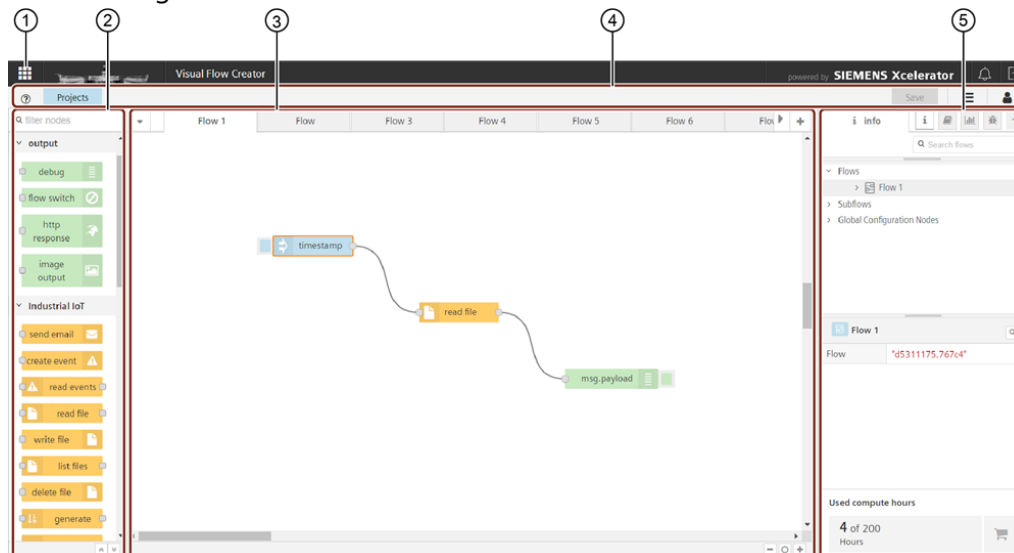
3

3.1 User interface of Visual Flow Creator

You can operate Visual Flow Creator using the "main navigation", "node palette", "working area" and "info window".

Start screen

The following screenshot shows the different elements of the Visual Flow Creator user interface:

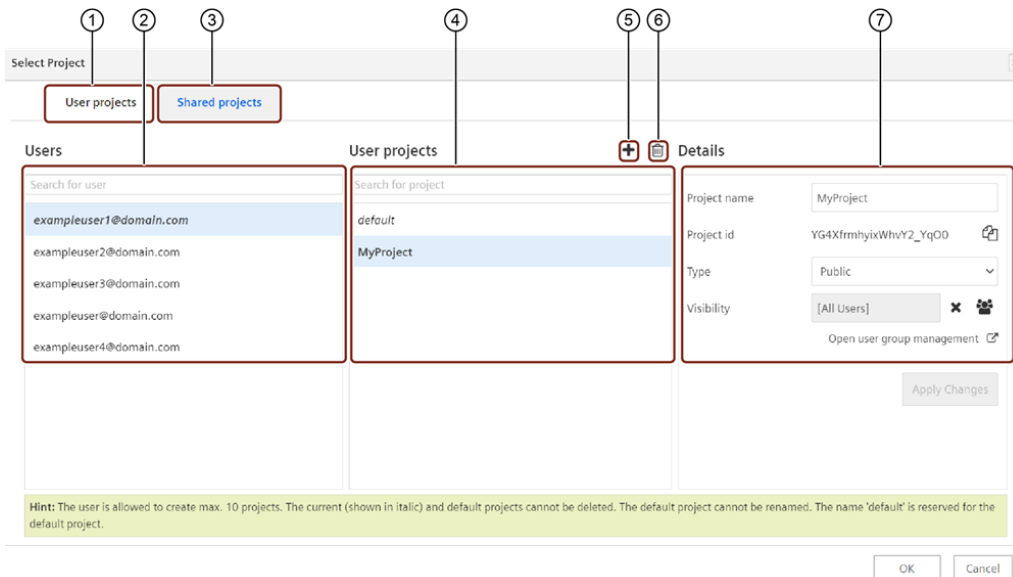


- ① Opens the Launchpad
- ② Node palette
- ③ Working area
- ④ Main navigation
- ⑤ Info window/ dashboard window/ debug window

Projects

Projects allows you to separate your flows from other user flows and organize your flows into small logical coherent pieces based on your working tasks. You can create, rename, search and delete the "Projects" from the environment. A default project is already available for each user. You can also access the other user flows by selecting user from the list of "Projects". You can access "Projects" from the top left corner of the UI:

3.1 User interface of Visual Flow Creator



- ① User projects tab
- ② Search and select the user from "Users" list
- ③ Shared projects tab
- ④ Search and select the project from "User projects" list
- ⑤ Creates a new project
- ⑥ Deletes the selected project
- ⑦ Displays the selected project details

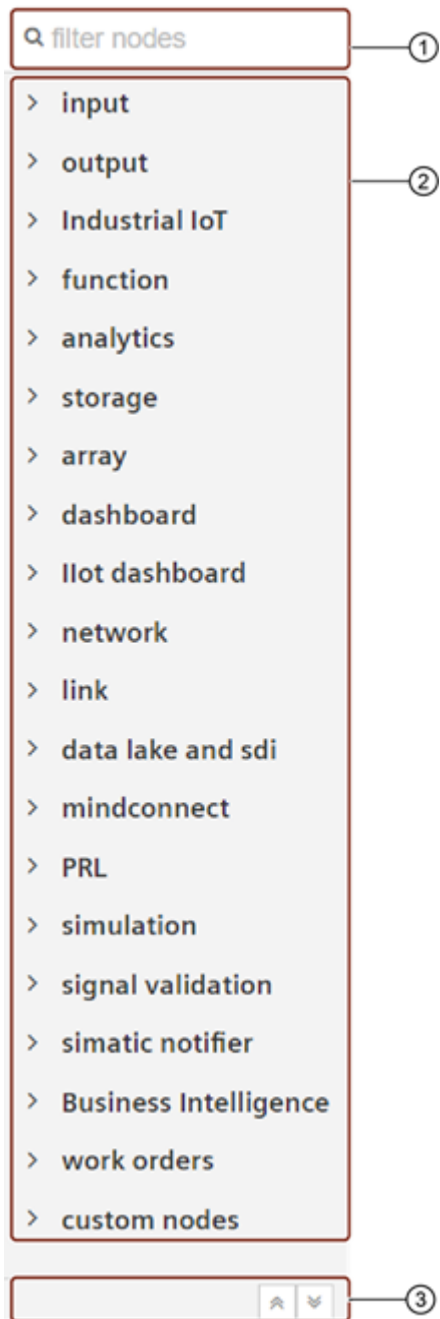
- Up to 10 projects can be created per user including the "default" project.
- "Default" project cannot be renamed or deleted.
- If the user selects the project type as "Public" for the particular project, then it will be visible to the other users of the environment.
- If the user selects the project type as "Private" for the particular project, then it will not be visible to the other users of the environment.

Project Properties	Description
Users	Displays the user's list and allows to search for the required.
User projects	Displays the "default" project and other projects list.
Shared projects	Displays the shared projects to other users of the environment.
Project name	Displays the project name and allows you to rename the project name.
Project id	Displays the "Project id" of the project. "Project id" is useful for accessing API of the particular Project.

Project Properties	Description
Type	<ul style="list-style-type: none"> - Public: This type of project will be visible for all the other users of the environment. You can share public projects to the other user to edit the flows inside the project. - Private: Only visible for the creator and VFC admins of the environment. <p>Note: Admin can change the project details.</p>
Visibility	<p>It allows to add or remove "user group" for the selected project.</p> <p>Note: To add the user group to the selected project, ensure the user group is already created in "Settings" application.</p>
Open user group management	<p>Navigates to the "User groups" feature in Settings application. For more information about managing "User groups" feature, refer to Managing user groups.</p>

Node palette

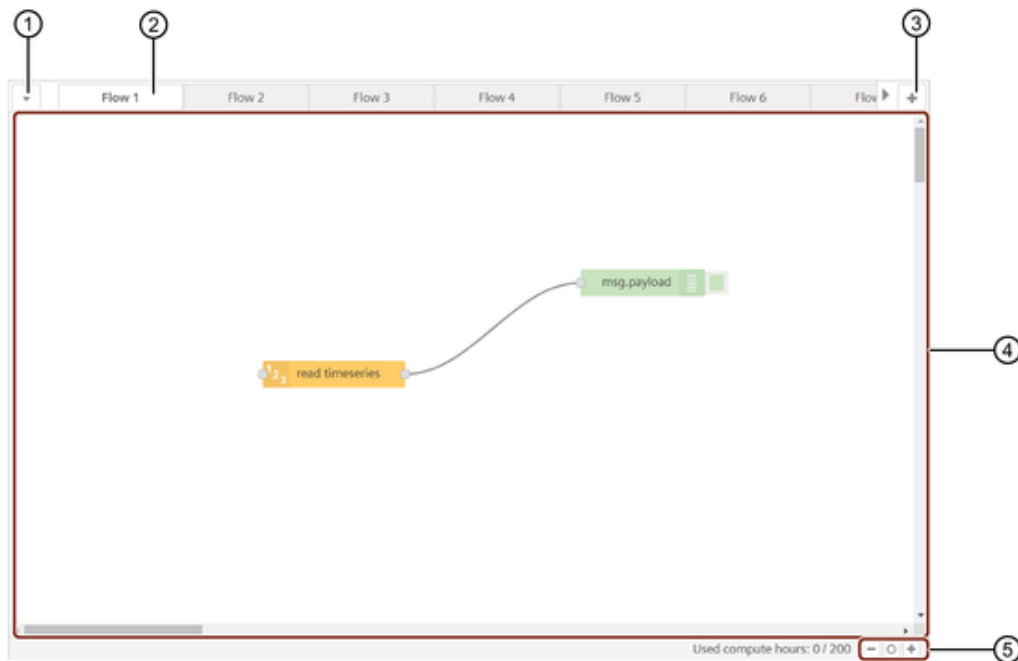
The node palette is located on the left side of the editor. The palette shows you the different types of nodes available in the UI.



- ① Filter the list of nodes
- ② Node palette
- ③ Collapse or expand all categories

Working area

You can drag nodes from the palette and wire them to develop the flows in the working area.

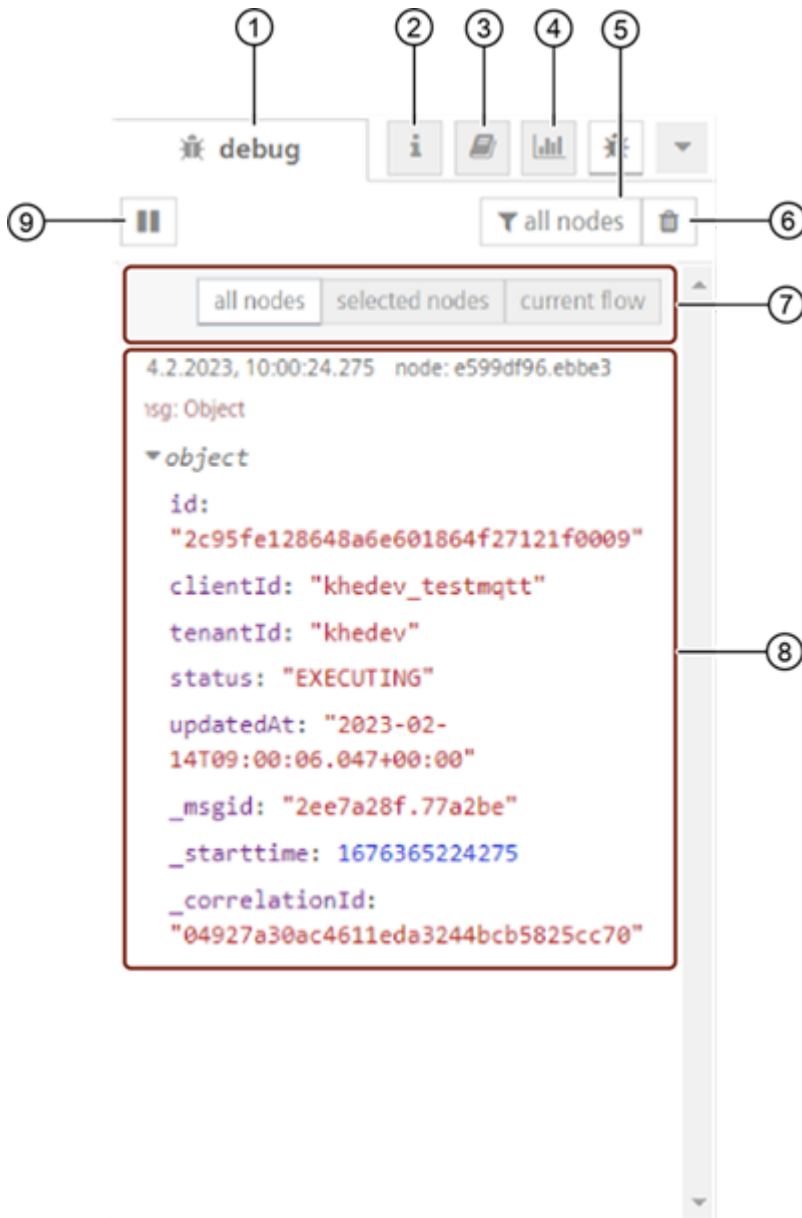


- ① Opens a dropdown in order to quickly navigate through all flows
- ② "Flow" tab with display of user email address
- ③ Creates a new tab. Multiple flows in one tab possible.
- ④ Working area
- ⑤ Zoom function/standard zoom level

Sidebar

"Visual Flow Creator" has the feature to allow you to view the information of the flows in different ways. You can use the sidebar window to configure the nodes and flows and view the results. It has three tabs:

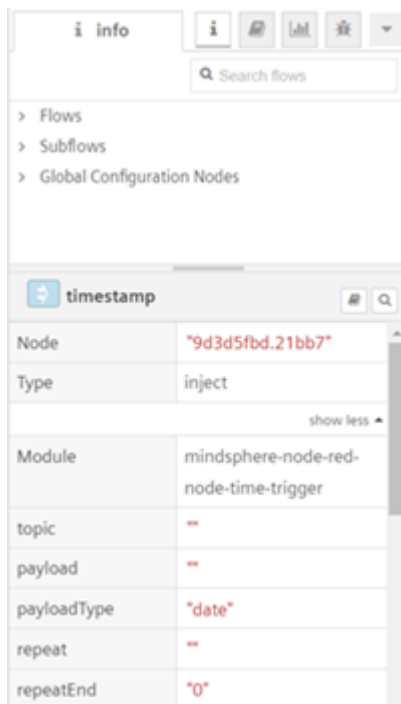
- Info tab: It shows information of the nodes of the current project as well as node properties.
- Help tab: It displays node information about how to perform various functions.
- Dashboard tab: It displays the layout, site and theme of the nodes or flow.
- Debug tab: It displays the debug messages.



- ① Displays debug messages
- ② Displays information about the selected node in the node palette
- ③ Help tab
- ④ Dashboard tab
- ⑤ Filters debug messages
- ⑥ Deletes all debug messages
- ⑦ Filters the options
- ⑧ Log area
- ⑨ Pauses the display of new log messages

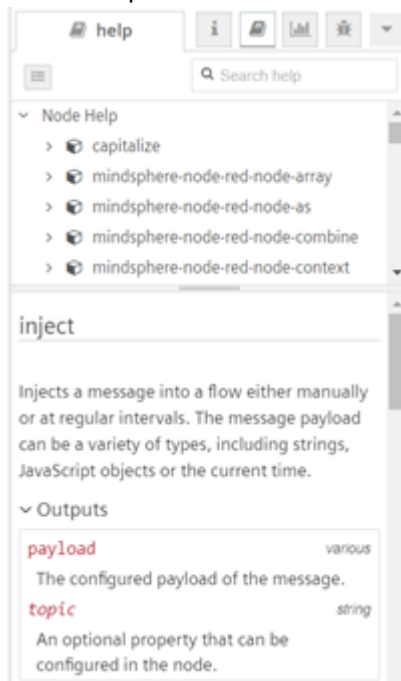
Info tab


The info tab shows the properties of the selected node as well as information on how to use the node. If no node is selected, the info window will show the properties of the current flow displayed in the workspace.

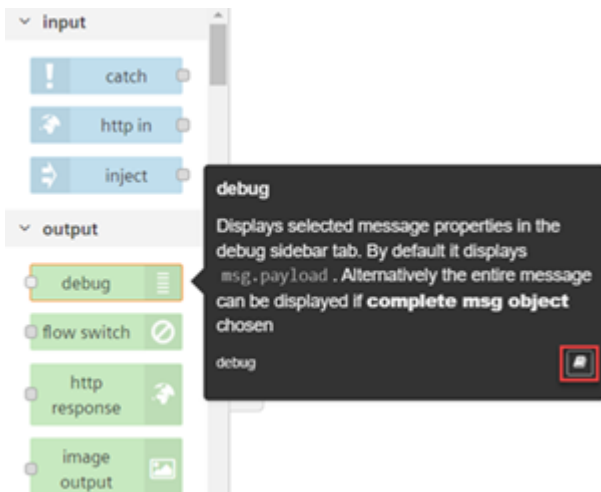


Help tab

The help tab shows the details about how to perform various functions of a selected node. If no node is selected, the help window will show the details of the current selected node displayed in the workspace.

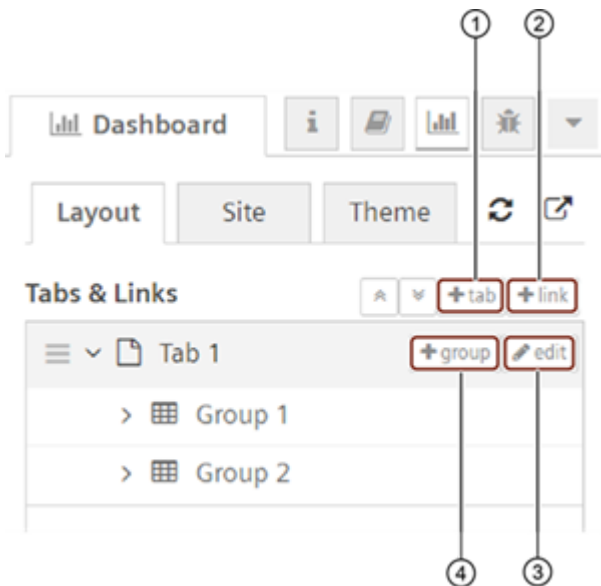


When a node is opted, it displays its description as well as a shortcut key to the help tab. To open the help tab, click .



Dashboard tab

To know the working of the dashboard nodes, it is important to know some terms and their definitions:



- ① Add tabs in the dashboard
- ② Add external web links to display in the dashboard
- ③ Edit tab properties in the dashboard
- ④ Add groups in the tab

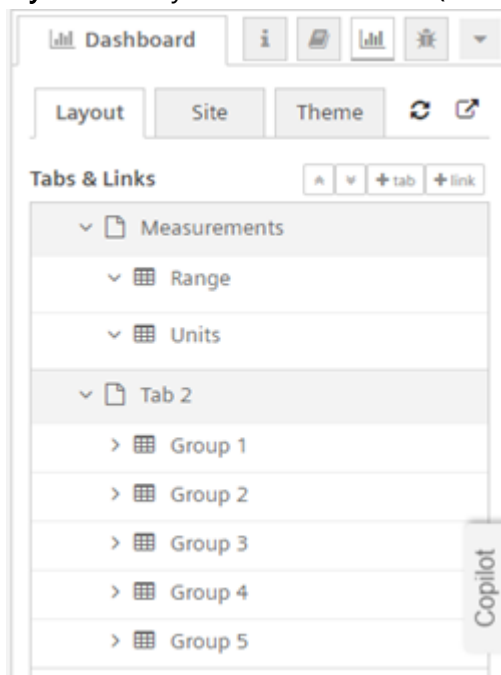
1. **Tabs:** Tab is created under dashboard tab. You can add groups, edit groups and edit the group properties under the tab. You can add or edit multiple groups in the tab.



2. **Group:** A group is a container for multiple dashboard elements. You can add or edit a group for each flow and you can develop multiple groups under the tab.


3. **Links:** You can create links to other web pages in the dashboard panel of the sidebar tab.

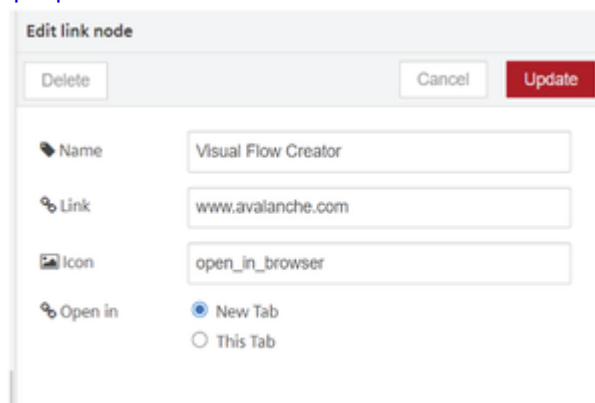
The dashboard window has three panels:

- **Layout:** The layout defines the tabs (with containing dashboard elements) and links.



After creating tabs, you can add multiple groups in each of the tabs. The tabs allow you to reorder the tabs and groups. To add a new group, click the  icon in the created tab. For editing any group, use the  icon. The tabs and groups are expandable for editing as well as collapsible.

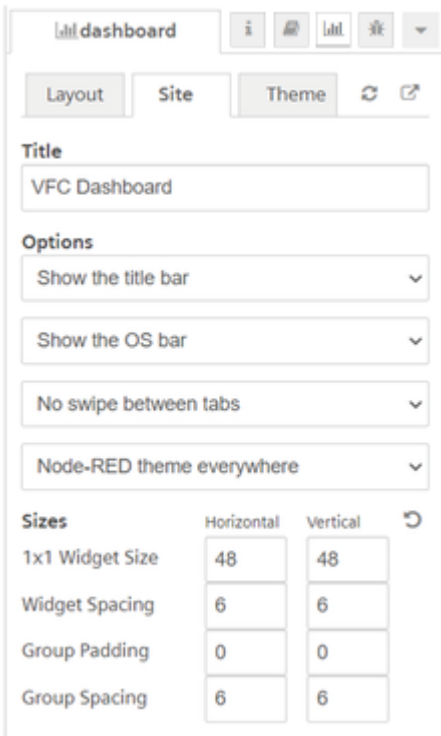
Similarly, you can define the properties of the created links by clicking  the button. For configuring a dashboard node, the groups and tabs created can be used to define the dashboard node properties. For more information, refer to [Configure dashboard node properties](#)



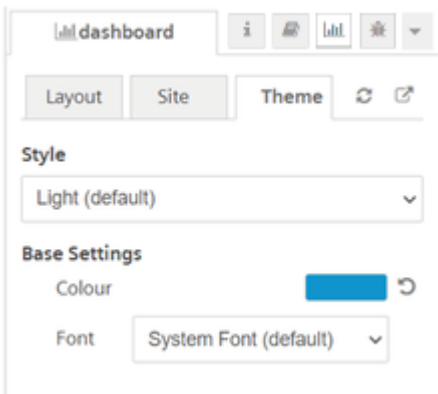
Site: The site panel allows you to configure the user interface behavior of the dashboard


- site.

You can also set the title of your dashboard here.



- **Theme:** You can set the themes, style, color and font of the UI in the dashboard sidebar.



 The creation of groups / tabs / links are applicable only with the dashboard nodes.

For more information about creating and working of dashboard nodes, refer to [Dashboard nodes](#).

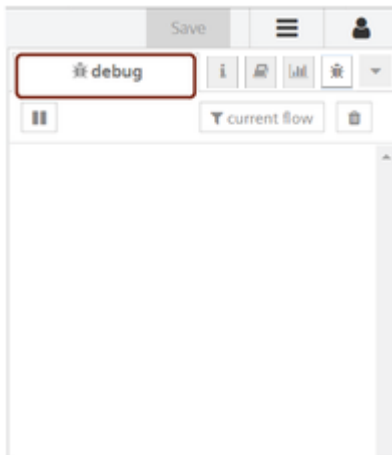
Debug tab

You can inject the node and message payload is displayed in the Debug window. This window display debug information for:

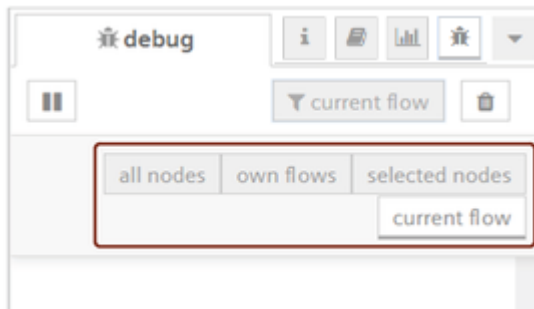
- IIoT
- Function
- Analytics

- Storage
- Array
- Simple anomaly
- Dashboard
- Custom

Click the "debug" window to view the message payload.




The "debug window" displays message payload of the nodes in the flows. The messages can be filtered using the node filter button. The node panel has the following filter options:



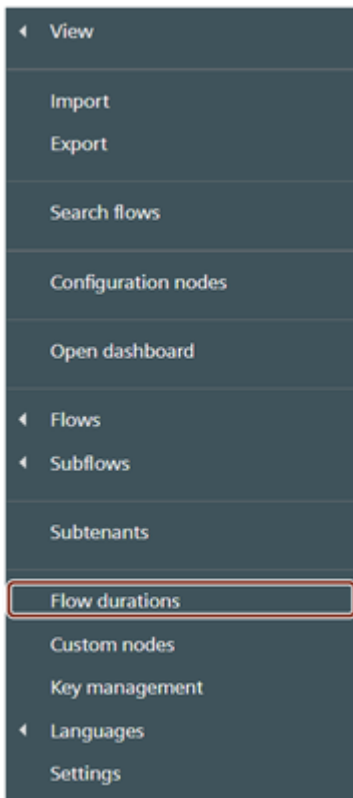
- All nodes: Displays all the messages.
- Own nodes: Displays the messages of the current user's nodes.
- Selected nodes: Displays messages only of the selected nodes from all the available nodes.
- Current flow: Displays the messages of the current working flow.

If there are large number of messages, the users have the possibility to filter them, in order to quickly find important debug information.

Flow durations

"Flow durations" displays the duration of the triggered flows in Visual Flow Creator by different users with time, user details and node information. Click  to access the "Flow durations":

3.1 User interface of Visual Flow Creator



You can find the list of triggered flows in Visual Flow Creator. You can filter by the user and redirect to the flows in Visual Flow Creator by clicking on the active node.

Flow Durations (current month, shown first 100 entries)

Filter:


7c84cc3f.43a234	default	markus.zechner@siemens.com		00:00:01.578
5bfe10b6.03087	current	stanislav.truntaev@siemens.com		00:00:00.883
74ab78b9.420df8	default	bianca.stadler@siemens.com		00:00:00.554
f8c95a33.40b678	default	bianca.stadler@siemens.com		00:00:00.444
3a342181.1b67ae	default2	hauke.nagel@siemens.com		00:00:00.377
494c498f.150dc8	current	stanislav.truntaev@siemens.com		00:00:00.320
9f6a3b9e.32aec8	default	bianca.stadler@siemens.com		00:00:00.302
7fbd6e54.79127	default2	hauke.nagel@siemens.com		00:00:00.278
ec76c15e.7bbf8	default	bianca.stadler@siemens.com		00:00:00.229
96181b27.097e78	default	bianca.stadler@siemens.com		00:00:00.109
3bead2ea.7590ae	current	stanislav.truntaev@siemens.com		00:00:00.099
5e24a148.d06b9	default	bianca.stadler@siemens.com		00:00:00.073

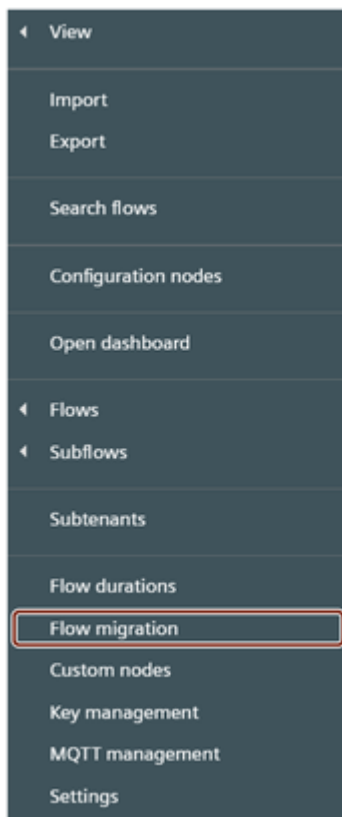
- ① Inactive node
- ② Active node



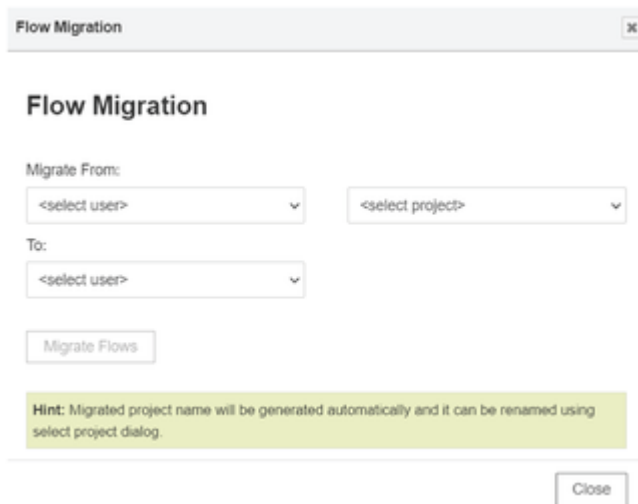
- If the triggered flow is deleted from Visual Flow Creator, then the node will be inactive.
- The flow duration will be the sum of the maximum compute hours of a triggered flow and the list will be descending based on the duration.

Flow migration

"Flow migration" allows you to migrate other user executed flows from the selected project in Visual Flow Creator. Click  to access the "Flow migration":



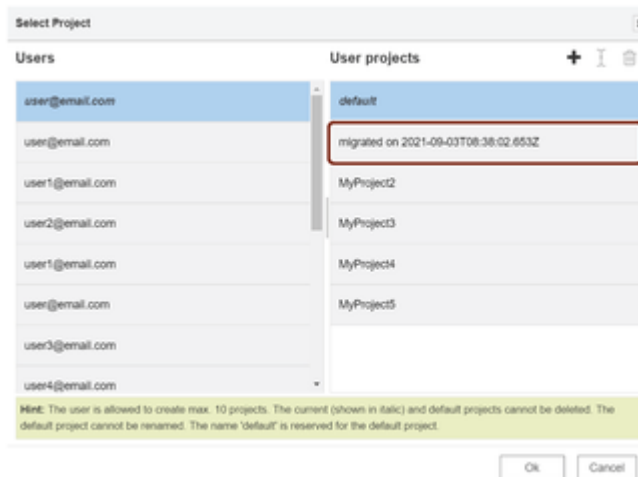
You can select the user and project to migrate the flows from one user to another user by clicking "Migrate Flows".



The "Flow Migration" dialog box contains the following elements:

- Flow Migration** (Title)
- Migrate From:** Two dropdown menus: "<select user>" and "<select project>".
- To:** One dropdown menu: "<select user>".
- Migrate Flows** (Button)
- Hint:** Migrated project name will be generated automatically and it can be renamed using select project dialog.
- Close** (Button)

You can access the migrated flows from "Projects".



The "Select Project" dialog box displays a list of users and user projects:

Users	User projects
user@email.com	default
user@email.com	migrated on 2021-09-03T08:38:02.653Z
user1@email.com	MyProject2
user2@email.com	MyProject3
user1@email.com	MyProject4
user@email.com	MyProject5
user3@email.com	
user4@email.com	


Hint: The user is allowed to create max. 10 projects. The current (shown in italic) and default projects cannot be deleted. The default project cannot be renamed. The name 'default' is reserved for the default project.

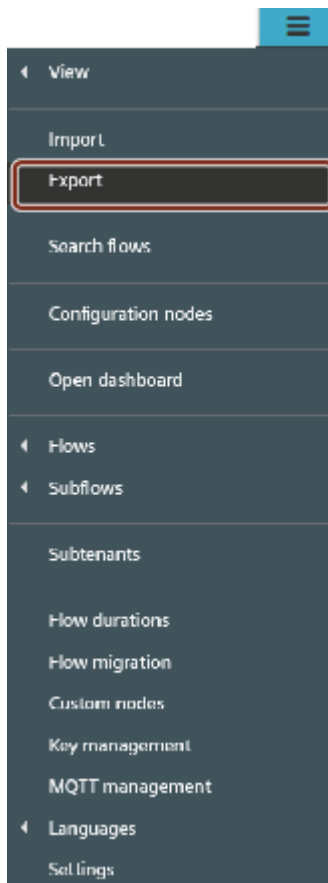
OK **Cancel** (Buttons)



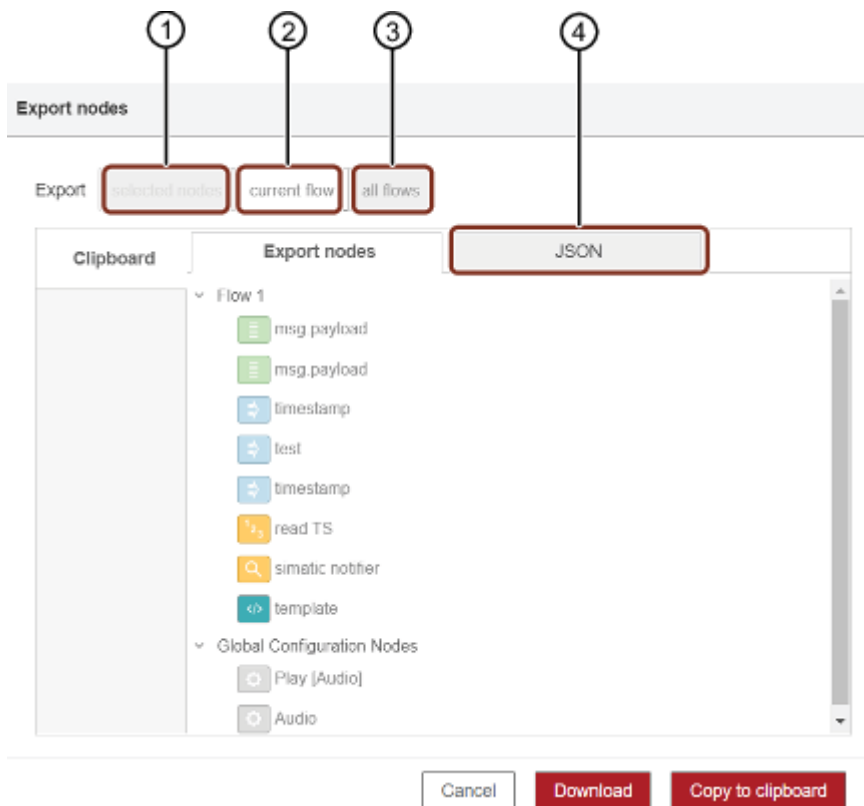
- Only Visual Flow Creator admin can use "Flow migration" page.
- After migration, project name will be replaced with the Visual Flow Creator generated project name.

Export

"Export" allows you to copy or download the other user or "Projects" flows in Visual Flow Creator. Nodes and flows can be copied in the JSON format by clicking "Copy to clipboard" or you can download them by clicking the "Download" button. Click  to access the "Flow migration":



You can export the single node, flows or all the flows of the current user or "Projects" in Visual Flow Creator.



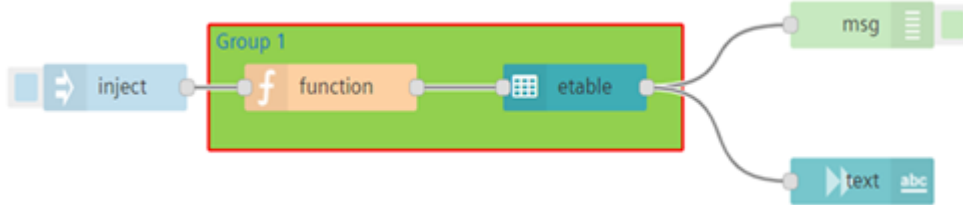
- ① Download or copy the selected node or flows
- ② Download or copy all flows the current tab
- ③ Download or copy the all nodes and flows


④ JSON preview tab

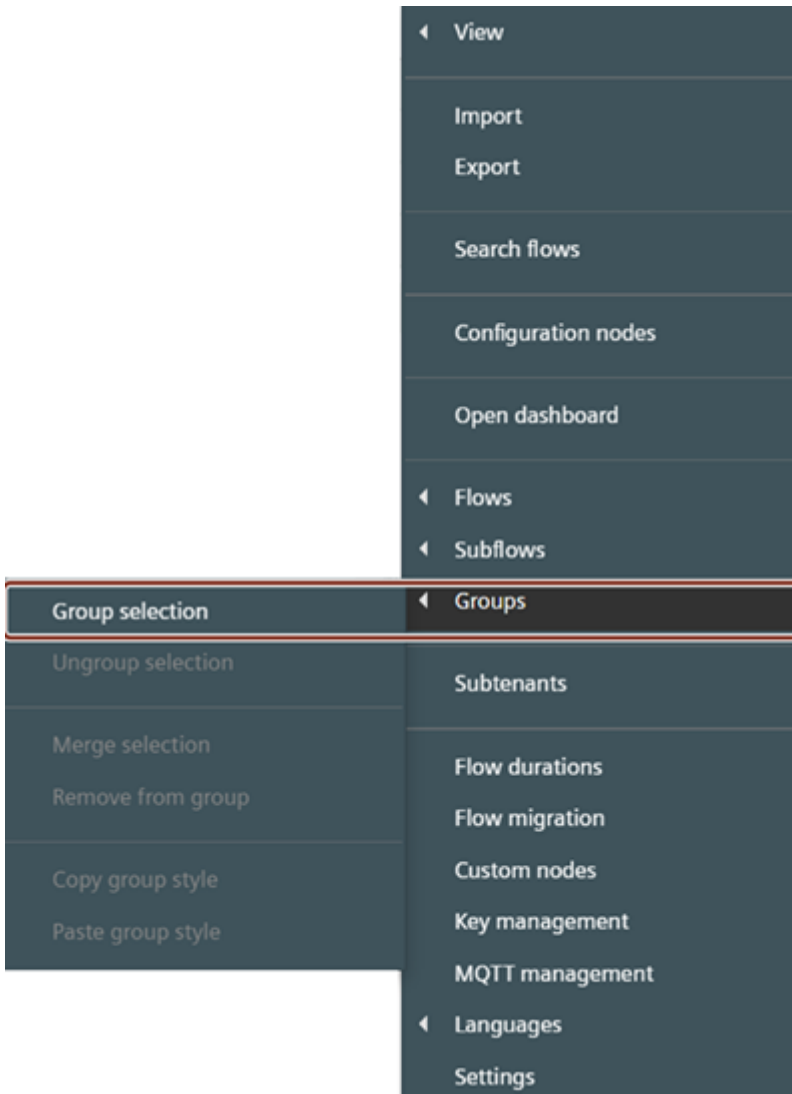
Groups

"Groups" allows you to combine the nodes into a group. It will create a boundary around the selected nodes and it can be moved, copied and perform other actions on the group.

You can create a flow and combine the nodes into a group as per the below example:



Click  to access "Groups":



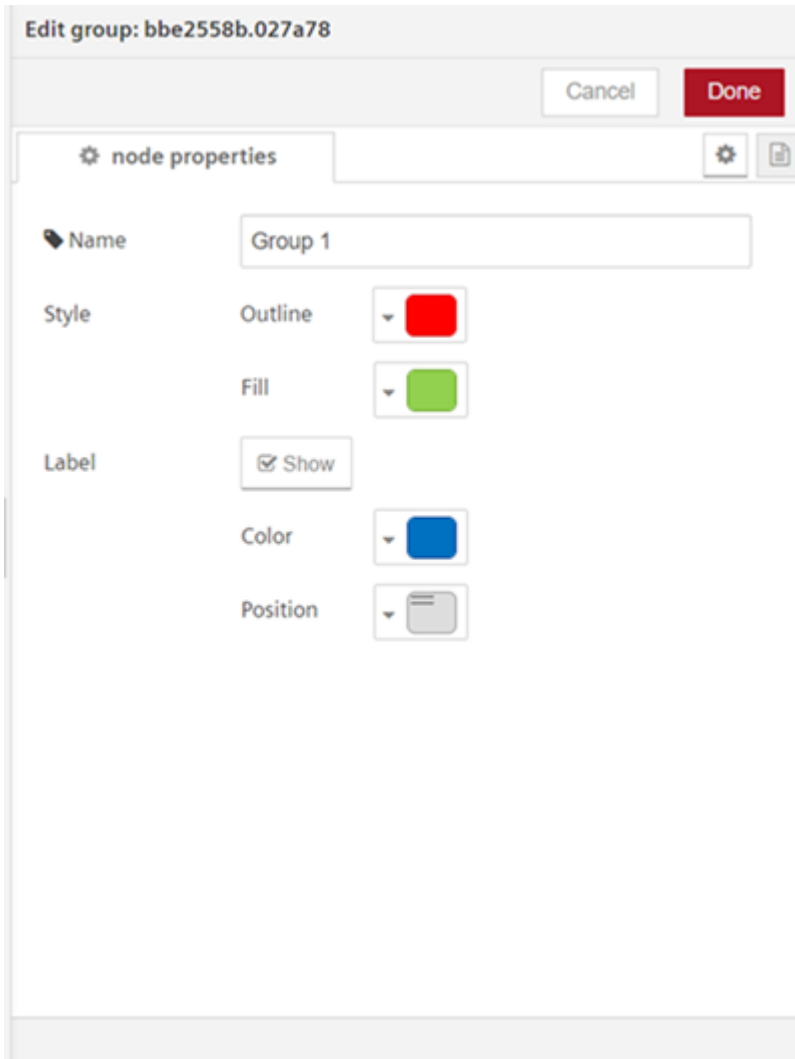
You can perform the following actions for the selected node:

- Group
- Ungroup
- Merge

- Remove
- Copy
- Paste

Edit Groups Properties

You can double click on the created "Groups" to change or update the properties of the Groups:



Properties	Description
Style	Allows to change or update the outline and fill colors of the groups.
Label	Allows to change or update the color and position of the label.

User rights in Visual Flow Creator

4

4.1 User rights

Visual Flow Creator adopts the user rights from [Settings](#). After activation of Visual Flow Creator, you receive all VFC related read and write permissions as a so-called environment administrator. As a environment administrator you use Settings to create additional users with modified permissions in Visual Flow Creator.

!!! Note - Permissions are defined as an act of authorizing or giving consent to carry out tasks. - User rights are access permissions that are predefined and given to certain user groups to perform tasks. - Roles are access permissions which are created and assigned to users.

The user rights depend on the following user roles:


- Administrator
- User
- Viewer

Please assign the admin role to just a few trusted employees.

The following table shows the permissions:

Permission	Role					
	Administrator Own flows	Administrator Other flows	User Own flows	User Other flows	Viewer Own flows	Viewer Other flows
View all environment's flows	✓	✓	✓	✓	✓	✓
Create nodes	✓	✓	✓			
Create tabs	✓		✓			
Delete tabs	✓	✓	✓			
Edit flows	✓	✓	✓			
Export nodes	✓	✓	✓	✓	✓	✓
Delete flows	✓	✓	✓			

Permission				Role		
Custom node	✓	✓				
MQTT	✓	✓				
Flow migration	✓	✓				

Browse the main menu  and you will find a section on subtenants in the drop down menu.

The subtenants have very limited access. The following rights have been granted to subtenants:

- They can view the assets which belong to them.
- They can view access and create their own flows.

Subtenants do not possess the rights to view the flows and assets of their peers.

Menu options are not available for subtenants.

Let us consider A is a main environment and B and C are subtenants of A. The following points can be concluded:

- A, B and C can view the main menu of "Visual Flow Creator".
- Tenant A has the rights to view and access all flows and assets.
- B and C can view their own assets and flows.
- B and C cannot view the assets and flows of each other.

Visual Flow Creator basics

5

5.1 Using nodes

Introduction to nodes

Definition

Nodes are configurable pieces of functionality that can process input data and produce output data. Examples range from data processing to complex functions and online services.

Categories

Visual Flow Creator palette contains set of node categories to create the flows in the working area.

It provides the following node categories:

- Basic nodes
 - Input
 - Output
- Industrial IoT nodes
- Function nodes
- Analytics nodes
- Storage nodes
- Array nodes
- Simple anomaly nodes
- IIoT Dashboard nodes
- Dashboard nodes
- Custom nodes

Node information

In the program user interface, you receive detailed information for each node in the info window. The info window displays the node type and the node ID for each node and other properties.

The info window provides information about the exact use of the node and shows the possible applications using examples.


To view the information for a node, click the node in the "Node palette" and open the "info" tab in the info window.

Edit nodes

You can edit or change the properties of the nodes as per the configuration to execute the flow in the Visual Flow Creator.

To use a node, you must move it from the node palette to the working area using drag-and-drop.

The following table shows the various node states:

Node states	Description
	<ul style="list-style-type: none"> - The blue circle identifies nodes that have not yet been saved. - The red triangle indicates missing inputs or errors in the node properties. - The text below the node shows status and error messages.

You can edit the node in the properties window, for example, to change the name of the node.

Each node has its specific properties window with different setting options. To open the properties window of a node, double-click a node placed on the working area.

The following graphic shows the properties window of the "inject" node:



Edit inject node

Delete Cancel Done

node properties

Name

Payload

Topic

Repeat

every

end in

endtime 12/13/2022, 16:28:50

Show next execution time

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Inject once at start?


Power Mode (Running time up to 120 sec.)

User-Defined Properties

+ Add New Property

Name	Value
No Properties defined yet	

> port labels

 **Power Mode**

In this mode, the flow runs up to 120 seconds for the heavy execution and the compute hours is calculated 2.5 times more than the normal execution. This mode is available for every node which is used to trigger and msg._powerMode.

Repeat	Description
None	In this mode, the node gets triggered only by clicking on its button. It will not be triggered automatically.
Interval	In this mode, the node can be configured to trigger and repeat until the specified end time. The node will be automatically terminated after reaching end time. The interval time units can be specified in seconds, minutes, hours or infinite. If required, choose "Start next execution time" parameter to display the next execution time for every execution. For example, the node can be configured to trigger at every 10 seconds with the specified end time to terminate the execution automatically.
Interval between times	In this mode, the node can be configured to trigger and repeat automatically in between the specified time on the specified days of the week. The interval time units can be specified in seconds, minutes or hours. For example, the node can be configured to trigger at every 1 minute in between the time 11am to 12pm of the weekday.
At a specific time	In this mode, the node can be configured to trigger and repeat automatically at a specified time on the specified days of the week. For example, the node can be configured to trigger once at 12pm of the weekday.

Processing input and output data

"input" and "output" nodes exchange data through messages. The messages themselves are JSON objects. They generally contain the "payload" parameter, which can be used for storing user data. A "msg object" can contain different properties of the "node properties". Different nodes may support different properties of the JSON object. For example, the "write timeseries" node uses the "Topic" property in order to retrieve the asset/ aspect/ variable-combination which it uses to access time series data.

The most often referred property is the "payload" property. Data can be of different types such as String, Array of structure mapping JavaScript objects and others.

For processing time series data, Visual Flow Creator uses the default format for timeseries. E.g.: { "_time": "2017-12-27T07:35:45", "pos_X": 112.2 }.

You can inject the input node at different intervals by using the "Repeat" element. The following options are available:

- interval: You can trigger to repeat every time as per the specified time interval.
- interval between time: You can trigger to repeat in between the specified time interval.
- at a specific time: You can trigger to repeat at a specific time interval.



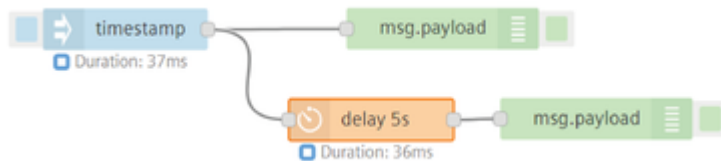
- The maximum interval that can be specified is about 596 hours or 24 days.
- More than 10 seconds time interval triggers can lead to high consumption of available compute hours.

Dynamic message properties

In Visual Flow Creator function nodes, the user has full control over the message including its message properties like "Topic" etc.

Execution duration of the flow

With this information, you get the duration of the complete execution time of the flow displayed under every node which starts or triggers the flow. Maximum execution time limit for a flow is 30 seconds, except the flow with power mode execution time limit is up to 120 seconds. In the below example, the execution time for the first flow output will be generated and thereafter the second flow output will be generated as per the specified delayed time in the delay node.




Usage of basic nodes

The Visual Flow Creator component has the following basic nodes:

- Input nodes
- Output nodes


Input nodes

Input nodes trigger data to the flow. Each input node has one output. Use input nodes to supply data from other services. They generate messages for downstream nodes.

Input nodes	Description
	- Shows an input node. - The blue button on the left triggers the input node.

Output nodes

Output nodes send the data from the flow to other services or to the debug tab. Output nodes have one input. They consume the messages and trigger an action (for example sending an http response).

Output Nodes	Description
	- Shows an output node. - The green button on the right deactivates or activates the output node.

Usage of flow switch node

The Flow Switch node allows you to activate or deactivate a Visual Flow Creator flow tab.

Example

To activate or deactivate a Visual Flow Creator flow tab, follow these steps:

1. Create the flow as shown below:



2. Edit http request node properties:

- Method: GET
- Use Industrial IoT service: Check the option
- Industrial IoT path: `/api/vfc/v3/flows/projects/0/tabs/8888a447.745cc8?user=steve@email.com`



"8888a447.745cc8" is a flow tab unique ID for the selected tab. This unique ID of the selected tab is available in Visual Flow Creator URL.

3. Edit function node properties:

- Code: `msg.payload = !msg.payload.disabled; return msg;`

4. Edit switch node properties:



"8888a447.745cc8" is a flow tab unique ID for the selected tab. This unique ID of the selected tab is available in Visual Flow Creator URL.

3. Edit function node properties:

- Code: `msg.payload = !msg.payload.disabled; return msg;`

4. Edit switch node properties:

5.Edit flow switch node properties:

6.Save and execute the flow.

Result

To view the output, activate and deactivate Visual Flow Creator flow tab by using toggle button in the dashboard. After activating or deactivating the toggle button in the dashboard, refresh Visual Flow Creator editor page to view the output.

Activated "Pump" tab




Deactivated "Pump" tab



After "Pump" flow tab is deactivated, the flows are disabled to execute under this tab.

Usage of image output node

The simple image output node allows you to preview the results of face detection, object recognition and so on.

Click  button of the image output node can be used to enable or disable the image preview. It is recommended (for performance) to disable or remove any Image-Output nodes that are not being used.

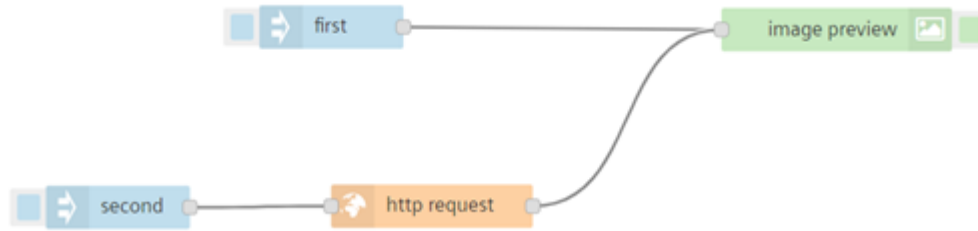
You can edit the Image output node properties with the following parameters:

Parameters	Description
Property	This node expects the selected <code>msg . property</code> (default <code>msg . payload</code>) to contain one of the following formats: <ul style="list-style-type: none"> - A base64 encoded string - A buffer - A URL to an image Sending a blank or null payload will remove/hide the preview panel.
Width (1px - 800px)	The width (in pixels) of the image needs to be specified to display in the flow.
Resize images on server side	When too much data is sent to the browser, the flow editor can become unresponsive. <ul style="list-style-type: none"> - When it is activated, the images will be resized (to the specified width) on the server side and then those small thumbnail images will be send to the browser, to reduce the bandwidth. - When it is de-activated, the (original) large images will be send to the browser. Once the image is arrived to the browser, it will resize the image to the specified width.

Example

To preview the image output using image output node, follow these steps:

1. Create the flow as shown below:



2. Edit the first inject node properties to enter the URL as a string value:

- Payload(string) example URL:

`https://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Siemens-logo.svg/250px-Siemens-logo.svg.png`

3. Edit the second inject node properties and set the payload value as timestamp

4. Edit http request node properties:

- Enter the example URL:

`https://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Siemens-logo.svg/250px-Siemens-logo.svg.png`

- Return: Select "a binary buffer" from the drop-down

5. Edit image output node properties:

Dialog box titled "Edit image-output node" with buttons: Delete, Cancel, Done.

node properties

- Name: Name
- Property: msg.payload
- Width (1px - 800px): 250
- Resize images on server side

port labels

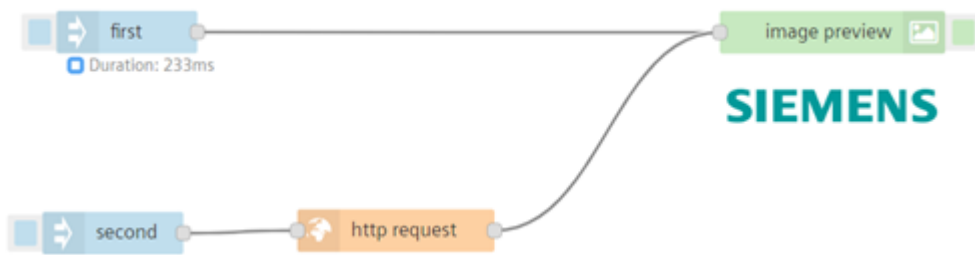
- Property: msg.payload
- Width (1px - 800px): 250
- Resize images on server side - Uncheck for "first" flow and check for "second" flow to reduce the bandwidth

6. Save and execute the flow.

Result

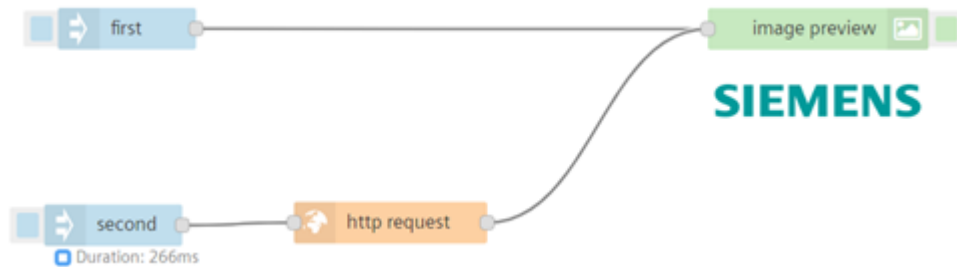
"first" flow

When you execute the "first" flow for multiple times, it will preview the same image as the result if the URL consists of multiple images.



"second" flow

When you execute the "second" flow for multiple times, it will preview the different image as the result if the URL consists of multiple images.



For both the flows "first" and "second", when you click on the image preview then it will be disappeared.

Usage of junction-nodes

Junction node allows you to combine the multiple wires which are connected to multiple nodes. To insert the junction, "Right-click" on the work area and click "Insert junction".

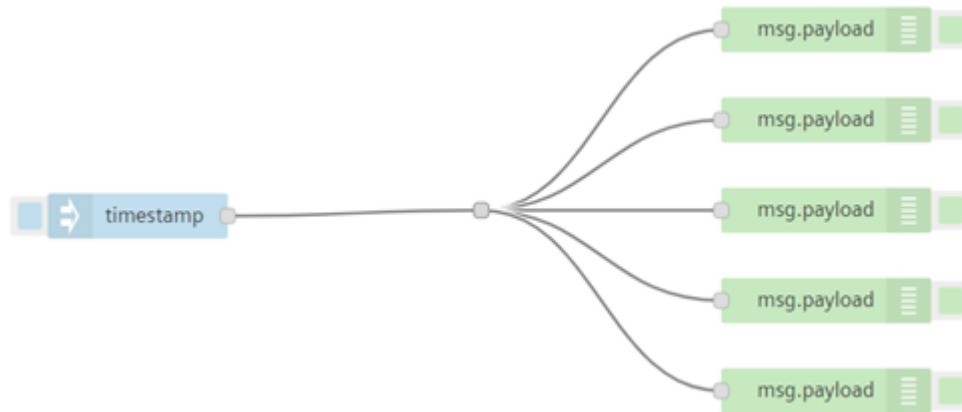
Insert node	
Insert junction	
Undo	ctrl-z
Redo	ctrl-ûz
Cut selected nodes	ctrl-x
Copy selected nodes	ctrl-c
Paste nodes	ctrl-v
Delete Selection	delete
Select All Nodes	ctrl-a
Export	ctrl-e
Group Selection	ctrl-ûg

Junction node is displayed as shown below:



Example

The following example displays the usage of the junction node:



5.2 Create flows

Introduction to flows

A flow consists of multiple interconnected nodes. The links join the nodes on the working area to form a flow chart. The data enters the flow via input nodes and is processed by other nodes. Each user can create up to ten tabs in the working area of Visual Flow Creator.

Create a basic flow

Example scenario

The smart city has alternative wind and solar power generation. The dependency on weather sometimes leads to severe constraints.

Objective

The basic flow is to convert data of a weather web service from a JSON format to a JavaScript object structure for further processing.

Requirement

A weather forecast web service supplies information on wind speed, amount of sunshine and temperature.

Procedure

To create a basic flow, follow these steps:

1. Move an "input" node to the working area using drag-and-drop.

Repeat step 1 with "function node" "http request", "function node" "json" and "output node"

2. "debug".

3. Interconnect the nodes. The following graphic shows the connected nodes:



4. Double-click the "http request" node.

5. Select the "GET" method in the drop-down menu.

Enter the API Call URL with the corresponding data of a weather web service. The graphic

6. below shows the parameters for the "http request" node:

Dialog box titled "Edit http request node" with buttons for "Delete", "Cancel", and "Done".

node properties

- Name:
- Method:
- Use MindSphere service
- URL:
- Use basic authentication
- User secrets: **+ Add New Secret**

Name	Value
No secrets defined yet	

- Return:
- Timeout:

> port labels

7. To start the flow, click the blue button to the left of the "timestamp" node.

Result

You have created a flow in which weather data from a web service can be retrieved via an "http request" node.

The data is supplied in JSON format. In order to view the data in a "debug" node and evaluate it with JavaScript, a conversion to an object structure takes place using a "json" node.

The graphic below shows the output of the flow in the log area:


```
1/9/2018, 2:35:06 PM node: 4aeb8395.3bdffc
msg.payload : Object
  ▼ object
    ▶ coord: object
    ▼ weather: array[1]
      ▼ 0: object
        id: 800
        main: "Clear"
        description: "clear sky"
        icon: "01d"
        base: "stations"
    ▶ main: object
      visibility: 10000
    ▼ wind: object
      speed: 4.6
      deg: 120
    ▼ clouds: object
      all: 0
      dt: 1515504000
    ▶ sys: object
      id: 2929567
      name: "Erlangen"
      cod: 200
```

Debug flows

Using a debug node

To ensure correct execution and data processing of a flow, you can check the flow with a debug node. You can deactivate the debug node like an output node using the green button on the right. Deactivation of a debug node helps you to reduce the number of debug messages from multiple debug nodes.

To fully display the properties and contents of messages in the debug window, you must set the "Output" to "complete msg object" in the properties of a debug node. The debug window then shows the complete "msg-object" content, for example, can include the following fields:

- **topic**
- **payload**
- **_msgid**

You also have the option of displaying any other property (msg.payload).

Debug window

The node ID identifies the node in the debug window. A click on the node ID highlights the corresponding node in the working area. The selection helps you associate the corresponding

node with the message in the debug window. The debug window displays important information regarding the data processing and a preview of the message or array output.

The debug window can format arrays and objects in order to improve readability. For timeseries, an additional chart gets added which visualizes the time series data.

Copy flows

You have the option of copying nodes between multiple flows. For this purpose, you can select nodes in a flow and copy them to the clipboard. You can paste the nodes from the clipboard in a new flow.

Procedure

To copy a flow, follow these steps:

1. Select the flow you want to copy.
2. Open the flyout menu in the main navigation.
3. Open the "Export nodes to clipboard" window under "Export > Clipboard".
4. Select the export parameters, for example "selected nodes"
5. To complete the export, click "Export to clipboard".
6. Open the flow tab where the flow is to be copied.
7. To paste the flow, open the flyout menu in the main navigation.
8. Click "Import > Clipboard".
9. Click inside the "Paste nodes here" window in the "Import nodes" window.
10. To paste the nodes, press the key combination CTRL+V.
11. To import the flow, click the "Import" button.
12. Place the flow in the working area.



Inside an instance from VFC, nodes can simply be copied by using CTRL+C and CTRL+V

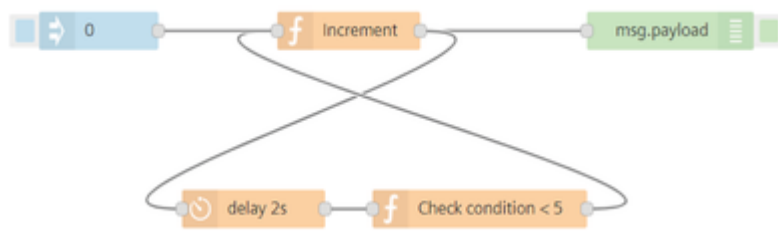
Allow cyclic flows

This feature can execute the cyclic flows for each tab individually in Visual Flow Creator. The "Allow cyclic flows" can be enabled or disabled by using the toggle button at the edit flow tab properties.

Example

The below given is an example to execute the cyclic flows using delay node, follow these steps:

1. Design the flow as shown below:



2. Edit inject node properties:

- number: 0

3. Edit function node properties:

- Name: Increment
- Language: javascript
- Code: msg.payload++; return msg;

4. Edit function node properties:

- Name: Check condition < 5
- Language: javascript
- Code:

```
if (msg.payload < 5) {
  return msg;
}
```

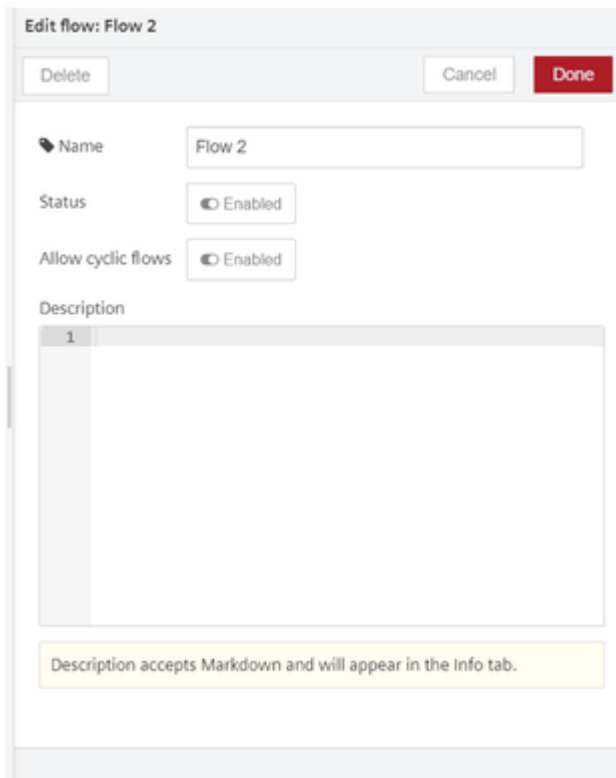
5. Edit delay node properties:

- Action: Delay each message
- Fixed delay
- For: 2 seconds



If you proceed to save and deploy the flow, a notification will be displayed with the message "Deploy failed: cycles are detected. This can lead to high compute hours consumption!"

6. Edit flow tab properties:

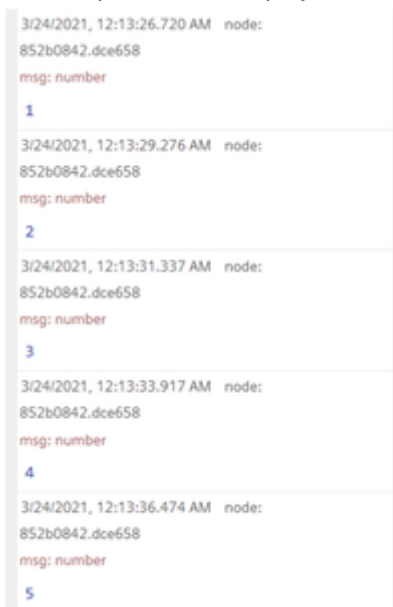


- Allow cyclic flows: Enabled

7. Save and execute the flow.

Result

The output will be displayed in the debug tab.



Nodes in Industrial IoT

6.1 Nodes in Industrial IoT

Introduction to Industrial IoT nodes

Industrial IoT nodes are specific nodes for processing data. The Industrial IoT nodes are only available in the Visual Flow Creator component.

Visual Flow Creator groups the Industrial IoT nodes into the following categories:

- **Industrial IoT nodes** There are two types of Industrial IoT nodes. You can use the Industrial IoT in nodes to read data, such as time series, to a flow. Industrial IoT out nodes are used to write the data using services.
- **Function nodes** These are programmable and user defined nodes which can be customized to perform different functions.
- **Analytics nodes** The analytics nodes prepare the time series data read out from Industrial IoT with analytics methods.
- **Storage nodes** The storage nodes provide functionalities to access MongoDB or Postgres databases.
- **Array nodes** These nodes represent function methods like filter, map, find and reduce for arrays.
- **Simple anomaly nodes** These nodes to detect anomalies in a data set.
- **Integrated Data Lake nodes** These nodes allows you to store structured and unstructured data or objects in its native format as needed.
- **Semantic Data Interconnect (SDI) nodes** These nodes provides a simple way to prepare data for establishing semantic correlations and data query processing.
- **Signal Validation nodes** These nodes validates sensor time series data of an entity.
- **Pump-Simulation node** These nodes allows to generate the time series data of a pump.
- **IIoT Dashboard nodes** The dashboard allow you to build, create and interchange dashboard nodes and default dashboard nodes in one single dashboard.
- **Insights Hub Business Intelligence nodes** These nodes allows to create custom visualizations of your data.

- **Simatic Notifier node** These nodes allow to integrate with Simatic Notifier application.
- **Predictive Learning (PRL) nodes** These nodes combine analytics, statistics and machine learning algorithms to provide unmatched insight into trends in your data.
- **Dashboard nodes** These nodes provide a set of nodes for reading, visualizing and analyzing data on a live dashboard.
- **Custom nodes** The custom nodes help to design and deploy your own customized nodes if any of the required nodes are not available in Visual Flow Creator.



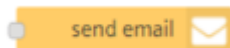
- Rules are configured by a specific user and thereafter, these rules are executed in a system context for the entire environment.
- With this, the user access to specific assets is managed by the rule is restricted at the later point of time. The rule will be executed irrespective of access restrictions of the user. This is an intended system behavior.

6.2 Industrial IoT node library

Usages of Industrial IoT nodes

The following listing shows the specific "Industrial IoT" nodes and their respective functions:

Send email



The "send email" node sends an email to one or more recipients.

You use this node to:

- Notify the user about an important event.
- Attach the files stored in an asset.



It supports only PDF, ZIP, CSV and JSON file types.

Requirements:

- The email address must be valid.

Example

To send an email from VFC, follow these steps:

1. Create the flow as shown below:



2. Edit read file node properties:

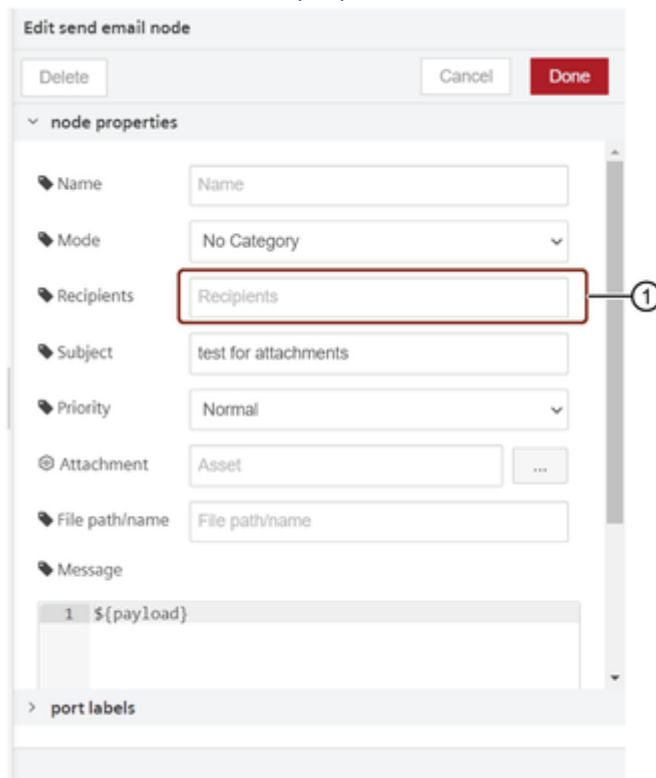
- Asset: Select an Asset file.
- File path: Enter the file path.

3. Edit function node properties:

- Code:


```
msg.attachments = {  
  content: msg.payload, filename: 'example.zip'  
};  
return msg;
```

4. Edit send email node properties:



① Enter the email address

5. Save and execute the flow.

 **Content of "send mail" node**

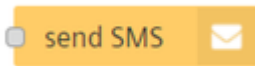
Users of Visual Flow Creator are responsible for the content of all emails that are sent using the "send mail" node.

Output

The email is successfully received with the attachment.



Send SMS



The "send SMS" node sends a message to receive the notification directly on the mobile phone.

Example

To send a message to your mobile phone from VFC, follow these steps:

1. Create the flow as shown below:



2. Edit inject node properties:

- Payload (string): Dear Customer, Welcome to VFC.

3. Edit send SMS node properties:

Edit send SMS node

Delete Cancel Done

node properties

Name Name

Recipients +498888888888

Message

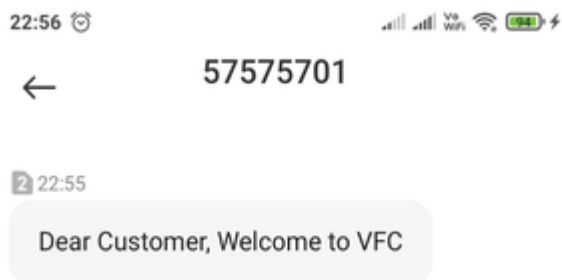
1 `${payload}`

port labels

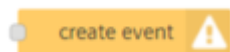
4. Save and execute the flow.

Output

The message is successfully received on your mobile phone.



Create event node



The "create event" node writes events for a specific asset to Industrial IoT.

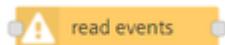
You use this node to:

- Document events such as errors, status changes in Industrial IoT.
- Make unusual events visible in other applications, for example, Insights Hub Monitor.

Requirements:

- An asset must already exist in Industrial IoT.

Read events node



The "read events" node reads event data from Industrial IoT.

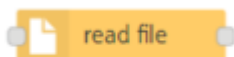
You use this node to:

- Evaluate existing events for an asset.
- Enable statistical analyses.
- Check status of events.
- In the interval mode, the dates can be passed as an ISO string or as a date object formats.

Requirements:

- An asset must already exist in Industrial IoT.

Read file node

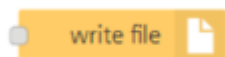


The "read file" node reads files from Industrial IoT and outputs the file contents.

You use this node to:

- Send the file contents as a message (only for non-binary files)

Write file node



The "write file" node writes a file and links it to a certain asset.

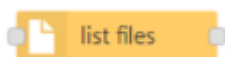
You use this node to:

- Write the received message as file contents to a file.
- Define the mime type of the file using "Predefined type" element.
- Overwrite to replace the existing file using "Overwrite" element.

Requirements:

- An asset must already exist in Industrial IoT.

List file node



The "list files" node reads all file names for one asset as an array.


You use this node to:

- List all files of an asset

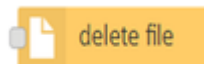
Requirements:

- An asset must already exist in Industrial IoT.

The following table shows "list files" node configuration properties:

Properties	Description
Asset	Select the asset to list the files.
Limit	Set the limit for the files of the list. Note: The maximum number of items being returned (default is 200)
Offset	Enter the offset number to skip the specified number of the files from the list.
Filter	Apply the filter to the files from the list. Click  to add the filter. Files can be filtered by name, path and updated field. The updated field will be converted into an ISO date string . For more information on filtering IoT file services, see IoT File Service .

Delete file node



The "delete file" node deletes the file of an asset from Industrial IoT.

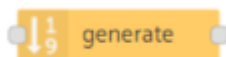
You use this node to:

- Delete the file of an asset.

Requirements:

- The file of an asset must already exist in Industrial IoT.

Generate node

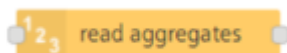


The "generate" node generates basic time series data and combines them. The "generate" node supports the creation of complex signals.

You use this node to:

- Simulate cyclical time series data for test purposes.
- Set signal type, amplitude, offset, etc. for the signal
- Use signal type "walk" to generate a continuous signal path

Read aggregates



The "read aggregates" node reads aggregates by using the aggregate or time series API from Industrial IoT.

You use this node to:

- Read aggregated interval data for specific interval.

- Create statistical values
- Allows applications to retrieve smaller data sets that cover a longer period of time with much better performance than processing all of the raw time series data.

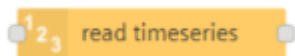
Requirements:

- Asset, aspect, variable must exist in Industrial IoT.
- Entity Id, the property set name, a time range, and a requested aggregation interval of the application.

Configurable properties	Description
API	In this property, you can configure using the aggregate API or the timeseries API.
Interval Unit	In this property, you can configure the interval unit using the possible values like minute, hour, day, week and month. (applicable for aggregate API only)
Interval Value	In this property, the value depends on the selected interval unit. (applicable for aggregate API only)
Count	In this property, you can specify the count of the result aggregates. (applicable for aggregate API only)
Mode	In this property, 2 modes are available: <ul style="list-style-type: none"> • Interval: In this mode, you need to set the interval with "From" and "To" timestamps and timezone. The dates can be passed as an ISO string or as a date object formats. • Period: In this mode, you need to set the period from the past to now.

For more information on API specifications, refer to [IoT Times Series Aggregates Service API](#).

Read timeseries node



The "read timeseries" node reads time series data from Industrial IoT.

You use this node to:

- Further process or convert data using a function or analytic node.
- Specifically evaluate or validate the time series data.
- Process multiple time series data together (using a "combine" node).
- Select different types of variables from different aspects.
- Read values of aspects defined for assets.

Mode	Description
------	-------------

Mode	Description
Interval	In this mode, you need to set the interval with "From" and "To" timestamps and timezone. The dates can be passed as an ISO string or as a date object formats.
Period	In this mode, you need to set the period from the past to now.
Last Value	In this mode, the last timeseries value will be read. Note: An empty value will be read, if the last timeseries entry doesn't contain required variable the empty array will be returned as output.
From last execution	In this property, the value depends on the selected interval unit. (applicable for aggregate API only)

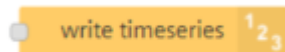
Requirements:

- Asset, aspect and a variable must exist in Industrial IoT.



Maximum count of timeseries data points is 2000.

Write timeseries node



The "write timeseries" node writes time series data to Industrial IoT.

You use this node to:

- Write unprocessed data to Industrial IoT.
- Update the existing time series data by activating "Use merging" option. For example, you can update the aspect value without changing the variables.

Requirements:

- Asset, aspect and a variable must already exist in Industrial IoT.

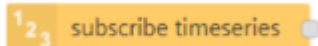


Update of time series data on new write operation

The "write timeseries" node overwrites all data and values of an existing timestamp. With this write operation, all variables of an asset or aspect for the respective timestamp are updated and overwritten. The "write timeseries" node overwrites existing data of variables that are not taken into consideration with zero.

The write operations can take some time. The written time series data cannot be read immediately after they are written - especially the generation of aggregations can take longer time. The written data could appear delayed (up to 15 minutes).

Subscribe timeseries node



The "subscribe timeseries" node subscribes time series data from Industrial IoT.

You use this node to:

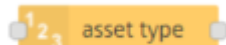
- Get notifications of new incoming time series data in order to perform tasks.
- Get updates on the new timeseries data which is posted in IOT.

Requirements:

- Asset, aspect and a variable must exist in Industrial IoT.

Once the subscription is created, it can take up to 15 minutes for the service to start working.

Asset type



The "asset type" node makes it possible to read data from multiple assets which belong to the same asset type.

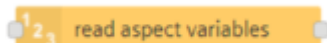
You use this node to:

- Get multiple messages with topic properties defining assets which belong to a particular asset type. The topic property messages can be directly forwarded on the input pin of read timeseries node.

Requirements:

- Asset type, asset, aspect and a variable must exist in Industrial IoT.

Read aspect static vars

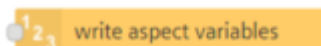


The "read aspect static vars" node reads the static variables from the aspects.

Requirements:

- Asset, aspect and a variable must already exist in Industrial IoT.

Write aspect static vars

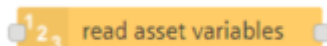


The "write aspect static vars" node updates the static variables in the aspects.

Requirements:

- Asset and aspect must already exist in Industrial IoT.

Read asset static vars

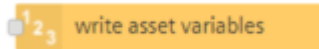


The "read asset static vars" node reads the static variables from the assets.

Requirements:

- Asset, aspect and a variable must already exist in Industrial IoT.

Write asset static vars

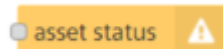


The "write asset static vars" node updates the static variables in the assets.

Requirements:

- Asset and aspect must already exist in Industrial IoT.

Asset status

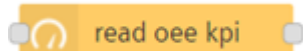


The "asset health" node updates the status of the asset health. The asset health status can be viewed in the info tab of Insights Hub Monitor.

Requirements:

- Aspect in the asset must already exist in Industrial IoT.

Read oee

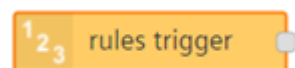


The "read oee" node reads OEE KPIs.

Requirements:

- Aspect in the asset must already exist in Industrial IoT.

Rules trigger



The "rules trigger" node gets triggered by the Insights hub rules engine.

Using Industrial IoT in nodes

Let's try to use described nodes in some simple scenario:

Example scenario

A new logistics center is being set up in the smart city. The power consumption (Kilowatt hour) of the cranes is acquired in Industrial IoT. The cranes are not operated continuously. The smart city has its own power generation through wind turbines, photovoltaics and heat pumps.

Objective

The owner of the logistics center would like to calculate the energy consumed by the crane operation in order to better estimate its cost accounting of energy consumption. The owner

would like to know whether a change to another energy provider is worthwhile. The following detailed data regarding the energy consumption of the cranes is needed for this:

- Magnitude of the share of the base load.
- Magnitude of the peak loads.

The owner would like to use the crane data for a comparison of different energy producers.

Requirement

- The crane is connected to Industrial IoT and collects the energy data.
- The energy production of the different energy suppliers of the smart city is acquired in Industrial IoT.

Procedure

To use a "Industrial IoT in" node, follow these steps:

To create a "timestamp" node on the working area, move an "input node" to the working

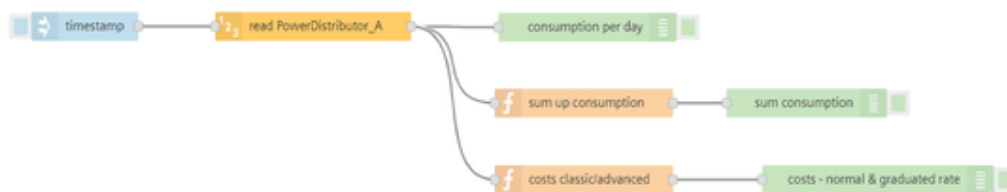
1. area using drag-and-drop.

2. Move the read timeseries to the working area using drag-and-drop.

3. Repeat step 1 with "function node" and "output node" "debug".

- In this example, the function nodes contain JavaScript code for calculation of the energy consumption and the costs.

4. Interconnect the nodes.



5. Double-click the "read timeseries" node.

6. To enter the asset data, click on next to the topic.

Select the asset, aspect and variable that are to be read, for example crane, energy data, consumption (KWh).

8. Enter the time range using the drop-down menus of "Mode" or "Period".

9. To start the flow, click the blue button to the left of the "timestamp" node.

Result

The flow reads the energy data of the cranes. The nodes can be flexibly modified and used with data of another energy supplier. Continuous checking of the energy costs is ensured. The data can be compared with other energy suppliers.

Using Industrial IoT out nodes

To use described nodes in an simple scenario:

Example scenario

The smart city has its own energy generation through wind energy. In the event of a storm, certain measures must be taken, for example implementation of a plan for maintaining the energy supply through other sources or temporary reduction of energy demand.

Objective

The smart city would like to send an email to its executives in the event of a storm warning.

Requirement

- A weather forecast is required for next 2 days.
- An automatable check of weather data, for example through an API call of a weather service.
- The data must include the wind speed.
- You have created the basic flow for a weather web service. For more information about the basic flow, refer to [Create a basic flow](#).

Procedure

To send an email, follow these steps:

1. Move the "Industrial IoT out" node "send email" to the working area using drag-and-drop.
2. Connect the "function" node to the "send email" node.



3. Double-click the "send email" node.
4. Enter the e-mail address of the recipient.
5. To activate the flow, click the blue button to the left of the "'timestamp'." node.

Result

The information supplied by the weather web service is converted to an object structure in the "json" node in order for the information to be processed in the "function" node. The "function" node uses a script to check for a storm warning in the list of weather data. The "send email" node sends an email to the executives in the event of a storm warning.

Example using Industrial IoT nodes

Example Scenario

The maintenance team of a turbine company needs to regulate the pressure and temperature of the existing assets in the organization.

Objective

The maintenance team needs to read all data from the existing assets so as to know if the readings of all the assets are in control and operating properly.

Let us try to display the data of the assets from the given asset list - one at a time.

Requirements

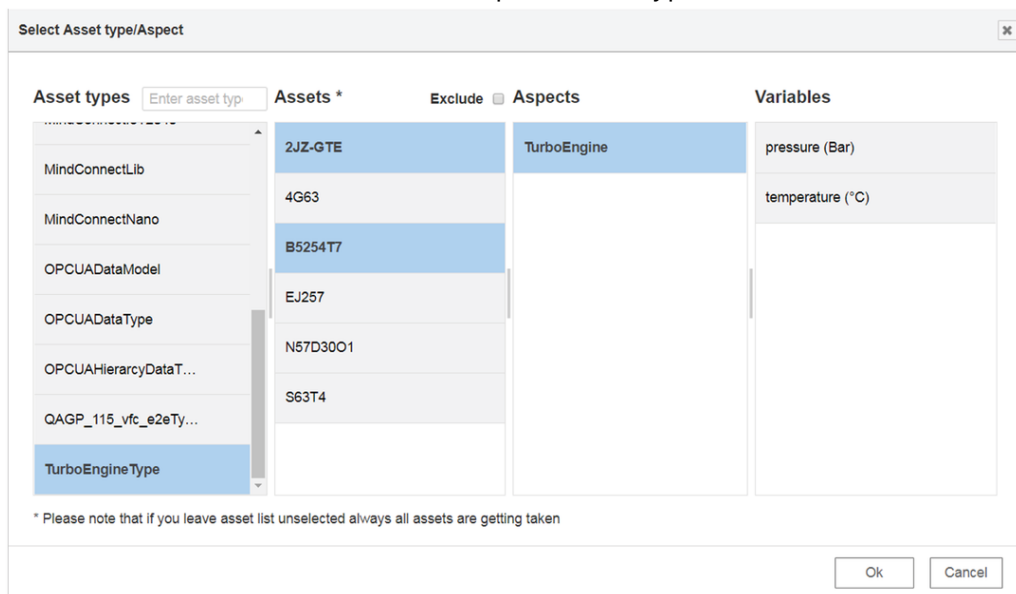
- A timestamp input
- Asset type node
- Read timeseries node
- Message payload

Procedure

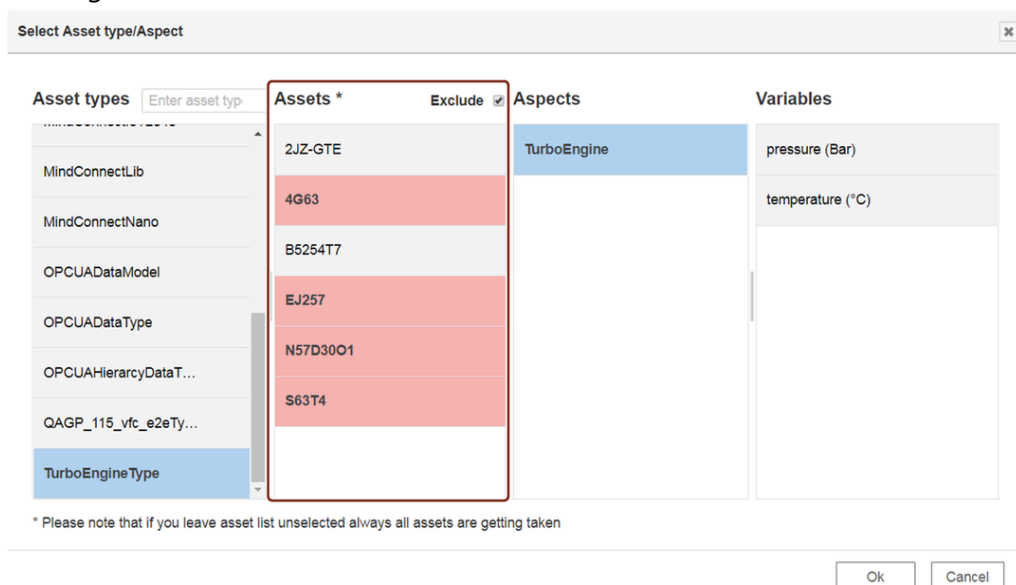
1. Import the "asset type" node and configure the topic properties as show below:

Double click the "asset type" node. The editor for "asset type" node pops up. Browse

- and select the topic for an asset.
- Define the selections for the variables, aspects, asset type of an asset.



You can also deselect assets from the "asset type" node pop-up by using the "Exclude" feature. Click the "Exclude" checkbox and select the assets for which data readings are not to be included.



2. Design and connect the nodes as shown below:



3. Inject the timestamp.

Result

You can view the results in the message payload.

In this example the "read timeseries" node reads the data from the selected asset and displays the data in the message payload.

The left screenshot shows a message with a red line graph. The right screenshot shows a message with a JSON array containing 60 objects, each with 'temperature', 'pressure', and '_time' fields, and a red line graph at the bottom.

Example using asset health node

Example

The maintenance team of a turbine company needs to update the health status of the existing assets in the organization. The asset health status can be viewed in Insights Hub Monitor.

Objective

The maintenance team needs to check the health status of an asset and update it using asset health node.

Procedure

To update the asset health status from the previous health status to current health status, follow these steps:



1. Design the flow as shown below:

Edit asset status node

Delete Cancel Done

node properties

Name


Asset DemoPump (15d9abfa663c41019f67a6ea6f3f) ...

Health status Information

Unconfirmed health value (must be acknowledged by user)

port labels

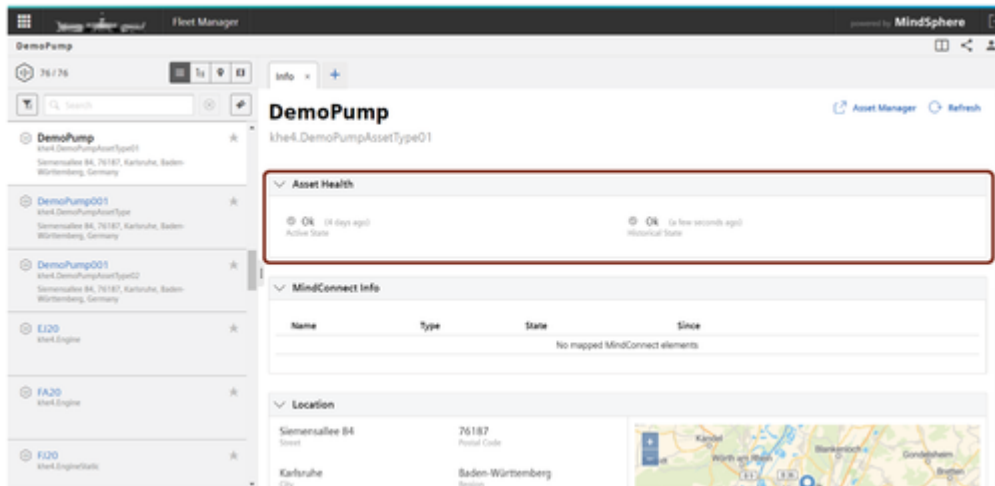
2. Edit the asset health node properties:

- Click  to select an asset.
- Health status: Change from "Information" to "Ok".

3. Save and execute the flow.

Result

To view the result, open "Info" tab in Insights Hub Monitor. The asset health status is successfully updated from "Information" to "Ok".

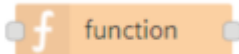


If the asset health status is unconfirmed, it should be acknowledge by the user.

6.3 Usages of function nodes

The following shows the specific "function" nodes that are available in Insights Hub with their respective functions. These are programmable and user defined nodes which can be customized. For more information about other function nodes, refer to [Node-RED documentation](#).

Function



The "function" node is used to run JavaScript code against the message object. You can use this node to:

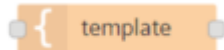
- Design your JavaScript code here which will run against the messages passed. This in turn will return zero or more messages to downstream nodes for processing.
- Design functions in the node properties and save them in the library. You can reuse the functions whenever required.

You can write your Javascript code in the editor.



1. The global context has name "glob".
2. setTimeout, setInterval are not available for use.

Template

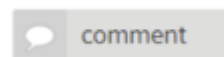


The "template" node sets a property based on the user defined template. By default, the template uses the "Mustache" format. The "Mustache" format can be switched off if required. You can use this node to:

- Design the template and set its property.

Mustache can be used for HTML, config files and source codes. It works by expanding tags in a template using values provided in an object.

Comment



The "comment" node allows you to add comments in the flows that you have created. The editor will accept information in the MarkDown syntax. You can use this node to:

- Add formatted text to the created flows that will be shown in the info tab.
- Sticky note is used to select the type of font as given below list:
 - Do not show: It will not show the font on the node in the working area.



- Big font: It will show the font on the node with bigger in size in the working area.



Small font: It will show the font on the node with smaller in size in the working

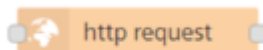
- area.





Sticky note messages will not allow MarkDown format.

http Request



The "http request" node sends HTTP requests and returns the response. You can use this node to:

- Access the data from external services.

- To use Insights Hub services, activate "Use Insights Hub service" option and configure the requested service path in the Insights Hub path field.
- If "Extended scopes" option is selected then, the request will be processed with a token that has more scopes for a normal user. For example, creating or deleting assets. Admin can create or modify the scopes for the normal user.

Scopes	Description
atm.apt.d	Permission allows user to delete aspect types.
atm.apt.w	Permission allows user to create or update aspect types.
atm.apt.r	Permission allows user to read aspect types.
atm.fa.w	Permission allows user to assign files to asset types.
atm.fa.d	Permission allows user to delete file assignments.
atm.r	Permission allows user to read asset types.
atm.w	Permission allows user to create or update asset types.
atm.d	Permission allows user to delete asset types.
as.ad.u	Permission allows user to use Anomaly Detection API - without batch endpoints.
as.adb.u	Permission allows user to use Anomaly Detection Batch API.
as.kc.u	Permission allows user to use KPI Calculation API.
as.sa.fft	Permission allows user to use Spectrum Analysis API.
as.tp.u	Permission allows user to use Trend Prediction API.
asm.c	Permission allows user to create assets.
asm.d	Permission allows user to delete assets.
asm.m	Permission allows user to move assets.
asm.r	Permission allows user to read assets.
asm.u	Permission allows user to update assets.
asm.fr	Permission allows user to read files.
asm.fw	Permission allows user to create or update files.
asm.fd	Permission allows user to delete files.
asm.fa.w	Permission allows user to assign files to assets.

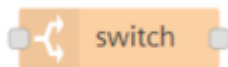
Scopes	Description
asm.fa.d	Permission allows user to delete files assignments.
asm.h.d	Permission allows user to delete hierarchy type assets.
asm.h.w	Permission allows user to create or update hierarchy type asset
asm.loc.w	Permission allows user to create or update locations.
asm.loc.d	Permission allows user to delete locations.
asm.rh.d	Permission allows user to delete root assets.
asm.rh.w	Permission allows user to create or update root assets.
dl.ds.r	Permission allows user to read data staging
dl.ds.w	Permission allows user to write data staging
dl.ds.d	Permission allows user to delete data staging
dl.de.r	Permission allows user to read event subscription
dl.de.w	Permission allows user to create event subscription
dl.de.d	Permission allows user to delete event subscription
dl.da.r	Permission allows user to read data access
dl.da.w	Permission allows user to create cross account
dl.da.d	Permission allows user to delete data access
dl.dat.r	Permission allows user to read data access token
dl.tsi.w	Permission allows user to create time series import
dl.tsi.r	Permission allows user to read time series imports
dl.tsi.d	Permission allows user to delete time series import jobs
em.c	Permission allows user to create events in Event Management
em.et.r	Permission allows user to read event types in Event Management
em.r	Permission allows user to read events in Event Management
em.u	Permission allows user to update events in Event Management
em.d	Permission allows user to delete events in Event Management
em.et.c	Permission allows user to create event types in Event Management

Scopes	Description
em.et.u	Permission allows user to update event types in Event Management
em.et.d	Permission allows user to delete event types in Event Management
emds.ent.r	Permission allows user to read entities via EntityMasterDataService
iot.fil.d	Permission allows user to delete file
iot.fil.r	Permission allows user to read file
iot.fil.w	Permission allows user to write file
iot.tim.d	Permission allows user to delete time series
iot.tim.r	Permission allows user to read time series
iot.tim.w	Permission allows user to write time series
iot.tsa.r	Permission allows user to read time series aggregations
as.kc.u	Permission allows user to use KPI Calculation API.
nose.ac	Permission allows user to grant access to administration console.
nose.se	Permission allows user to grant access to send e-mail message, Push Notification and SMS.
as.ea.u	Permission allows user to use Event Analytics API.
as.sc.u	Permission allows user to use Signal Calculation API.
as.sv.u	Permission allows user to use Signal Validation API.
tm.t.r	Permission allows user to read environments.
tm.st.r	Permission allows user to read subtenants.
uts.rc	Permission allows user to grant access to report console
uts.su	Permission allows user grant access to send usage information
uts.ri	Permission allows user grant user to request usage information
uts.qi	Permission allows user grant access to quota information
vfc.a	Permission allows user to impersonate
vfc.r	Permission allows user to read projects/tabs/nodes
vfc.w	Permission allows user to create or update tabs/nodes
pl.de.r	Permission allows user to list folder contents and download data

Scopes	Description
pl.de.w	Permission allows user to upload and delete data. It implies the pl.de.r
sdi.dip.r	Permission allows user to grant access to read the job status for data ingest process
sdi.dip.w	Permission allows user to grant access to start data ingest process
sdi.dqp.d	Permission allows user to grant access to delete a data query
sdi.dqp.e	Permission allows user to grant access to create or get query execution jobs
sdi.dqp.r	Permission allows user to grant access to read data query result
sdi.dqp.w	Permission allows user to grant access to create a data query
sdi.dqp.x	Permission allows user to grant access to execute a data query
sdi.reg.d	Permission allows user to grant access to delete data registry information
sdi.reg.r	Permission allows user to grant access to read data registry information
sdi.reg.w	Permission allows user to grant access to create or update data registry information
sdi.smd.d	Permission allows user to grant access to delete a semantic data model
sdi.smd.r	Permission allows user to grant access to read a semantic data model
sdi.smd.w	Permission allows user to grant access to create a semantic data model
oc.pr.r	Permission allows user to read data from Opcenter Intelligence
oc.pr.w	Permission allows user to write data to Opcenter Intelligence

- You can add "User secrets" to use in URL with name and value pair to secure requested service. These secret parameters are not visible after they are got added.
- You can add the custom user defined data using "Secret headers" via HTTP-Headers.

Switch

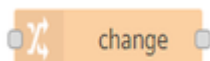


The "switch" node routes messages based on the values set in the node properties. The node evaluates each of the values in the defined rules and thereafter forwards the message to its corresponding output once a match is found.

You can use this node to:

- Check the msg.payload, simultaneously analyze the property and depending on the value set (true or false) and decide to route the messages.
- To evaluate the rules against an individual message property or the result of a JSONata expression.

Change



The "change" node sets, changes, deletes or moves properties of a message as required. The node can specify multiple rules that will be applied in the order they are defined.

You can use this node to:

- Set a property.
- Search and replace parts of a property.

- Delete a property.
- Move or rename a property.

Range

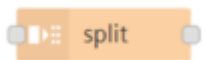


The "range" node maps a numeric value to a different range.

You can use this node to:

- Linear scaling of the received value.
- Map values to a new range.

Split

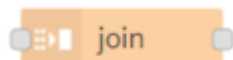


The "split" node splits a message into a series of messages. The property set of the node will define how messages can be split from the parent message.

You can use this node to:

- Create flows which perform common actions across a sequence of messages.

Join

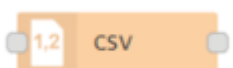


The "join" node joins a sequence of messages into a single message. The property set of the node will define how a message is joined into a single message.

You can use this node to:

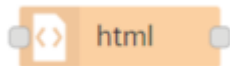
- Create flows which perform common actions across a single message.

Csv



The "csv" nodes converts and represents the csv (comma-separated values) format into its javascript object and vice versa.

Html



The "html" extracts elements from an html document held in msg.payload using a CSS selector.

Json



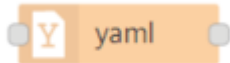
The "json" nodes converts and represents the json format into its javascript object and vice versa.

Xml



The "xml" nodes converts and represents the xml (Extensible Markup Language) format into its javascript object and vice versa.

Yaml



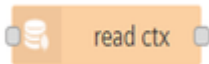
The "yaml" nodes converts and represents the yaml (Yaml Ain't Markup Language) format into its javascript object and vice versa.

Combine



The "combine" node combines several messages into one and then sends it by default as an array of particular payloads. It also has the mode to combine multiple timeseries payloads into one array too.

Read-context



The "read-context" node reads the values from the context with the associated context type and key.

Context type	Description
Flow	With this context type, you can access the values only from the current tab.

Context type	Description
Global	With this context type, you can access the values from any tab.
Tenant	With this context type, all the environment users can access the values from any tab.

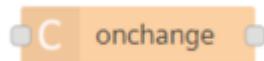
Store-context



The "store-context" node stores the values into the context with the specified context type and key. If the Time-To-Live (TTL) parameter is set, the value will disappear from the context after the given time period. The time period value must be between 1 and 15811200 seconds (183 days).

Context type	Description
Flow	With this context type, you can store the values only from the current tab.
Global	With this context type, you can store the values from any tab.
Tenant	With this context type, all the environment users can store the values from any tab.

OnChange

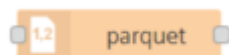


The "OnChange" node caches the payload of the last received message internally and compares it with the payload of a newly arrived message. If there are differences between the payloads, the node forwards the message.

You use this node to:

- React on changes and trigger the actions, for example send an email when the time series data has changed.

Parquet



The "parquet" node converts the array format into parquet format and also it can read parquet file to generate output as JSON array. The following payload types will be converted using this node into array: buffer, base64 string or array of numbers. The following payload types will be converted into parquet file: array of objects.

Delay



The "delay" node delays each message passing through the node or limits the rate at which they can pass. The message delay is set in milliseconds to the message. This option only applies if the node is configured to allow the message to provide the delay interval.



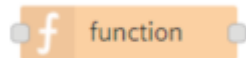
The "Combine and OnChange" node are exclusively available only in "Visual Flow Creator"

6.4 Usages of function nodes

The following shows the specific "function" nodes with their respective functions. These are programmable and user defined nodes which can be customized.

For more information about other function node, refer to [Node-RED documentation](#).

Function



The "function" node is used to run JavaScript code against the message object.

You can use this node to:

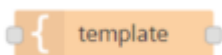
- Design your JavaScript code here which will run against the messages passed. This in turn will return zero or more messages to downstream nodes for processing.
- Design functions in the node properties and save them in the library. You can reuse the functions whenever required.

You can write your Javascript code in the editor.



1. The global context has name "glob".
2. setTimeout, setInterval are not available for use.

Template



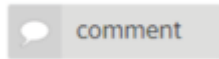
The "template" node sets a property based on the user defined template. By default, the template uses the "Mustache" format. The "Mustache" format can be switched off if required.

You can use this node to:

- Design the template and set its property.

Mustache can be used for HTML, config files and source codes. It works by expanding tags in a template using values provided in an object.

Comment

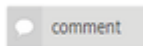


The "comment" node allows you to add comments in the flows that you have created.

The editor will accept information in the MarkDown syntax.

You can use this node to:

- Add formatted text to the created flows that will be shown in the info tab.
- Sticky note is used to select the type of font as given below list:
 - Do not show: It will not show the font on the node in the working area.



- Big font: It will show the font on the node with bigger in size in the working area.



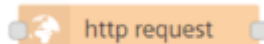
Small font: It will show the font on the node with smaller in size in the working

- area.



Sticky note messages will not allow MarkDown format.

http Request



The "http request" node sends HTTP requests and returns the response.

You can use this node to:

- Access the data from external services.
- To use services, activate "Use Industrial IoT service" option and configure the requested service path in the path field.
- If "Extended scopes" option is selected then, the request will be processed with a token that has more scopes for a normal user. For example, creating or deleting assets. Admin can create or modify the scopes for the normal user.

Scopes	Description
atm.appt.d	Permission allows user to delete aspect types.
atm.appt.w	Permission allows user to create or update aspect types.
atm.appt.r	Permission allows user to read aspect types.
atm.fa.w	Permission allows user to assign files to asset types.
atm.fa.d	Permission allows user to delete file assignments.
atm.r	Permission allows user to read asset types.
atm.w	Permission allows user to create or update asset types.
atm.d	Permission allows user to delete asset types.
as.ad.u	Permission allows user to use Anomaly Detection API - without batch endpoints.
as.adb.u	Permission allows user to use Anomaly Detection Batch API.
as.kc.u	Permission allows user to use KPI Calculation API.
as.sa.fft	Permission allows user to use Spectrum Analysis API.
as.tp.u	Permission allows user to use Trend Prediction API.
asm.c	Permission allows user to create assets.
asm.d	Permission allows user to delete assets.
asm.m	Permission allows user to move assets.
asm.r	Permission allows user to read assets.
asm.u	Permission allows user to update assets.
asm.fr	Permission allows user to read files.
asm.fw	Permission allows user to create or update files.
asm.fd	Permission allows user to delete files.
asm.fa.w	Permission allows user to assign files to assets.
asm.fa.d	Permission allows user to delete files assignments.
asm.h.d	Permission allows user to delete hierarchy type assets.
asm.h.w	Permission allows user to create or update hierarchy type asset
asm.loc.w	Permission allows user to create or update locations.

Scopes	Description
asm.loc.d	Permission allows user to delete locations.
asm.rh.d	Permission allows user to delete root assets.
asm.rh.w	Permission allows user to create or update root assets.
dl.ds.r	Permission allows user to read data staging
dl.ds.w	Permission allows user to write data staging
dl.ds.d	Permission allows user to delete data staging
dl.de.r	Permission allows user to read event subscription
dl.de.w	Permission allows user to create event subscription
dl.de.d	Permission allows user to delete event subscription
dl.da.r	Permission allows user to read data access
dl.da.w	Permission allows user to create cross account
dl.da.d	Permission allows user to delete data access
dl.dat.r	Permission allows user to read data access token
dl.tsi.w	Permission allows user to create time series import
dl.tsi.r	Permission allows user to read time series imports
dl.tsi.d	Permission allows user to delete time series import jobs
em.c	Permission allows user to create events in Event Management
em.et.r	Permission allows user to read event types in Event Management
em.r	Permission allows user to read events in Event Management
em.u	Permission allows user to update events in Event Management
em.d	Permission allows user to delete events in Event Management
em.et.c	Permission allows user to create event types in Event Management
em.et.u	Permission allows user to update event types in Event Management
em.et.d	Permission allows user to delete event types in Event Management
emds.ent.r	Permission allows user to read entities via EntityMasterDataService
iot.fil.d	Permission allows user to delete file

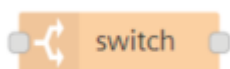
Scopes	Description
iot.fil.r	Permission allows user to read file
iot.fil.w	Permission allows user to write file
iot.tim.d	Permission allows user to delete time series
iot.tim.r	Permission allows user to read time series
iot.tim.w	Permission allows user to write time series
iot.tsa.r	Permission allows user to read time series aggregations
nose.ac	Permission allows user to grant access to administration console.
nose.se	Permission allows user to grant access to send e-mail message, Push Notification and SMS.
tm.t.r	Permission allows user to read environments.
tm.st.r	Permission allows user to read subtenants.
uts.rc	Permission allows user to grant access to report console
uts.su	Permission allows user grant access to send usage information
uts.ri	Permission allows user grant user to request usage information
uts.qi	Permission allows user grant access to quota information
vfc.a	Permission allows user to impersonate
vfc.r	Permission allows user to read projects/tabs/nodes
vfc.w	Permission allows user to create or update tabs/nodes
pl.de.r	Permission allows user to list folder contents and download data
pl.de.w	Permission allows user to upload and delete data. It implies the pl.de.r
sdi.dip.r	Permission allows user to grant access to read the job status for data ingest process
sdi.dip.w	Permission allows user to grant access to start data ingest process
sdi.dqp.d	Permission allows user to grant access to delete a data query
sdi.dqp.e	Permission allows user to grant access to create or get query execution jobs
sdi.dqp.r	Permission allows user to grant access to read data query result
sdi.dqp.w	Permission allows user to grant access to create a data query
sdi.dqp.x	Permission allows user to grant access to execute a data query

Scopes	Description
sdi.reg.d	Permission allows user to grant access to delete data registry information
sdi.reg.r	Permission allows user to grant access to read data registry information
sdi.reg.w	Permission allows user to grant access to create or update data registry information
sdi.smd.d	Permission allows user to grant access to delete a semantic data model
sdi.smd.r	Permission allows user to grant access to read a semantic data model
sdi.smd.w	Permission allows user to grant access to create a semantic data model

- You can add "User secrets" to use in URL with name and value pair to secure requested service. These secret parameters are not visible after they are got added.
- You can add the custom user defined data using "Secret headers" via HTTP-Headers.

The screenshot shows the 'Edit http request node' configuration interface. It includes a 'node properties' section with fields for Name, Method (set to GET), and URL (set to http://). There are checkboxes for 'Use MindSphere service' and 'Use basic authentication'. Below these are sections for 'User secrets' and 'Secret headers', each with a table for defining Name and Value pairs. The 'Return' type is set to 'a UTF-8 string' and the 'Timeout' is set to 'seconds'. At the bottom, there is a 'port labels' section.

Switch

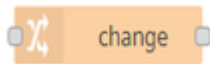


The "switch" node routes messages based on the values set in the node properties. The node evaluates each of the values in the defined rules and thereafter forwards the message to its corresponding output once a match is found.

You can use this node to:

- Check the `msg.payload`, simultaneously analyze the property and depending on the value set (true or false) and decide to route the messages.
- To evaluate the rules against an individual message property or the result of a JSONata expression.

Change



The "change" node sets, changes, deletes or moves properties of a message as required. The node can specify multiple rules that will be applied in the order they are defined.

You can use this node to:

- Set a property.
- Search and replace parts of a property.
- Delete a property.
- Move or rename a property.

Range



The "range" node maps a numeric value to a different range.

You can use this node to:

- Linear scaling of the received value.
- Map values to a new range.

Split

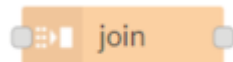


The "split" node splits a message into a series of messages. The property set of the node will define how messages can be split from the parent message.

You can use this node to:

- Create flows which perform common actions across a sequence of messages.

Join



The "join" node joins a sequence of messages into a single message. The property set of the node will define how a message is joined into a single message.

You can use this node to:

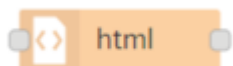
- Create flows which perform common actions across a single message.

Csv



The "csv" nodes converts and represents the csv (comma-separated values) format into its javascript object and vice versa.

Html



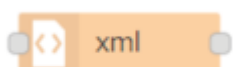
The "html" extracts elements from an html document held in msg.payload using a CSS selector.

Json



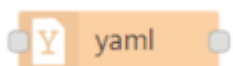
The "json" nodes converts and represents the json format into its javascript object and vice versa.

Xml



The "xml" nodes converts and represents the xml (Extensible Markup Language) format into its javascript object and vice versa.

Yaml



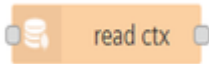
The "yaml" nodes converts and represents the yaml (Yaml Ain't Markup Language) format into its javascript object and vice versa.

Combine



The "combine" node combines several messages into one and then sends it by default as an array of particular payloads. It also has the mode to combine multiple timeseries payloads into one array too.

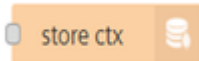
Read-context



The "read-context" node reads the values from the context with the associated context type and key.

Context type	Description
Flow	With this context type, you can access the values only from the current tab.
Global	With this context type, you can access the values from any tab.
Tenant	With this context type, all the environment users can access the values from any tab.

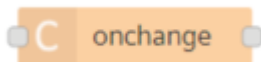
Store-context



The "store-context" node stores the values into the context with the specified context type and key. If the Time-To-Live (TTL) parameter is set, the value will disappear from the context after the given time period. The time period value must be between 1 and 15811200 seconds (183 days).

Context type	Description
Flow	With this context type, you can store the values only from the current tab.
Global	With this context type, you can store the values from any tab.
Tenant	With this context type, all the environment users can store the values from any tab.

OnChange



The "OnChange" node caches the payload of the last received message internally and compares it with the payload of a newly arrived message. If there are differences between the payloads, the node forwards the message.

You use this node to:

- React on changes and trigger the actions, for example send an email when the time series data has changed.

Parquet



The "parquet" node converts the array format into parquet format and also it can read parquet file to generate output as JSON array. The following payload types will be converted using this node into array: buffer, base64 string or array of numbers. The following payload types will be converted into parquet file: array of objects.

Delay



The "delay" node delays each message passing through the node or limits the rate at which they can pass. The message delay is set in milliseconds to the message. This option only applies if the node is configured to allow the message to provide the delay interval.



The "Combine and OnChange" node are exclusively available only in "Visual Flow Creator"

6.5 Using context nodes

Example scenario

You can store and read the values using context nodes with the associated context type and key.



You can store and retrieve data from context variables using specific nodes. The context variables is not yet supported for region Europe 2.

Objective

To store and read the values using context nodes.

Procedure to store into the context

To store the values into the context using store-context node, follow these steps:

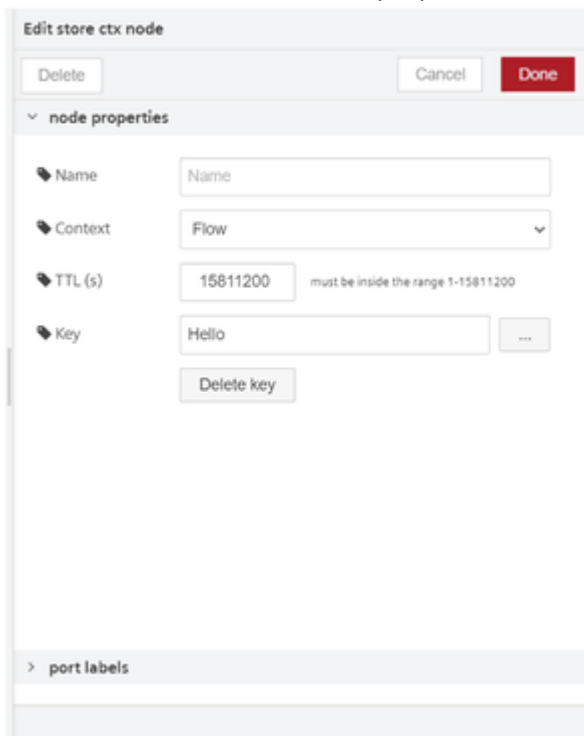
1. Design the flow as shown below:




2. Edit the inject node properties:

- String: Welcome to VFC

3. Edit the store-context node properties:



- Context: Select "Flow"
- Key: Enter "Hello"

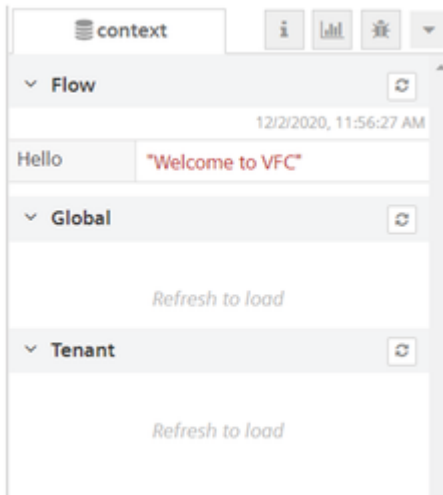






Click 138144714635.png to store or replace the values in the existing key.

4. Save and execute the flow.

Result

The values are stored into the context successfully. The output is displayed in the context data tab:

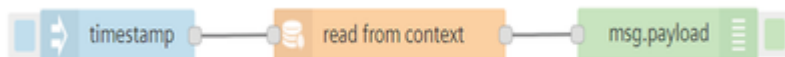


Symbol	Description
	Refresh to display the key and value stored into the context.
	Copy the value.
	Refresh to display the updated key and value.
	Deletes the key and value from the context.

Procedure to read from the context

To read the values from the context using read-context node, follow these steps:

1. Design the flow as shown below:

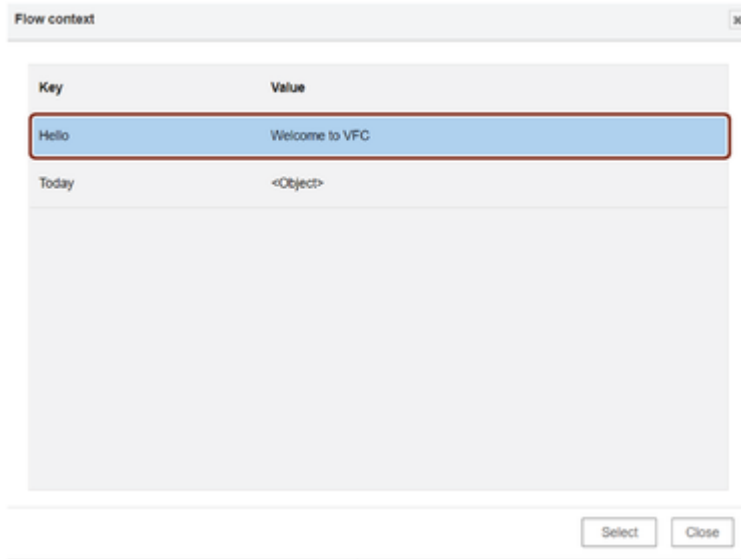


2. Edit the read-context node properties:

- Context: Select "Flow"

3. Click .

4. Select the Key.



5. Save and execute the flow.

Result

The output is displayed in the debug tab:

```
8/18/2020, 4:36:19.951 PM node: 5e3319b9.f001f8  
msg: string[14]  
"Welcome to VFC"
```

6.6 Using parquet node

Example scenario

You can get the array object into parquet format.

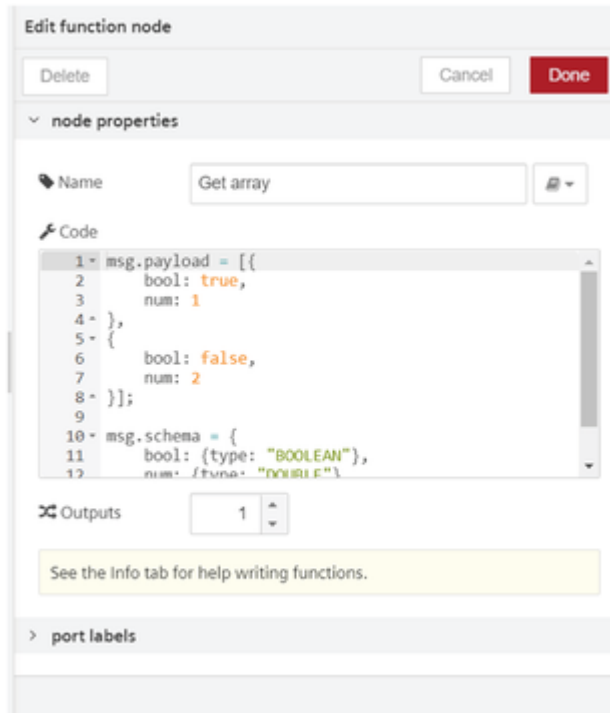
Procedure

To get the array object into parquet format using parquet node, follow these steps:

1. Design the flow as shown below:



2. Edit the function node properties:



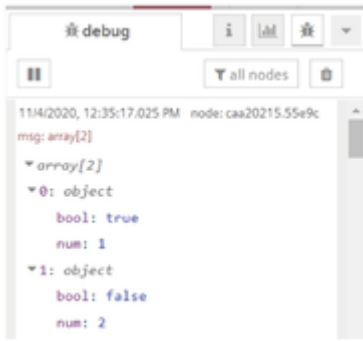
- Name: Get array
- Code :

```
msg.payload = [{
  bool: true,
  num: 1
},
{
  bool: false,
  num: 2
}];
msg.schema = {
  bool: {type: "BOOLEAN"},
  num: {type: "DOUBLE"},
};
return msg;
```

3. Save and execute the flow.

Result

The output will be displayed in the debug tab.



6.7 Usages of analytics nodes

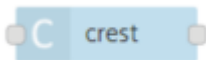
The analytics nodes are helpful to design flows which have array or time series data. The maximum array input size is limited to 2000.

Extrapolate



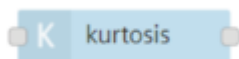
The node returns the input arrays or time series data. The return value also includes the extrapolated datapoints.

Crest



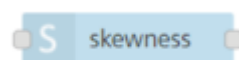
The crest node computes the crest factor for an inputted array values and time series data.

Kurtosis



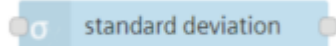
The kurtosis node computes the kurtosis for the input of array values and time series data.

Skewness



The skewness node computes the skewness for the input of array values and time series data.

Standard deviation



The node computes the sample or population standard deviation for the values referenced by the defined input parameter of the array or time series.

Variance



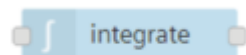
The skewness node computes the variance for the input of array values and time series data.

Derive



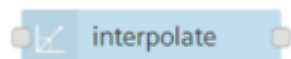
For inputs of array values and time series data, derivative is calculated for a set of data points.

Integrate



The nodes compute the integral for the input of array values and time series data.

Interpolate



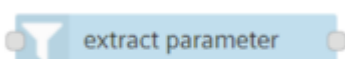
The node interpolates values referenced by one parameter of the time series using linear or cubic spline interpolation.

FFT



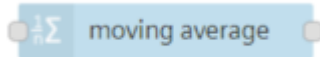
The node returns an array with the parameter values of the defined parameter, discarding other time series information.

Extract parameter



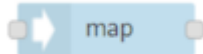
The node returns array values which are defined to be extracted in the parameter.

Moving average



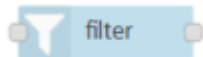
For inputs of array values and time series data, moving average is calculated for a set of data points.

Map



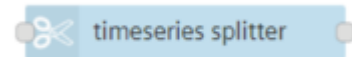
The map node maps time series data values to new values.

Filter



For a given array or time series data, the node filters and passes the filtered values to the output.

Timeseries splitter



For a given array or time series data, the node splits a message payload into several small messages if the gaps between any two time series is greater than the given interval

6.8 Using analytics nodes

You can use analytics nodes to analyze and process time series data from Industrial IoT with standardized analytics methods. The assignable parameters control the analytics node.

Example scenario

The owner of the logistics center of the smart city would like to view the energy consumption of its cranes over different time periods: hour, day, week and month.

The previously created flow allows the power peaks of a crane to be identified.

Objective

A distribution of the energy demand is to be determined and the average displayed.

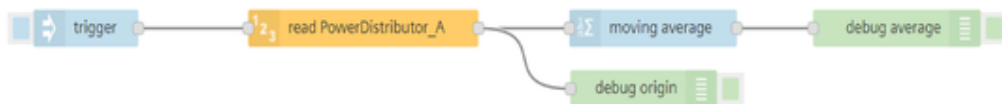
Requirement

- The crane is connected to Industrial IoT and collects the energy data.
- The flow for determining the energy consumption of the cranes has already been created in Visual Flow Creator.

Procedure

To use an analytics node, follow these steps:

1. Open the flow for determining the energy consumption of the cranes.
2. Move the "moving average" node to the working area using drag-and-drop.
3. Insert the "moving average" node in the flow for determining the energy consumption of the



crane.

4. To activate the flow, click the blue button to the left of the ""timestamp"" node.

Result

The existing flow uses the analytics node "moving average" to generate a moving average of the energy consumption of the crane.

The steep edges of the energy consumption acquired become flatter and reflect a consumption profile that can be interpreted much more effectively based on a lower resolution.

The following graphic shows the graph without smoothed data:

1/12/2018, 4:20:56 PM node: debug origin

msg.payload : array[31]

▸ [object, object, object, object, object, object, object, object, object, object, object ...]

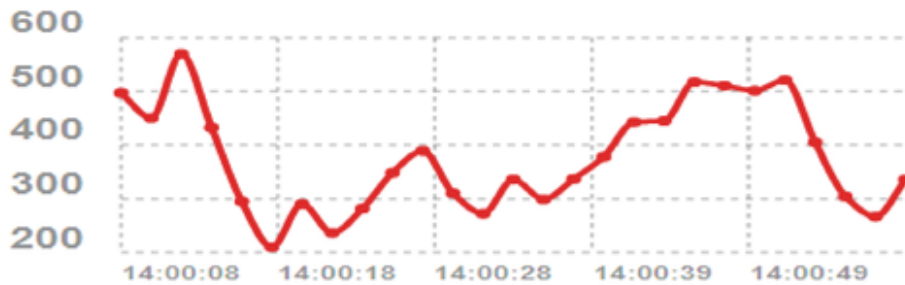


The following graphic shows the graph with smoothed data:

1/12/2018, 4:20:56 PM node: debug average

msg.payload : array[27]

▶ [object, object, object, object, object, object, object, object, object, object, object ...]



6.9 Storage node library

The following listing shows the specific "storage" nodes and their respective functions:

Postgres node



This node allows reading and writing data from a Postgres database. You can set the database queries in the properties of the node. The node delivers the result of the query in msg.payload or in the configuration with optional query parameters in msg.queryParameters. The parameters in the query must be specified as \$fieldName. For more information, refer to [node-postgres-named](#).

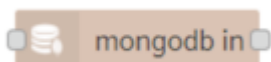
You use this node to:

- Query data form a Postgres database.

Restriction

- At the moment you cannot connect Cloud Foundry Backing Services Instances with Visual Flow Creator.
- You cannot access Cloud-Foundry based databases which belong to third party services.

Mongodb in node



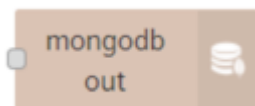
This node allows reading data from a MongoDB database.

You can use the following operations:

- **find:** Returns the contents of the database.
- **count:** Returns a count of the number of documents in a collection.
- **aggregate:** Provides access to the aggregation pipeline.

For more information on Node-RED, refer to [MongoDB section](#).

Mongodb out node



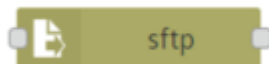
This node allows writing data into a MongoDB database.

You can use the following operations:

- **save:** Will update an existing object or insert a new object if one does not already exist.
- **insert:** Inserts a new object
- **update:** Will modify an existing object or objects
- **remove:** Removes objects that match the query passed in on msg.payload.

For more information on Node-RED, refer to [MongoDB section](#).

SFTP node



This node allows to read and write files from or to SFTP server.

You can use the following operations:

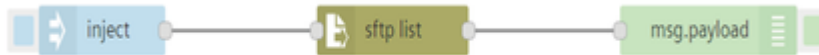
- **List Directory:** It will list the directory content and returns an array of the found files.
- **Get:** It will get a file from the SFTP server as Buffer object.
- **Put:** It will create the defined directory on the SFTP server in the specified working directory.
- **Delete File:** It will delete a file from the SFTP server as per the specified filename.
- **Make Directory:** It will create a directory on the SFTP server as per the specified working directory.
- **Remove Directory:** It will remove the defined directory and all sub-directories on the SFTP server in the specified working directory.

For more information in Node-RED, refer to [SFTP](#).

Example

To get the list of the directories from the SFTP server, follow these steps:

1. Create the flow as shown below:



2. Edit sftp node and enter the details:

Edit sftp node

Delete Cancel Done

node properties

Name

Host Port

Username

Password

Operation

Working Directory

port labels

3. Save and execute the flow.

Result

The output is displayed in the message payload:

```
5/3/2021, 5:12:31.433 PM node: fbdf5f65.bfcd1
msg: array[2]
  array[2]
    0: object
      type: "d"
      name: "upload"
      size: 0
      modifyTime: 1620042842000
      accessTime: 1620042842000
      rights: object
        owner: 0
        group: 0
    1: object
      type: "d"
      name: "download"
      size: 0
      modifyTime: 1616211000000
      accessTime: 1616211000000
      rights: object
        owner: 0
        group: 0
```

6.10 Array node library

Usages of Array nodes

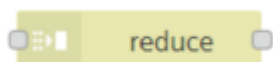
Array nodes represent function methods like filter, map, find and reduce for JavaScript arrays. The following listing shows the array nodes and their respective functions:

find node



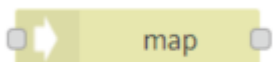
The "find" node will return the value of the first element of an array that fulfills the requirements of the included testing function.

reduce node



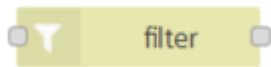
The "reduce" node reduces an array to a single value. The node will apply a function against an accumulator and each element in the array (from left to right).

map node



The "map" node modifies an array with the implemented function. As a result you will get a new array.

filter node



The "filter" node will return a new array that contains all elements that pass the conditions of the implemented function.

Using array nodes

Example scenario

A simple array is defined with a sine signal wave. The scenario is to filter and map values.

Objective

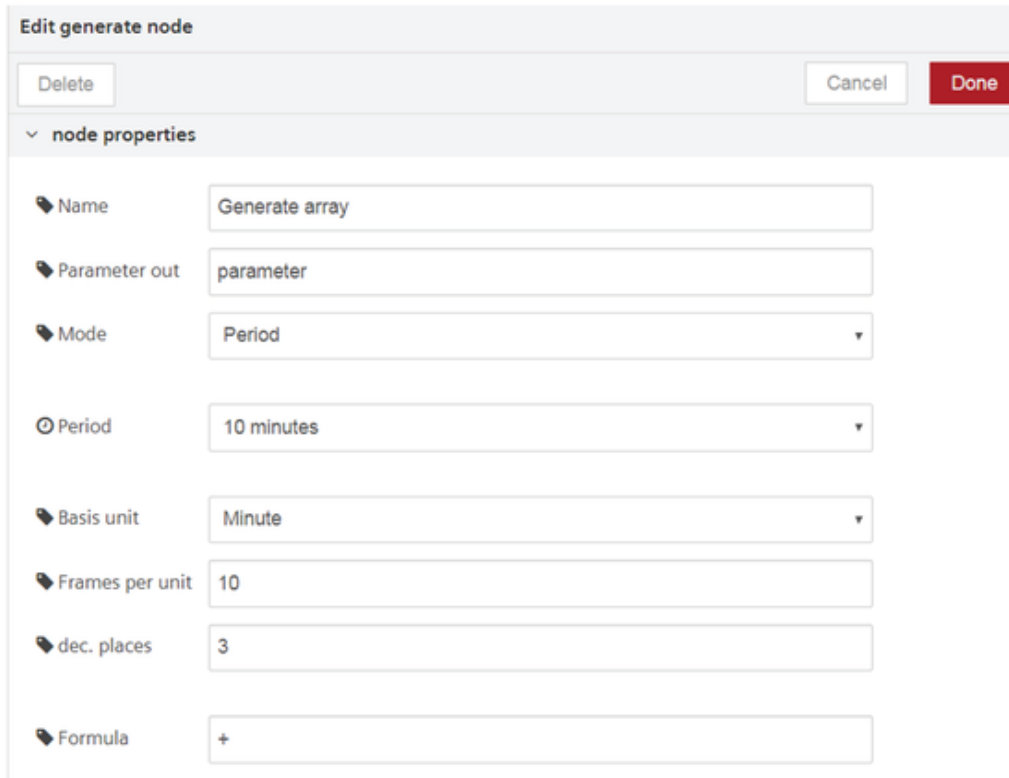
To filter the elements which are greater than 0 from an array and map them by adding an added value of 100. Also, it is required to display the sine wave generated in the message payload.

Requirements

- 1 input timestamp node
- 1 IIoT generate node
- 1 array filter node
- 1 array map node
- 1 output debug node

Procedure

1. Configure generated node as shown below:



Edit generate node

Delete Cancel Done

node properties

Name Generate array

Parameter out parameter

Mode Period

Period 10 minutes

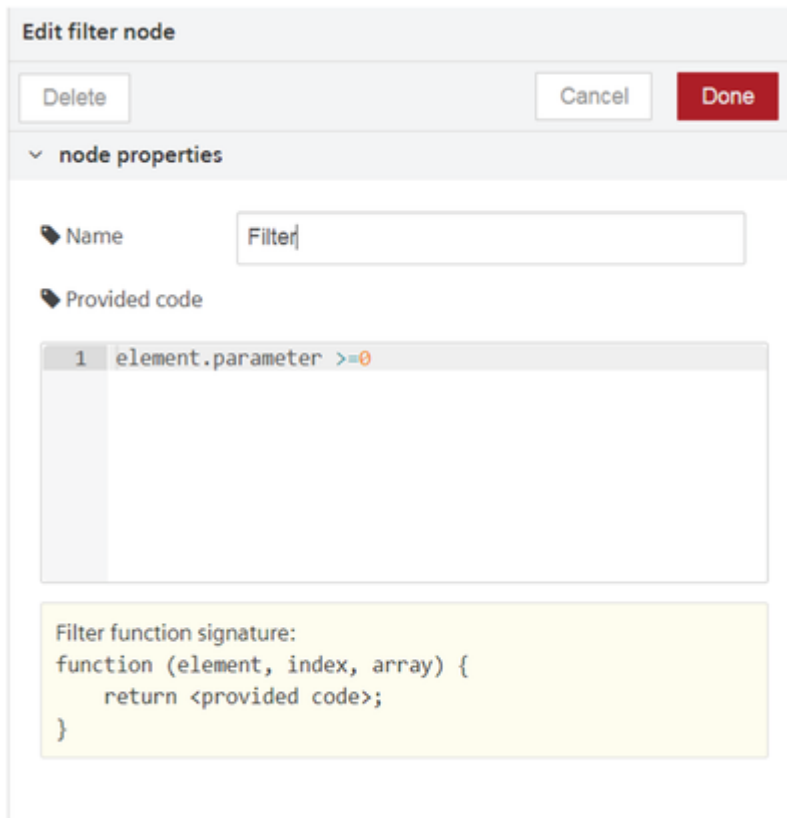
Basis unit Minute

Frames per unit 10

dec. places 3

Formula +

2. Filter the array filter node with elements greater than 0.



Edit filter node

Delete Cancel Done

node properties

Name Filter

Provided code

```
1 element.parameter >=0
```

Filter function signature:
function (element, index, array) {
 return <provided code>;
}

The function is already defined in the filter node. You just have to add the condition in the "provided code" section.

3. Configure the array map node with an added value of 100.

Edit map node

Delete Cancel Done

node properties

Name: Map the array

Provided code

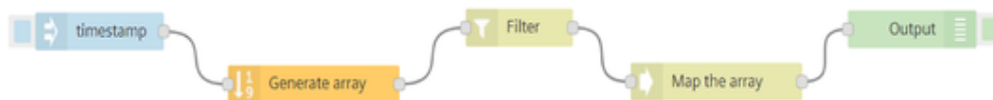
```
1 {  
2   newparameter: element.parameter + 100,  
3   _time: element._time  
4 }
```

Multiline function body

Map function signature:
function (element, index, array) {
 return <provided code>;
}

The function is already defined in the filter node. You just have to add the condition in the "provided code" section.

4. Connect all the nodes as shown below:

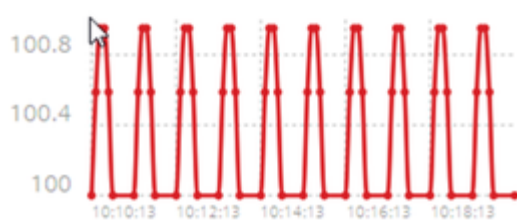


5. Inject the input node.

Result

The values are mapped in the objects and the necessary sine wave has also been generated in the message payload:


```
1/28/2019, 9:14:52.256 AM node: Output
msg: array[61]
▼ array[61]
▼ [0 ... 9]
  ▼ 0: object
    newparameter: 100
    _time: "2019-01-28T04:40:13.000Z"
  ▼ 1: object
    newparameter: 100.588
    _time: "2019-01-28T04:40:19.000Z"
  ▼ 2: object
    newparameter: 100.951
    _time: "2019-01-28T04:40:25.000Z"
  ▼ 3: object
    newparameter: 100.951
    _time: "2019-01-28T04:40:31.000Z"
  ▼ 4: object
    newparameter: 100.588
    _time: "2019-01-28T04:40:37.000Z"
  ▶ 5: object
  ▶ 6: object
  ▶ 7: object
  ▶ 8: object
  ▶ 9: object
  ▶ [10 ... 19]
  ▶ [20 ... 29]
  ▶ [30 ... 39]
  ▶ [40 ... 49]
  ▶ [50 ... 59]
  ▶ [60 ... 60]
```



6.11 Simple Anomaly node library

Introduction to simple anomaly nodes

The anomaly nodes help to detect anomalies in a data set.

The simple anomaly nodes in Visual Flow Creator are of two types:

- Training model

- Reasoning model

!!! note "Region deviation Support for "Simple Anomaly nodes" is in progress for Region Europe 2.

Terms and definitions

Cluster: A cluster is a high density region in a defined space. Each cluster is a set of data points which defines a dense region.

Anomaly points: The number of anomaly points available outside a data cluster.

Neighborhood: The neighborhood of a point is a set of all points that are within a distance set by the neighborhood parameter.

Epsilon (ϵ or eps): The epsilon (ϵ) defines the distance defined of neighborhood around a certain point in the selected algorithm (Euclidean, Manhattan, Chebychev).

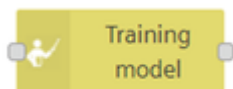
Anomaly extent: Distance between an anomaly point and its centre point (ϵ) of the cluster.

Anomalies are minimal for lesser anomaly extents. For greater distances, anomalies are larger.

Algorithms: Euclidean: The Euclidean distance is defined by a straight line between two points.

Manhattan: The Manhattan distance measure algorithm finds its application on high dimensional vectors. The sum of the absolute differences among their coordinates is defined as the distance between two points. **Chebychev:** The maximum distance from the centre of a cell to its adjacent cells centre is defined as the Chebychev distance between two points.

Training model node



The training model node defines a new model of some data sets in Visual Flow Creator.

The data set information gets configured as read time series node with one condition - you can select only one aspect. The data set is defined by asset/ aspect/ variable. The data set can be selected using "Select asset aspect" selection dialog box.

Training model created will be stored in Industrial IoT exchange storage.

Training model node properties

You can configure the node by editing its node properties.

Edit Training model node

▼ **node properties**

📌 **Name**

📌 **Model name**

🗨 **Description**

1	epsilon 0.5
2	points 5

📄 **Topic**

📌 **Topic Summary**

Asset
a1ac2fa99bd94d95bef997e413eaab70 (CPU)
Aspect
CoreTemperature
Variable
Int

📌 **Mode** ▼

🕒 **Timezone** ▼

🕒 **From**

🕒 **To**

📌 **Epsilon**

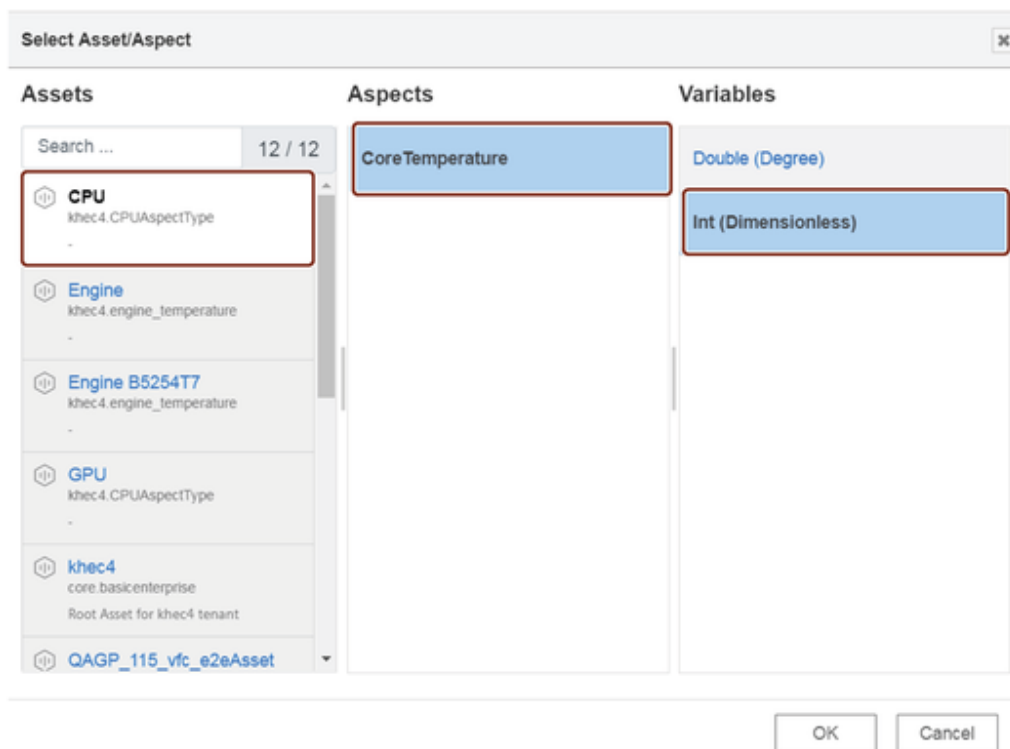
📌 **Min. Points per Cluster**

📌 **Distance Measure Algorithm** ▼

Field Name	Description	Mandatory
Name	Select a display name for the node.	No
Model Name	Name the new training model.	Yes
Description	Enter the description.	No
Topic	Import the required time series data from the "Topic" field. The menu will redirect to select read time series data with asset /aspect variable.	Yes

Field Name	Description	Mandatory
Topic Summary	Each asset will have aspects which are defined by variables. Select the required variable of the selected aspect of its asset to import the data in the "Training model" node properties. This will autofill the "Topic Summary" field.	Yes
Mode	Select the mode from the drop-down menu. "Mode" is either defined by period of time or an interval.	Yes
Period		
Period	Select the time period from the drop-down menu. The time period drop-down has values defined in minutes, hours, days and weeks.	Yes
Offset	Select the offset value for balancing the effects. The offset values are defined in seconds.	Yes
Interval		
Timezone	Select the required time zone from the drop-down menu.	Yes
From	Select the start date and time from the calendar and time menu.	Yes
To	Select the end date and time from the calendar and time menu.	Yes
Epsilon	Set the ϵ for the new model.	Yes
Min. points per cluster	Define the minimum number of points which are in each cluster.	Yes
Distance Measure Algorithm	Select the required distance measure algorithm from the drop-down. Three algorithms are available: - Euclidean - Manhattan - Chebychev	Yes

When you set the topic in "Select Asset/ Aspect" dialog box, select the necessary asset, aspect and its variable as shown below:



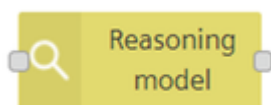
Jobs run involved

While injecting a timestamp to the "training model" node, a set of batch jobs run in the following series:

- Fetching IoT data
- Create model job
- Export job and generating output

Reasoning model node

The reasoning model node is used for analyzing data for a stored training model.



Reasoning model node properties

You can configure the node by editing its node properties.

Edit Reasoning model node

▼ **node properties**

📌 Name

📌 Model name ...

📌 Topic ...

📌 Topic Summary

Asset

a1ac2fa99bd94d95bef997e413eaab70 (CPU)

Aspect

CoreTemperature

Variable

Double

📌 Mode ▼

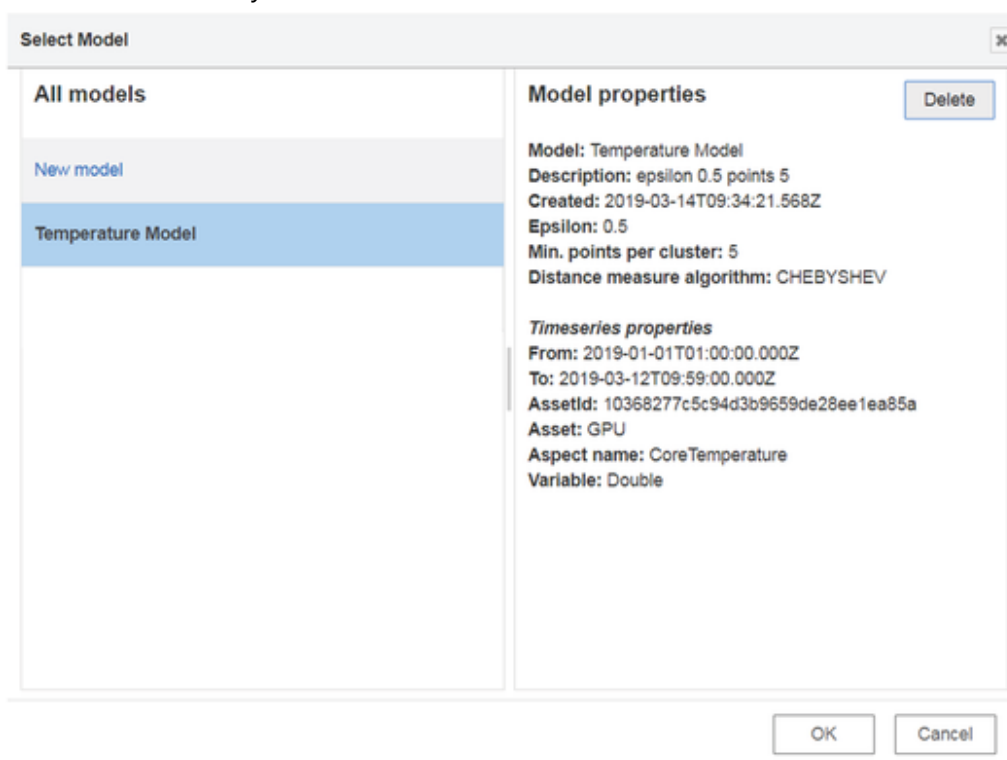
📌 Period ▼

📌 Offset ▼

Field Name	Description	Mandatory
Name	Select a display name for the node.	No
Model Name	Select a training model from the menu.	Yes
Description	Enter the description.	No
Topic	Import the required time series data from the "Topic" field. The menu will redirect to select read time series data with asset /aspect /variable. The import will capture the actual details of the model selected. However, the aspects and variables imported should be same as that selected for the "Model Name".	Yes
Topic Summary	Each asset will have aspects which are defined by variables. Select the required variable of the selected aspect of its asset to import the data in the "Training model" node properties. This will autofill the "Topic Summary" field.	Yes
Mode	Select the mode from the drop-down menu. "Mode" is either defined by period of time or an interval.	Yes
Period		
Period	Select the time period from the drop-down menu. The time period drop-down has values defined in minutes, hours, days and weeks.	Yes

Field Name	Description	Mandatory
Offset	Select the offset value for balancing the effects. The offset values are defined in seconds.	Yes
Interval		
Timezone	Select the required time zone from the drop-down menu.	Yes
From	Select the start date and time from the calendar and time menu.	Yes
To	Select the end date and time from the calendar and time menu.	Yes

When you select a model name, the data of the selected model will be imported to the reasoning model node for analysis.



The property details of the selected model will also be displayed to the user.

Jobs run involved

While injecting a timestamp to the "reasoning model" node, a set of batch jobs run in the following series:

- IoT import job creation
- Import the job model
- Apply the job model
- Export the job and save in the database

Using simple anomaly nodes

Example scenario

Configure a training model node. Analyze the anomalies in the model.

Objective

To analyze the anomalies in the newly created training model with the help of a reasoning model node.

Requirements

Import the data from Fleet Manager to create a training model.

Configuration of the training model is given below:

Field Name	Data
Name	CPU temp measure
Model Name	Temperature Model
Description	epsilon 0.5 points 5
Topic	10368277c5c94d3b9659de28ee1ea85a
Topic Summary	Asset: 10368277c5c94d3b9659de28ee1ea85a (GPU) Aspect: CoreTemperature Variable: Double
Mode	Interval
Period	
Timezone	UTC
From	2019/01/01 01:00:00
To	2019/03/12 09:59:00
Epsilon	0.5
Min. points per cluster	5
Distance Measure Algorithm	Chebychev

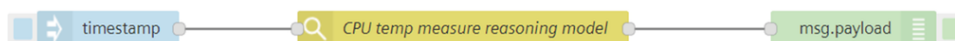
Procedure

1. Select a reasoning model node from the "simple anomaly" nodes section.
2. Configure the reasoning model node. The configuration of the node is given below:

Field Name	Data

Field Name	Data
Name	CPU temp measure reasoning model
Model Name	Temperature Model
Topic	a1ac2fa99bd94d95bef997e413eaab70
Topic Summary	Asset: a1ac2fa99bd94d95bef997e413eaab70 (CPU) Aspect: CoreTemperature Variable: Double
Mode	Period
Period	
Period	1 hour
Offset	2 seconds

3. Insert an input timestamp and a message payload to the reasoning model node to generate the output in the message payload.



The "Topic Summary" properties in training model node should match the "Topic Summary" properties of the reasoning model node. This means that aspects and the variables for both should be same else the output will conflict and result in a failure.

Result

Inject the timestamp to process the flow.

You get the following result in the message payload:



Some of the initial anomaly points are highlighted in the graph as well in the array description.

6.12 Integrated Data Lake nodes

Usage of Integrated Data Lake nodes

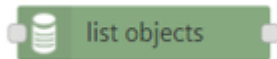
The Integrated Data Lake is a repository that allows you to store structured and unstructured data or objects in its native format as needed. It handles large data pools for which the schema and data requirements are not defined until the data is queried.

Integrated Data Lake nodes allows you to list, read, write, delete, subscribe and query the files or objects from Integrated Data Lake.



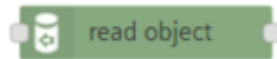
Integrated Data Lake nodes are only available on AWS.

List objects



The "list objects" node lists the files or objects from Integrated Data Lake and stores the data in the message payload. Additionally, you can filter the files or objects by "Sub-tenant" property. It is possible by the main environment.

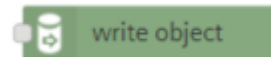
Read object



The "read object" node reads the file or object content from the specified path and place(s) the content in the message payload. The mode parameter defines the type of content to be read from the file or object. The read object mode parameters are given below:

- Object: Reads only the content of the file or object.
- Object+Metadata: Reads both content and metadata of the file or object.
- Metadata: Reads only the metadata of the file or object.

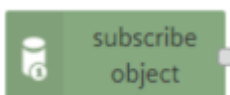
Write object



The "Write object" node writes or updates the file or object to the specified path and place(s) the content in the message payload. The mode parameter defines the type of content to be updated to the file or object. The write object mode parameters are given below:

- Object: Writes only the content to the file or object.
- Object+Metadata: Writes both content and metadata to the file or object.
- Metadata: Writes only the metadata to the file or object.

Subscribe object



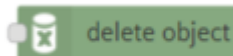
The "subscribe object" node will notify the modifications that took place in the specified path. The path should be specified in the edit properties dialog box to receive the notification in the message payload. For example, the message object is shown below:

```
msg: Object
  > { eventType:
    "MODIFY"
    , timestamp:
    "2020-04-29T06:20:18.032Z"
    , path:
    "/testdoc.txt"
  }
```



Metadata changes cannot be subscribed.

Delete object



The "delete object" node deletes the files or objects from the specified path. You must select the path from the "Select File" dialog box.



If the files or objects is deleted then the metadata will also be deleted.

Using Integrated Data Lake nodes

Example scenario

You can read the files or objects of the turbo engine assets by using "read object" node from Integrated Data Lake. The extracted data can be used for further analysis as per customer's requirement.

Objective

To read the files or objects from Integrated Data Lake.

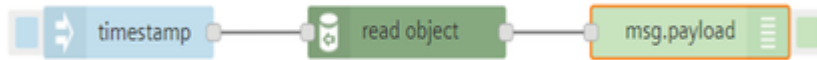
Requirements

- Inject node
- read object node
- Debug node


Procedure

Drag and drop the Inject, read object and debug nodes from the dashboard palette.

1. Interconnect the nodes:



2. Double click the read object node to edit the properties:

3. Click  to select the file.

4. Select the Mode.

5. Click "Done".

6. Save and deploy.



- Path should be specified for the "List object" and "Subscribe object" nodes.
- Select the path from the "Select File" dialog box for the "Read object", "Write object" and "Delete object" nodes from the file browser dialog box.
- Mode defining is applicable for "Read object" and "Write object" nodes.

Result

You can view the results in the message payload.

In this example the "read object" node reads the file or object from the selected path and displays the data in the message payload.

```
msg: string[20]
"Current temp 10.08°C"
```

6.13 SDI nodes

Usage of SDI nodes

The following listing shows the Semantic Data Interconnect (SDI) nodes and their respective functions:

SDI create query

Simple Anomaly node library

Introduction to simple anomaly nodes

The anomaly nodes help to detect anomalies in a data set.

The simple anomaly nodes in Visual Flow Creator are of two types:

- Training model
- Reasoning model

Terms and definitions

Cluster: A cluster is a high density region in a defined space. Each cluster is a set of data points which defines a dense region.

Anomaly points: The number of anomaly points available outside a data cluster.

Neighborhood: The neighborhood of a point is a set of all points that are within a distance set by the neighborhood parameter.

Epsilon (ϵ or eps): The epsilon (ϵ) defines the distance defined of neighborhood around a certain point in the selected algorithm (Euclidean, Manhattan, Chebychev).

Anomaly extent: Distance between an anomaly point and its centre point (ϵ) of the cluster.

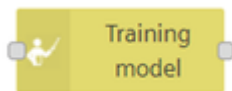
Anomalies are minimal for lesser anomaly extents. For greater distances, anomalies are larger.

Algorithms: Euclidean: The Euclidean distance is defined by a straight line between two points.

Manhattan: The Manhattan distance measure algorithm finds its application on high dimensional vectors. The sum of the absolute differences among their coordinates is defined as the distance between two points.

Chebychev: The maximum distance from the centre of a cell to its adjacent cells centre is defined as the Chebychev distance between two points.

Training model node



The training model node defines a new model of some data sets in Visual Flow Creator.

The data set information gets configured as read time series node with one condition - you can select only one aspect. The data set is defined by asset/ aspect/ variable. The data set can be selected using "Select asset aspect" selection dialog box.

Training model created will be stored in Industrial IoT exchange storage.

Training model node properties

You can configure the node by editing its node properties.

Edit Training model node

▼ node properties

📌 Name

📌 Model name

🗨 Description

1	epsilon 0.5
2	points 5

📄 Topic

📌 Topic Summary

Asset
a1ac2fa99bd94d95bef997e413eaab70 (CPU)
Aspect
CoreTemperature
Variable
Int

📌 Mode

🕒 Timezone

🕒 From

🕒 To

📌 Epsilon

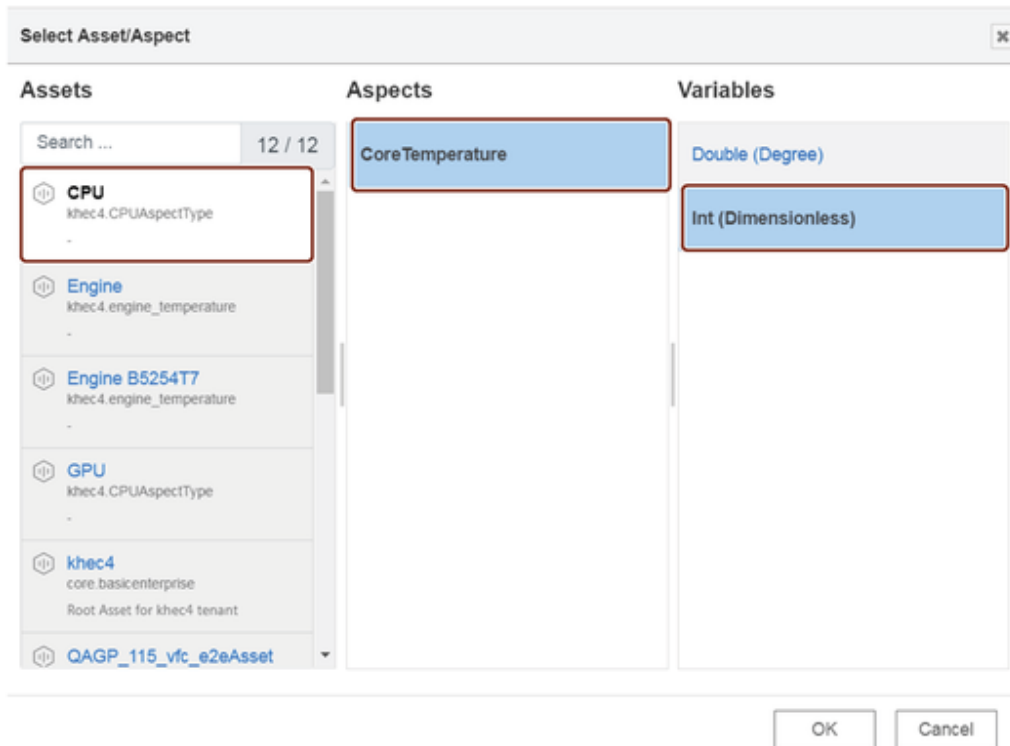
📌 Min. Points per Cluster

📌 Distance Measure Algorithm

Field Name	Description	Mandatory
Name	Select a display name for the node.	No
Model Name	Name the new training model.	Yes
Description	Enter the description.	No
Topic	Import the required time series data from the "Topic" field. The menu will	Yes
	redirect to select read time series data with	

Field Name	Description	Mandatory
	asset /aspect variable.	
Topic Summary	Each asset will have aspects which are defined by variables. Select the required	Yes
	variable of the selected aspect of its asset to import the data in the	
	"Training model" node properties. This will autofill the "Topic Summary"	
Mode	Select the mode from the drop-down menu. "Mode" is either defined by period of	Yes
	time or an interval.	
Period		
Period	Select the time period from the drop-down menu. The time period drop-down has	Yes
	values defined in minutes, hours, days and weeks.	
Offset	Select the offset value for balancing the effects. The offset values are	Yes
	defined in seconds.	
Interval		
Timezone	Select the required time zone from the drop-down menu.	Yes
From	Select the start date and time from the calendar and time menu.	Yes
To	Select the end date and time from the calendar and time menu.	Yes
Epsilon	Set the ϵ for the new model.	Yes
Min. points per cluster	Define the minimum number of points which are in each cluster.	Yes
Distance Measure Algorithm	Select the required distance measure algorithm from the drop-down.	Yes
	Three algorithms are available:	
	- Euclidean	
	- Manhattan	
	- Chebychev	

When you set the topic in "Select Asset/ Aspect" dialog box, select the necessary asset, aspect and its variable as shown below:



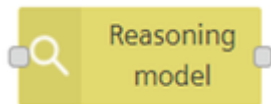
Jobs run involved

While injecting a timestamp to the "training model" node, a set of batch jobs run in the following series:

- Fetching IoT data
- Create model job
- Export job and generating output

Reasoning model node

The reasoning model node is used for analyzing data for a stored training model.



Reasoning model node properties

You can configure the node by editing its node properties.

Edit Reasoning model node

▼ node properties

🔍 Name

🔍 Model name ...

📄 Topic ...

🔍 Topic Summary

Asset
a1ac2fa99bd94d95bef997e413eaab70 (CPU)
Aspect
CoreTemperature
Variable
Double

🔍 Mode ▼

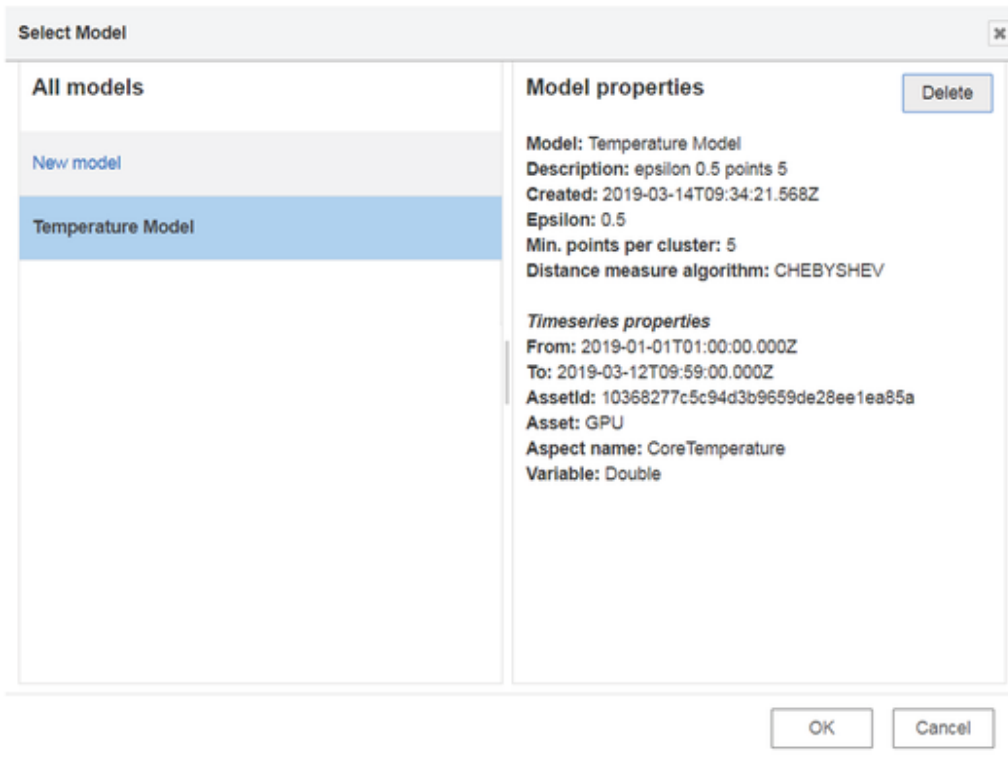
🕒 Period ▼

🕒 Offset ▼

Field Name	Description	Mandatory
Name	Select a display name for the node.	No
Model Name	Select a training model from the menu.	Yes
Description	Enter the description.	No
Topic	Import the required time series data from the "Topic" field. The menu will	Yes
	redirect to select read time series data with asset /aspect /variable.	
	The import will capture the actual details of the model selected. However,	
	the aspects and variables imported should be same as that selected for	
	the "Model Name".	
Topic Summary	Each asset will have aspects which are defined by variables. Select the required	Yes
	variable of the selected aspect of its asset to import the data in the	
	"Training model" node properties. This will autofill the "Topic Summary" field.	

Field Name	Description	Mandatory
Mode	Select the mode from the drop-down menu. "Mode" is either defined by period of	Yes
	time or an interval.	
Period		
Period	Select the time period from the drop-down menu. The time period drop-down has	Yes
	values defined in minutes, hours, days and weeks.	
Offset	Select the offset value for balancing the effects. The offset values are defined	Yes
	in seconds.	
Interval		
Timezone	Select the required time zone from the drop-down menu.	Yes
From	Select the start date and time from the calendar and time menu.	Yes
To	Select the end date and time from the calendar and time menu.	Yes

When you select a model name, the data of the selected model will be imported to the reasoning model node for analysis.



The property details of the selected model will also be displayed to the user.

Jobs run involved

While injecting a timestamp to the "reasoning model" node, a set of batch jobs run in the following series:

- IoT import job creation
- Import the job model
- Apply the job model
- Export the job and save in the database

Using simple anomaly nodes

Example scenario

Configure a training model node. Analyze the anomalies in the model.

Objective

To analyze the anomalies in the newly created training model with the help of a reasoning model node.

Requirements

Import the data from Insights Hub Monitor to create a training model.

Configuration of the training model is given below:

Field Name	Data
Name	CPU temp measure
Model Name	Temperature Model
Description	epsilon 0.5
	points 5
Topic	10368277c5c94d3b9659de28ee1ea85a
Topic Summary	Asset: 10368277c5c94d3b9659de28ee1ea85a (GPU)
	Aspect: CoreTemperature
	Variable: Double
Mode	Interval
Period	
Timezone	UTC

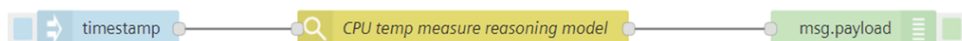
Field Name	Data
From	2019/01/01 01:00:00
To	2019/03/12 09:59:00
Epsilon	0.5
Min. points per cluster	5
Distance Measure Algorithm	Chebychev

Procedure

1. Select a reasoning model node from the "simple anomaly" nodes section.
2. Configure the reasoning model node. The configuration of the node is given below:

Field Name	Data
Name	CPU temp measure reasoning model
Model Name	Temperature Model
Topic	a1ac2fa99bd94d95bef997e413eaab70
Topic Summary	Asset: a1ac2fa99bd94d95bef997e413eaab70 (CPU)
	Aspect: CoreTemperature
	Variable: Double
Mode	Period
Period	
Period	1 hour
Offset	2 seconds

3. Insert an input timestamp and a message payload to the reasoning model node to generate the output in the message payload.





The "Topic Summary" properties in training model node should match the "Topic Summary" properties of the reasoning model node. This means that aspects and the variables for both should be same else the output will conflict and result in a failure.

Result

Inject the timestamp to process the flow.

You get the following result in the message payload:



Some of the initial anomaly points are highlighted in the graph as well in the array description.

Simple Anomaly node library

Introduction to simple anomaly nodes

The anomaly nodes help to detect anomalies in a data set.

The simple anomaly nodes in Visual Flow Creator are of two types:

- Training model
- Reasoning model

Terms and definitions

Cluster: A cluster is a high density region in a defined space. Each cluster is a set of data points which defines a dense region.

Anomaly points: The number of anomaly points available outside a data cluster.

Neighborhood: The neighborhood of a point is a set of all points that are within a distance set by the neighborhood parameter.

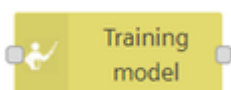
Epsilon (ϵ or eps): The epsilon (ϵ) defines the distance defined of neighborhood around a certain point in the selected algorithm (Euclidean, Manhattan, Chebychev).

Anomaly extent: Distance between an anomaly point and its centre point (ϵ) of the cluster. Anomalies are minimal for lesser anomaly extents. For greater distances, anomalies are larger.

Algorithms: Euclidean: The Euclidean distance is defined by a straight line between two points.

Manhattan: The Manhattan distance measure algorithm finds its application on high dimensional vectors. The sum of the absolute differences among their coordinates is defined as the distance between two points. **Chebychev:** The maximum distance from the centre of a cell to its adjacent cells centre is defined as the Chebychev distance between two points.

Training model node



The training model node defines a new model of some data sets in Visual Flow Creator.

The data set information gets configured as read time series node with one condition - you can select only one aspect. The data set is defined by asset/ aspect/ variable. The data set can be selected using "Select asset aspect" selection dialog box.

Training model created will be stored in Industrial IoT exchange storage.

Training model node properties

You can configure the node by editing its node properties.

Edit Training model node

Delete
Cancel
Done

▼ node properties

Name

Model name

Description

1	epsilon 0.5
2	points 5

Topic ...

Topic Summary

Asset
a1ac2fa99bd94d95bef997e413eaab70 (CPU)
Aspect
CoreTemperature
Variable
Int

Mode

Timezone

From

To

Epsilon

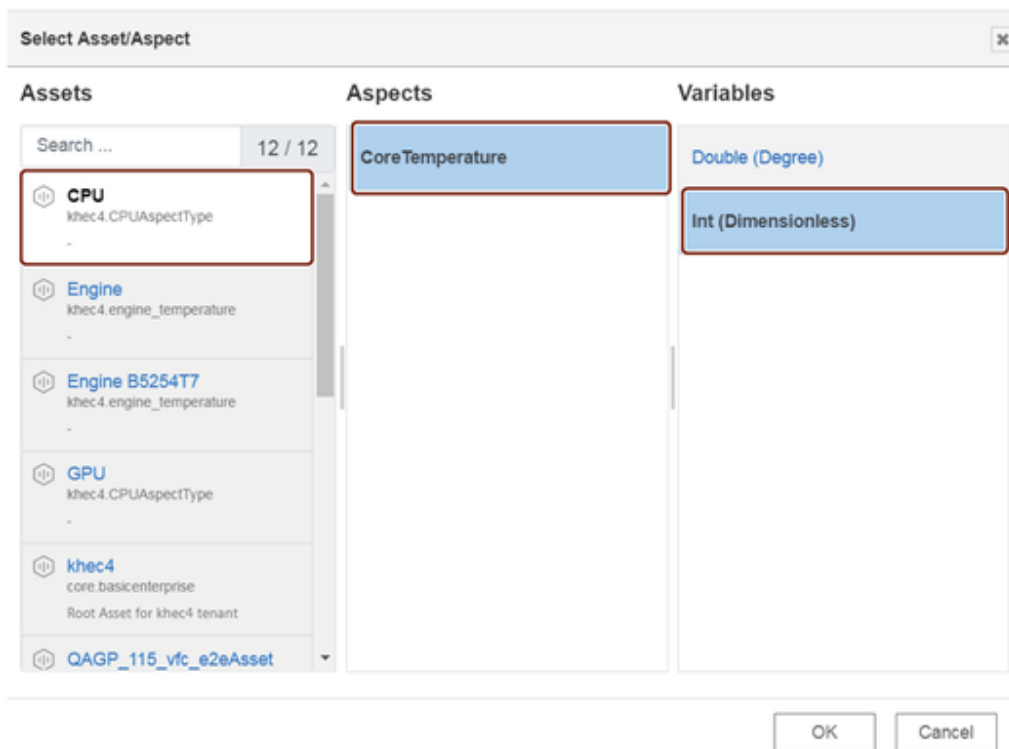
Min. Points per Cluster

Distance Measure Algorithm

Field Name	Description	Mandatory
Name	Select a display name for the node.	No
Model Name	Name the new training model.	Yes
Description	Enter the description.	No
Topic	Import the required time series data from the "Topic" field. The menu will	Yes
	redirect to select read time series data with	

Field Name	Description	Mandatory
	asset /aspect variable.	
Topic Summary	Each asset will have aspects which are defined by variables. Select the required	Yes
	variable of the selected aspect of its asset to import the data in the	
	"Training model" node properties. This will autofill the "Topic Summary"	
Mode	Select the mode from the drop-down menu. "Mode" is either defined by period of	Yes
	time or an interval.	
Period		
Period	Select the time period from the drop-down menu. The time period drop-down has	Yes
	values defined in minutes, hours, days and weeks.	
Offset	Select the offset value for balancing the effects. The offset values are	Yes
	defined in seconds.	
Interval		
Timezone	Select the required time zone from the drop-down menu.	Yes
From	Select the start date and time from the calendar and time menu.	Yes
To	Select the end date and time from the calendar and time menu.	Yes
Epsilon	Set the ϵ for the new model.	Yes
Min. points per cluster	Define the minimum number of points which are in each cluster.	Yes
Distance Measure Algorithm	Select the required distance measure algorithm from the drop-down.	Yes
	Three algorithms are available:	
	- Euclidean	
	- Manhattan	
	- Chebychev	

When you set the topic in "Select Asset/ Aspect" dialog box, select the necessary asset, aspect and its variable as shown below:



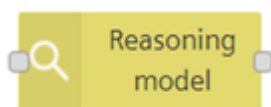
Jobs run involved

While injecting a timestamp to the "training model" node, a set of batch jobs run in the following series:

- Fetching IoT data
- Create model job
- Export job and generating output

Reasoning model node

The reasoning model node is used for analyzing data for a stored training model.



Reasoning model node properties

You can configure the node by editing its node properties.

Edit Reasoning model node

▼ **node properties**

🔍 **Name**

🔍 **Model name** ...

📄 **Topic** ...

🔍 **Topic Summary**

Asset

a1ac2fa99bd94d95bef997e413eaab70 (CPU)

Aspect

CoreTemperature

Variable

Double

🔍 **Mode** ▼

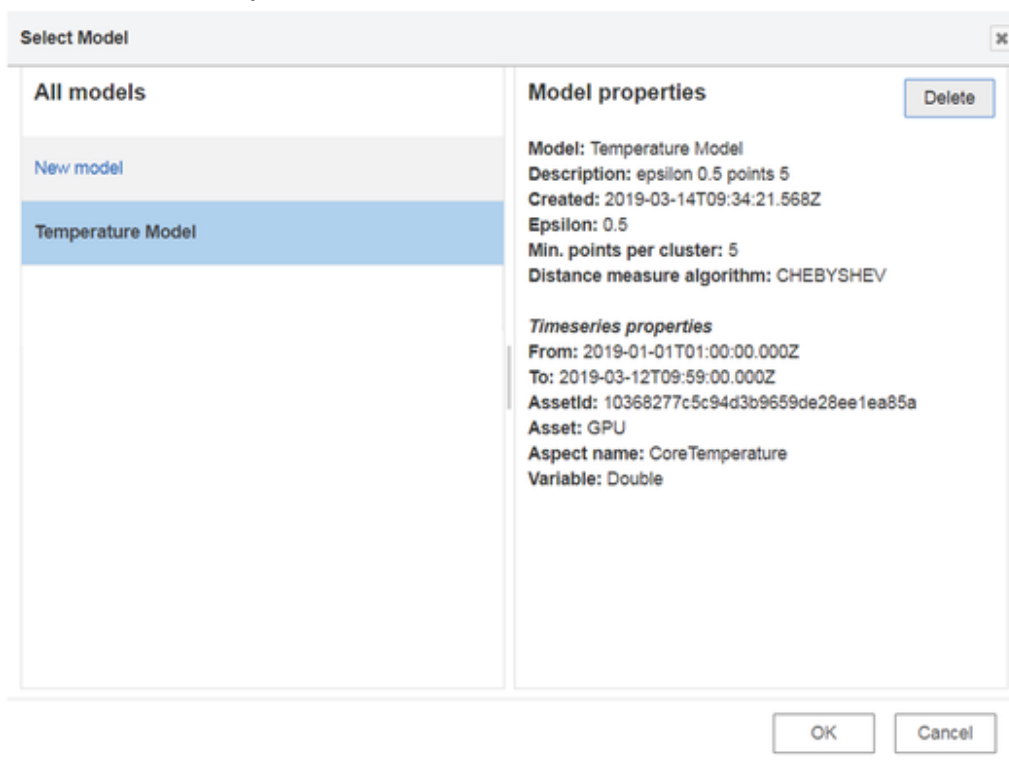
🕒 **Period** ▼

🕒 **Offset** ▼

Field Name	Description	Mandatory
Name	Select a display name for the node.	No
Model Name	Select a training model from the menu.	Yes
Description	Enter the description.	No
Topic	Import the required time series data from the "Topic" field. The menu will	Yes
	redirect to select read time series data with asset /aspect /variable.	
	The import will capture the actual details of the model selected. However,	
	the aspects and variables imported should be same as that selected for	
	the "Model Name".	
Topic Summary	Each asset will have aspects which are defined by variables. Select the required	Yes
	variable of the selected aspect of its asset to import the data in the	
	"Training model" node properties. This will autofill the "Topic Summary" field.	

Field Name	Description	Mandatory
Mode	Select the mode from the drop-down menu. "Mode" is either defined by period of	Yes
	time or an interval.	
Period		
Period	Select the time period from the drop-down menu. The time period drop-down has	Yes
	values defined in minutes, hours, days and weeks.	
Offset	Select the offset value for balancing the effects. The offset values are defined	Yes
	in seconds.	
Interval		
Timezone	Select the required time zone from the drop-down menu.	Yes
From	Select the start date and time from the calendar and time menu.	Yes
To	Select the end date and time from the calendar and time menu.	Yes

When you select a model name, the data of the selected model will be imported to the reasoning model node for analysis.



The property details of the selected model will also be displayed to the user.

Jobs run involved

While injecting a timestamp to the "reasoning model" node, a set of batch jobs run in the following series:

- IoT import job creation
- Import the job model
- Apply the job model
- Export the job and save in the database

Using simple anomaly nodes

Example scenario

Configure a training model node. Analyze the anomalies in the model.

Objective

To analyze the anomalies in the newly created training model with the help of a reasoning model node.

Requirements

Import the data from Insights Hub Monitor to create a training model.

Configuration of the training model is given below:

Field Name	Data
Name	CPU temp measure
Model Name	Temperature Model
Description	epsilon 0.5
	points 5
Topic	10368277c5c94d3b9659de28ee1ea85a
Topic Summary	Asset: 10368277c5c94d3b9659de28ee1ea85a (GPU)
	Aspect: CoreTemperature
	Variable: Double
Mode	Interval
Period	
Timezone	UTC

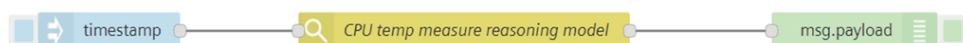
Field Name	Data
From	2019/01/01 01:00:00
To	2019/03/12 09:59:00
Epsilon	0.5
Min. points per cluster	5
Distance Measure Algorithm	Chebychev

Procedure

1. Select a reasoning model node from the "simple anomaly" nodes section.
2. Configure the reasoning model node. The configuration of the node is given below:

Field Name	Data
Name	CPU temp measure reasoning model
Model Name	Temperature Model
Topic	a1ac2fa99bd94d95bef997e413eaab70
Topic Summary	Asset: a1ac2fa99bd94d95bef997e413eaab70 (CPU)
	Aspect: CoreTemperature
	Variable: Double
Mode	Period
Period	
Period	1 hour
Offset	2 seconds

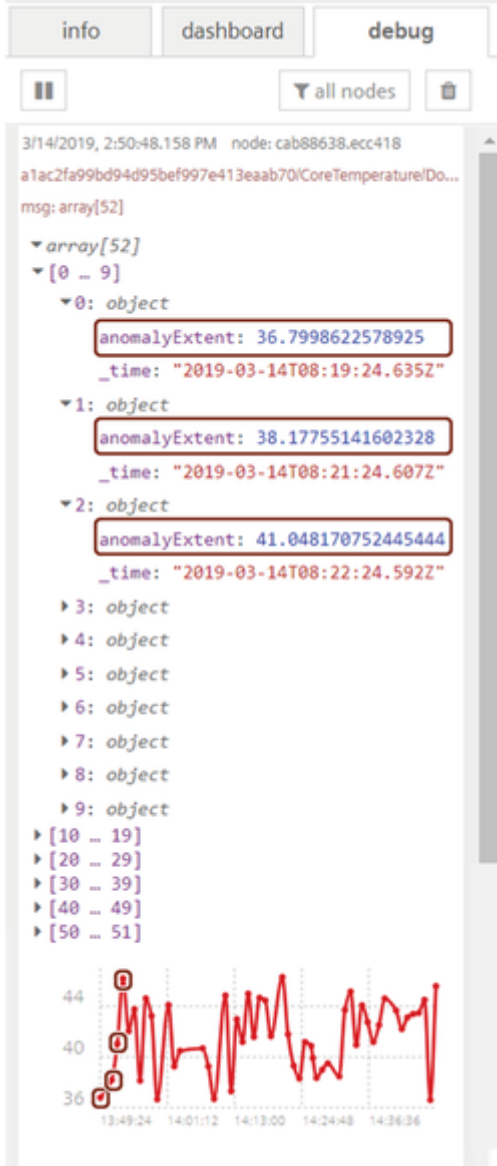
3. Insert an input timestamp and a message payload to the reasoning model node to generate the output in the message payload.



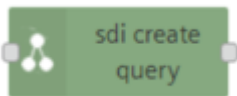
The "Topic Summary" properties in training model node should match the "Topic Summary" properties of the reasoning model node. This means that aspects and the variables for both should be same else the output will conflict and result in a failure.

Result

Inject the timestamp to process the flow.
You get the following result in the message payload:

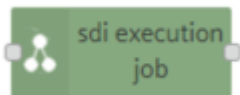


Some of the initial anomaly points are highlighted in the graph as well in the array description.



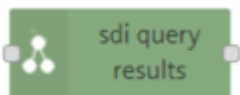
This node allows to create a query and retrieve the data. Query can be static or dynamic. The flow generates the output with a query ID and the data will be stored for further analysis.

SDI execution job



This node allows to execute a job for the dynamic queries. The query ID generated from the sdi create query node should be mentioned in the node properties of the sdi execution job node. The flow generates the output with execution job ID and the data will be stored for further analysis.

SDI query results

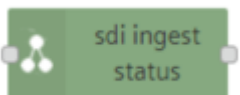


This node shows the result for a query and it retrieves both static and dynamic data. The query ID and execution job ID should be mentioned in the node properties. The flow generates the output and the data will be stored for further analysis.



If the flow is static, execution job ID is not required.

SDI ingest status



This node allows to get the status of the SDI ingest jobs. You can query the list of all ingest jobs or a single one.

Using SDI nodes

In order to use the SDI nodes with the integrated data lake (IDL), you have to do some preparation work. You cannot do that in VFC, but have to use the Insights Hub APIs. The steps are:

- Connect IDL to SDI (create a data lake record)
- Create a data registry
- Upload your data to IDL

Make sure that you upload the data in the "sdi" directory and add the meta-tag: "registryid_{yourregistryid}"

If these requirements are fulfilled, SDI will be able to use the data.

See the [SDI documentation](#) for further details.

You can use these SDI nodes to create queries, execution jobs and execute queries.

Example scenario

Retrieve the dynamic data from the create queries, execution jobs and execute queries.

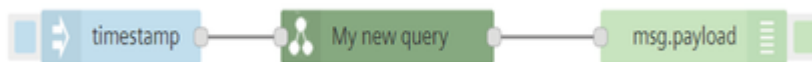
Objective

To retrieve the dynamic data from the create queries, execution jobs and execute queries and store it for further analysis.

SDI create query node procedure

To use SDI create query node, follow these steps:

1. Select the "sdi create query" node from the "data Lake and sdi" palette.
2. Connect the inject node with sdi create query node and debug node as shown below:



3. Double click the sdi create query node to edit the properties:
 - Name: My new query
 - Description: Creating a query
 - Dynamic: Yes
 - SQL Statement: `SELECT
airnow_aqi.parametername,airnow_aqi.reportingunits,airnow_aqi.value from airnow_aqi
where airnow_aqi.sitename=:\"airnow_aqi.sitename\"`
4. Save and deploy.

SDI create query output

The output is displayed in the debug window:

SDI Create Query node output

```
5/19/2020, 5:28:20.461 PM node: 9157dbde.48de98
msg: Object
{ id:
  "5ec3c9dc55e38921e78e1f66"
, createdAt:
  "2020-05-19T11:58:20.439Z"
, description:
  "Port au Choix query"
, executable: true, isBusinessQuery:
  false ... }
```

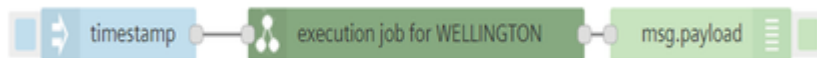
- ① Query ID

SDI execution job node procedure

The SDI execution job is used to create dynamic queries only. For more information, see [SDI execution job](#).

To use SDI execution job node, follow these steps:

1. Select the "sdi execution job" node from the "data Lake and sdi" palette.
2. Connect the inject node with sdi execution job node and debug node as shown below:



3. Double click the sdi execution job node to edit the properties:

- Name: execution job for WELLINGTON
- Description: creating execution job
- Query ID: 5ec3c9dc55e38921e78e1f66
- Parameters: Name="airnow_aqi.sitename" and Value="WELLINGTON"

4. Save and deploy.

SDI execution job output

The output is displayed in the message payload:

SDI execution job node output

```

5/19/2020, 8:03:54.415 PM node: 5d496708.fc7708
msg: Object
  {
    id: "5ec3ee5255e38921e78e1f79"
    , status: "IN_PROGRESS"
    , message: "Query execution is in progress..."
  }
  
```

- ① Execution job ID

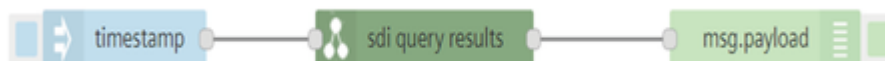


Execution job node is only for dynamic queries.

SDI query results node procedure

To use SDI query results node, follow these steps:

1. Select the "sdi query results" node from the "data Lake and sdi" palette.
2. Connect the inject node with sdi query results node and debug node as shown below:



3. Double click the sdi create query node to edit the properties:

- Name: sdi query results
- Query ID: 5ec3c9dc55e38921e78e1f66

- Execution job ID: 5ec3ee5255e38921e78e1f79

4. Save and deploy.

SDI query results

The output is displayed in the message payload:

SDI query result node output

```
5/19/2020, 8:02:37.301 PM node: cd03cadc.6bc9b8
msg: array[5]
  array[5]
    0: object
      airnow_aqi.parametername: "NO"
      airnow_aqi.reportingunits: "PPB"
      airnow_aqi.value: 0
    1: object
      airnow_aqi.parametername: "NO2"
      airnow_aqi.reportingunits: "PPB"
      airnow_aqi.value: 0
    2: object
      airnow_aqi.parametername: "OZONE"
      airnow_aqi.reportingunits: "PPB"
      airnow_aqi.value: 10
    3: object
      airnow_aqi.parametername: "PM2.5"
      airnow_aqi.reportingunits: "UG/M3"
      airnow_aqi.value: 1
    4: object
      airnow_aqi.parametername: "PM2.5"
      airnow_aqi.reportingunits: "ABS"
      airnow_aqi.value: 11
```

SDI ingest status node procedure

The SDI ingest status node is used to get the status of the SDI ingest jobs. For more information, see [SDI ingest status](#).

To use SDI query results node, follow these steps:

1. Select the "sdi ingest status" node from the "data Lake and sdi" palette.
2. Connect the inject node with sdi ingest status node and debug node as shown below:



3. Save and deploy.

SDI ingest status output

The output is displayed in the message payload:

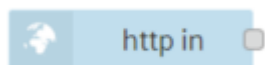
SDI ingest status node output

```
6/16/2020, 4:34:27.439 PM node: e78b9ae.107e168
msg: array[14]
  array[14]
    [0 - 9]
      0: object
      1: object
      2: object
      3: object
      4: object
      5: object
      6: object
      7: object
      8: object
      9: object
    [10 - 13]
      10: object
      11: object
      12: object
      13: object
```

6.14 http-in node

Usage of Http-in node

http-in nodes



You can create an http endpoint for the web services.

You use this node to:

- To create an http endpoint for the web services with specified access.

Requirements:

- Endpoint should be specified.



- Endpoint should start with '/'.
- You cannot send any response without the flow to complete the request.
- You cannot create the flow in other user flow tab to generate the output.

Http URL access:

Access	Description
Only the user of this flow	The flow created user can have the permission to access the flow.

Access	Description
All users of this environment	The users with environment access can have the permission to access the flow.
Public access using keys	All the public users will have the permission to access the flow by accessing the generated link with key as per the specified expiry date.

Key Management:

You can manage the access key by the "Key Management" in http edit node properties.

Using Http-in node

Example scenario

Create an endpoint for timeseries data to display it in the web services.

Objective

To create an endpoint and display the timeseries data in the web services for further analysis.

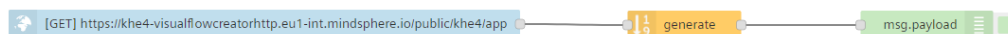
Requirements

- http-in node
- generate node
- http response node

Procedure

Accessing http node:

1. Design the http flow as shown below:



2. Double click http-in node to edit the properties:

Edit http in node

Delete Cancel Done

node properties

Name

Method

Endpoint

URL

Access

Available keys

Generate new Key

Valid till

[Open key management page](#)

Power Mode (Running time up to 120 sec.)

> port labels

- Select a method
- Specify the endpoint

- Select the type of access
- Generate the key with expiry date (optional)

3. Click "Done".

4. Click "Save".

Accessing key management:




To manage the key from http edit node:

1. Click 

Close


Key Management

Endpoint Valid till Generate key

Endpoint	Key	Created	Valid until	
/app 	04f1c40dfcb192f81b80859cb378d3c3d2d32ee707e6409d6f5f3ccd9514... 	6/15/2022	6/22/2022	

2. To delete the existing key, click .

3. To generate new key, click Generate key.



You can create multiple keys for an endpoint.

Result

You can click the URL link to display the output data in the web services.

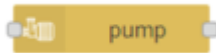
```
[{"key": "2020-03-31780-16-53.0000", "parameter": "0"}, {"key": "2020-03-31780-16-59.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-17-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-17-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-17-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-17-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-17-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-18-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-18-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-18-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-18-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-18-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-19-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-19-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-19-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-19-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-19-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-20-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-20-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-20-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-20-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-20-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-21-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-21-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-21-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-21-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-21-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-22-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-22-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-22-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-22-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-22-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-23-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-23-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-23-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-23-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-23-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-24-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-24-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-24-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-24-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-24-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-25-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-25-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-25-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-25-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-25-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-26-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-26-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-26-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-26-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-26-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-27-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-27-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-27-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-27-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-27-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-28-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-28-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-28-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-28-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-28-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-29-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-29-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-29-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-29-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-29-29.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-30-05.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-30-11.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-30-17.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-30-23.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-30-29.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-31-05.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-31-11.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-31-17.0000", "parameter": "0.9511"}, {"key": "2020-03-31780-31-23.0000", "parameter": "0.5881"}, {"key": "2020-03-31780-31-29.0000", "parameter": "0.9511"}]
```


6.15 Pump-Simulation node

Usage of Pump-Simulation node

The following node shows the Pump-Simulation node and its respective functions:

Pump-simulation



This node allows to generate the time series data of a simulated water pump. The two modes in water pump simulation provides the following function:

- Whole scenario simulation
- Real time simulation

Whole scenario simulation

In this simulation mode, each trigger generates time series for the desired past time. It generates one time series object for each second. You can choose the scenario length in the configuration or specify the desired scenario using the property message.scenario.

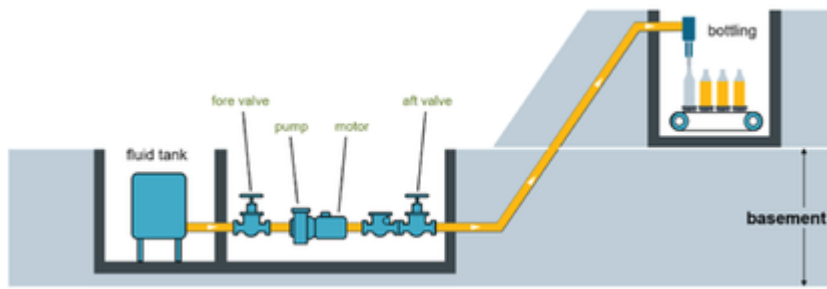
Real time simulation

In this simulation mode, each trigger generates time series that fill the gap in the last execution, but not longer than 30 minutes (1800 time series). You can specify the next scenario by sending message with the property scenario. The simulation accepts the next scenario but does not send any message.

Using Pump-Simulation node

Example scenario

In F&B industry, pumping liquids is one of the most frequent operations. Disturbance in pumps can cause interruptions in production process and downstream supply chain. If many pumps are in operation, then localizing the error and correction causes unplanned maintenance and costs. Usage of Industrial Internet of Things during pump operation enables to minimize down times and keep the production running constantly.



Simulating water pump node generates the time series data with three different scenarios like standard, plugging before pump and plugging after pump.

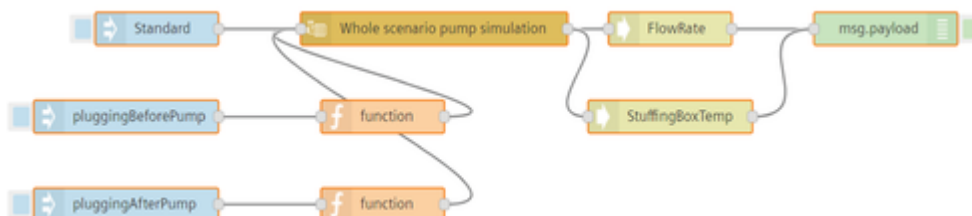
Objective

To monitor the pump installed on the bottling line in a brewery and connected to Industrial IoT. A pump consists of two valves (one before and one after), the pump and a fluid-flow meter as shown in the figure above. The pump is connected to a Siemens SIMATIC IoT2040 and collects the following data points: pressure before the pump (between the left valve and the pump), pressure after the pump (between the pump and the right valve), motor current of the pump, temperature of the motor (stuffing box) and percolation measured by the fluid-flow meter.

Procedure for Whole scenario simulation

To generate the time series data with three different scenarios with simulated water pump, follow these steps:

1. Create the flow as shown below:



2. Edit standard scenario inject node:

- Name: Standard
- Payload: timestamp

3. Edit scenario inject node:

- Name: pluggingBeforePump
- Payload: pluggingBeforePump (string)

4. Edit pluggingAfterPump scenario inject node:

- Name: pluggingAfterPump
- Payload : pluggingAfterPump (string)

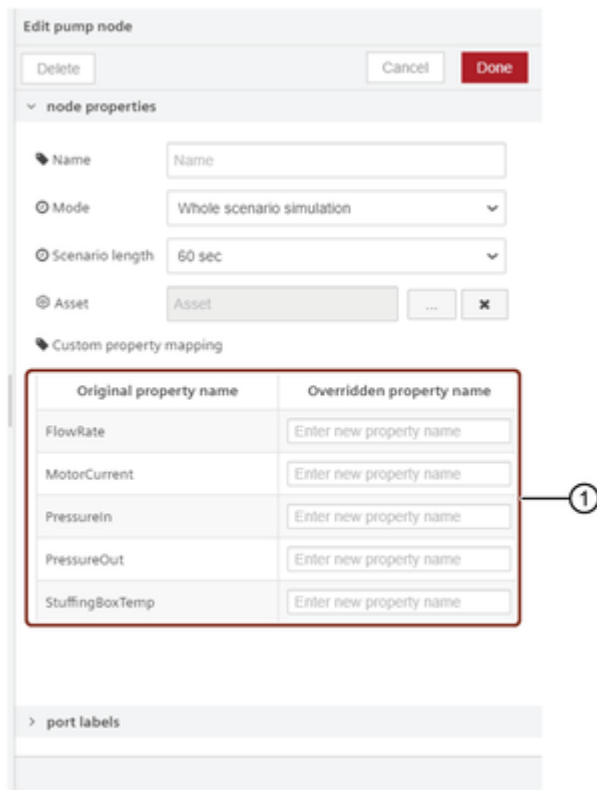
5. Edit pluggingBeforePump scenario function node:

- Code: `msg.scenario = 'pluggingBeforePump'; return msg;`

6. Edit pluggingAfterPump scenario function node:


- Code: `msg.scenario = 'pluggingAfterPump'; return msg;`

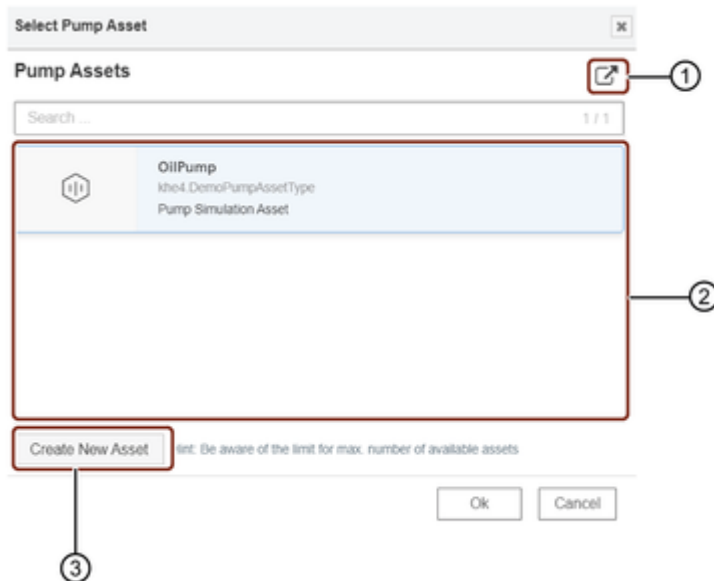
7. Edit pump node:



① Overrides the existing property name or specify the custom property name

- Mode: Whole scenario simulation

- Asset: Click  (./images/select-icon.png) to select an existing pump asset or create a new pump asset and click "Ok".



① Opens pump asset in Asset Manager

② List of the pump assets

③ Creates a new pump asset

Using "Create New Asset" button, you can create an asset for the pump in Industrial IoT.

8. Edit map node:

- Name: Motor_Current, Passage
- Provided code: { Motor_Current: element['Motor_Current'], _time: element['_time'], Passage: element['Passage']}

9. Edit map node:

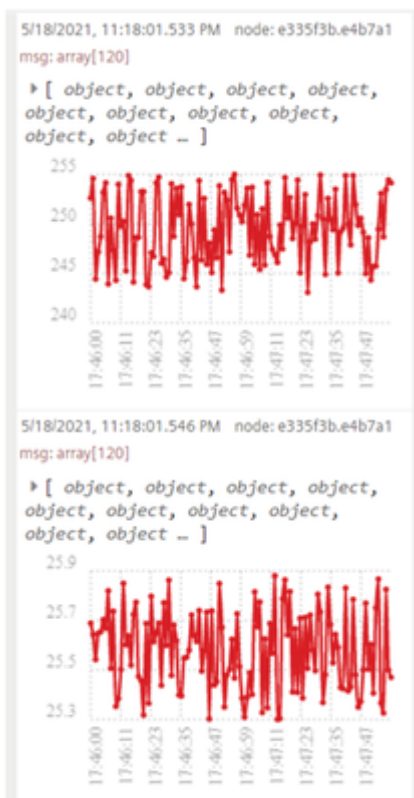
- Name: Motor_Current
- Provided code: { Motor_Current: element['Motor_Current'], _time: element['_time']}

10. Save the flow.

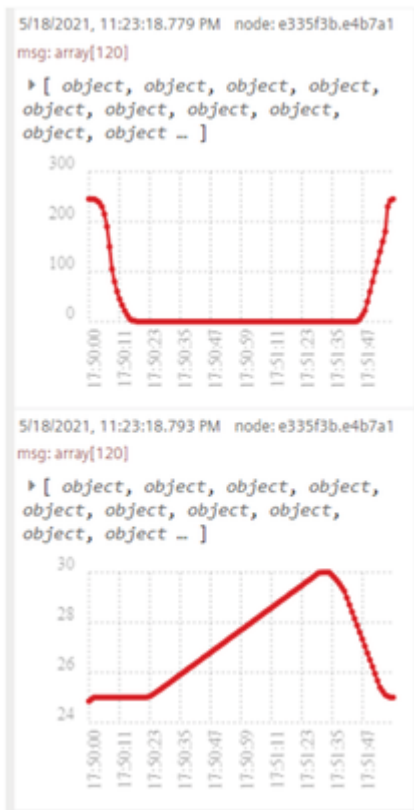
Result for Whole scenario simulation

The following graphic shows the graphs in the message payload:

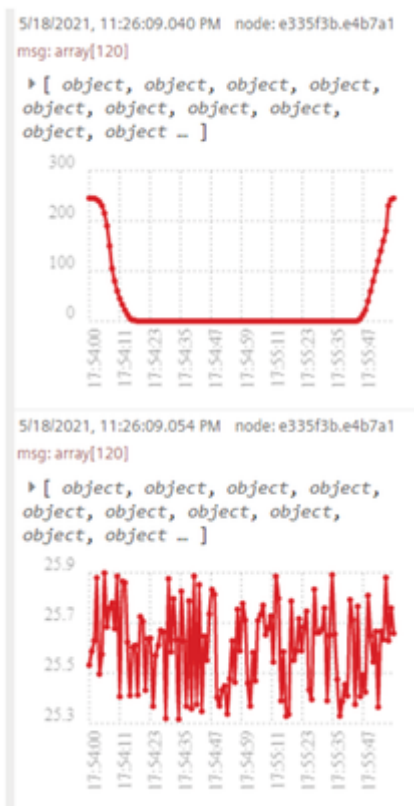
For standard scenario



For pluggingBeforePump scenario



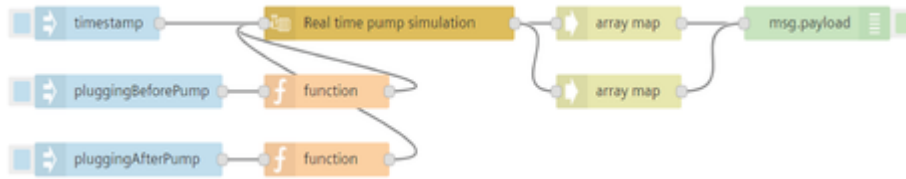
For pluggingAfterPump scenario



Procedure for Real time simulation

To generate the time series data with three different scenarios with simulated water pump, follow these steps:

1. Create the flow as shown below:



2. Edit standard scenario inject node:

- Name: Standard
- Payload: timestamp

3. Edit pluggingBeforePump scenario inject node:

- Name: pluggingBeforePump
- Payload: pluggingBeforePump (string)

4. Edit pluggingAfterPump scenario inject node:

- Name: pluggingAfterPump
- Payload : pluggingAfterPump (string)

5. Edit pluggingBeforePump scenario function node:

- Code: `msg.scenario = 'pluggingBeforePump'; return msg;`

6. Edit pluggingAfterPump scenario function node:

- Code: `msg.scenario = 'pluggingAfterPump'; return msg;`

7. Edit pump node:

- Mode: Real time simulation
- Asset: Select an existing pump asset or create a new pump asset.

8. Save the flow.

Result for Real time simulation

The following graphic shows the graphs in the message payload:

For Standard scenario



6.16 Usage of MQTT nodes

The most important terms in the MQTT telemetry protocol are explained below.



- Use MQTT nodes for productive use cases only or you can connect with connectivity features. For more information on Connectivity, refer to [Connectivity](#).
- These MQTT nodes are used with some limitations. For more information on limitations, refer to [Technical specifications](#).
- MQTT is now enabled for subtenants also.
- MQTT nodes are not available for Region Europe 2 and for private deployment options.

MQTT message

A message with MQTT consists of several parts:

- A defined subject ("Topic")
- An assigned criterion for "Quality of Service"
- The message text


MQTT Client

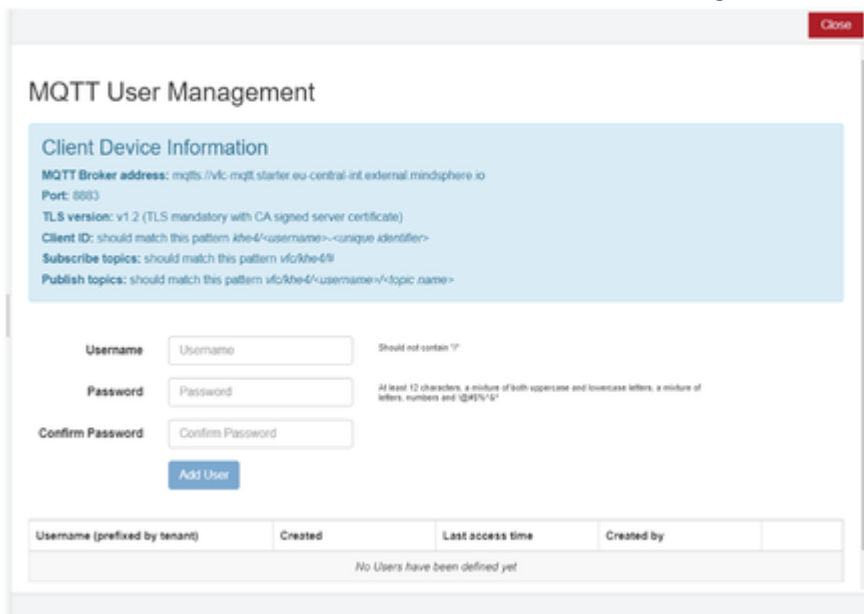
An MQTT Client is a program or device that uses MQTT. A client always actively establishes the connection to the broker. A client can perform the following functions:

- Send messages with a defined subject ("Topic"), in which other clients might be interested, to the MQTT Broker (Publish mechanism)
- Subscribe messages which follow a certain topic (Subscriber mechanism) at the MQTT Broker
- Unsubscribe yourself from subscribed messages
- Disconnect from the broker

On-boarding process

To onboard MQTT client, follow these steps:

1. In Visual Flow Creator, click  and select "MQTT Management"



MQTT User Management

Client Device Information

MQTT Broker address: mqtt://vc.mqtstarter.eu:central-int-external.mindsphere.io
Port: 8883
TLS version: v1.2 (TLS mandatory with CA signed server certificate)
Client ID: should match this pattern `!he-4<username>~<unique identifier>`
Subscribe topics: should match this pattern `vc/!he-4/!`
Publish topics: should match this pattern `vc/!he-4/<username>/<topic name>`

Username Should not contain '@'

Password At least 12 characters, a mixture of both uppercase and lowercase letters, a mixture of letters, numbers and `!@#%&'&*`

Confirm Password

Username (prefixed by tenant)	Created	Last access time	Created by
No Users have been defined yet			

2. Create the credentials for the client.
3. Enter the credentials from the client side.
 - Credential limitations: Username begin with `<tenant/username>` in MQTT User Management.
 - Client ID starts with `<tenant/username-unique identifier>` that could be selected by user.



- Adhere with all the client's unique identifiers to connect with the client.
- Only Visual Flow Creator admin can use "MQTT User Management" page.

MQTT Broker

An MQTT Broker is the central component of MQTT and can be a program or a device. The MQTT Broker acts as an intermediary between the sending MQTT Client and the subscribing MQTT Client. The MQTT Broker manages the topics including the messages contained therein and regulates the access to the topics. The MQTT Broker has the following functions:

- Accept network connections from the MQTT Clients
- Receive messages from an MQTT Client
- Edit subscription requests from MQTT Clients
- Forward messages to the MQTT Clients that match your subscription



Visual Flow Creator provides you the own MQTT broker with address mqttp://vfc-mqtt.starter.eu-central-int.external.mindsphere.io.

Topics

MQTT messages are organized in topics. A topic "describes" a subject area. The topics can be subscribed to by the MQTT Clients (subscriber mechanism). The sender of a message (Publisher mechanism) is responsible for defining content and topic when sending the message. The broker then takes care that the Subscribers get the news from the subscribed topics. The topics follow a defined scheme. They are similar to a directory path and represent a hierarchy.



The client can only subscribe on the topics and publish the topics that starts from vfc//.

MQTT ports

MQTT defines an OASIS or ISO standard (ISO/IEC PRF 20922). Depending on the protocols used, MQTT runs on different access ports. Visual Flow Creator offers the following ports:

8883: MQTT, encrypted

443: MQTT via WebSockets, encrypted

Architecture

The MQTT is a publish and subscribe protocol. This mechanism decouples a client sending messages (Publishers) from one or more clients receiving the messages (Subscribers). This also means that the "Publishers" know nothing about the existence of the "Subscribers" (and vice versa). There is a third component in the MQTT architecture, the MQTT Broker. The MQTT Broker is located between "Publisher" and "Subscriber". The MQTT Broker controls the communication.

Quality of Service (QoS)

The MQTT specification provides three service qualities for message transmission quality assurance:

- QoS "0": The lowest level 0 is a "fire'n'forget" method. This means that there is no guarantee that the message will arrive at all.
- QoS "1": The QoS level 1 ensures that the message ends up in the topic queue at least once. The MQTT Broker acknowledges receipt of the message.
- QoS "2": In the highest level 2, the MQTT Broker guarantees by multiple handshake with the MQTT Client that the message is exactly filed once.

Last will

MQTT supports the "Last Will and Testament" feature. This feature is used to notify other MQTT Clients if the connection to a MQTT Client has been disconnected accidentally.

Each MQTT Client can specify its last will while connecting to the MQTT Broker and notify the MQTT Broker. This last will is built like a normal MQTT message, including topic, QoS and payload. The MQTT Broker saves the last will. As soon as the MQTT Broker notices that the connection with the MQTT Client in question has been abruptly terminated, the MQTT Broker sends the last will as an MQTT message to all subscribers who have registered for the topic. In this way, the subscribers also learn that the MQTT Client has been disconnected.

Keep-Alive

MQTT supports the "Keep-Alive" feature. This ensures that the connection is still open and the MQTT Client and MQTT Broker are connected.

For the Keep-Alive, the MQTT Clients define a time interval and communicate it to the MQTT Broker during their connection setup. This interval is the largest possible tolerated time period in which the MQTT Client and the MQTT Broker may remain without contact. If the time is exceeded, the MQTT Broker must disconnect.

That means that, as long as the MQTT Client periodically sends messages to the broker within the Keep-Alive interval, the MQTT Client does not need to take any special action to maintain the connection. However, if the MQTT Client does not send any messages within the Keep-Alive interval, they must ping the MQTT Broker before the deadline expires. With this ping, the MQTT Client signals to the MQTT Broker that it is still available.

When a message or a ping packet has been sent to the MQTT Broker, timing for the Keep-Alive interval begins again.

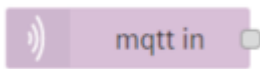
Message persistence

If the connection to an MQTT Client is interrupted, the broker can cache new messages for this client for later delivery.

Retained messages

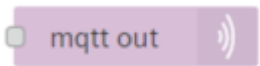
The first time an MQTT Client subscribes to a topic, it usually gets a message only when another MQTT Client sends a message with the subscribed topic the next time. With "Retained messages", the subscriber receives the last value sent to the topic prior to its subscription request, delivered immediately.

MQTT IN



The MQTT IN node allows you to subscribe to the specified topic.

MQTT OUT



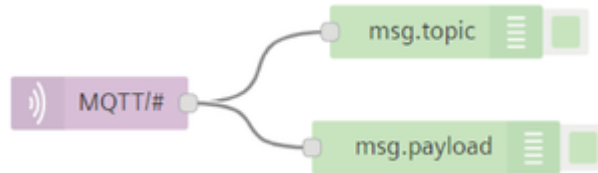
The MQTT OUT node allows you to publish the messages to MQTT broker. The following properties are available for MQTT5 and Visual Flow Creator supports MQTT5.

Node property	Description
User Properties	It allows to appear multiple times to represent multiple names and value pairs in the "Object" format
Response topic	It is used as the topic name for a response message in the "String" format
Correlation Data	It is used to identify the Response Message as per the "Request Message" received in the "String" format
Content Type	It describes the content of the application message in the following formats: <ul style="list-style-type: none"> - application/json - application/octet-stream - text/csv - text/html - text/plain - other
Expiry (secs)	It is the process time of the application message in seconds

Example for MQTT IN

The following example shows to subscribe to the messages from the specified topic using the MQTT IN node:

1. Create the flow as shown below:



2. Edit mqtt in node properties and enter the details:

The screenshot shows the 'Edit mqtt in node' dialog box. It has a 'Delete' button on the left, and 'Cancel' and 'Done' buttons on the right. Under 'node properties', there are several fields: 'Topic' with a dropdown set to 'vfo/khe4/' and a text input containing 'MQTT#'; 'QoS' with a dropdown set to '0'; 'Output' with a dropdown set to 'auto-detect (string or buffer)'; 'Flags' with two checkboxes: 'Do not receive messages published by this client' (unchecked) and 'Keep retain flag of original publish' (checked); 'Retained message handling' with a dropdown set to 'Send retained messages'; and 'Name' with a text input containing 'Name'. At the bottom, there is a section for 'port labels' which is currently collapsed.



The subscription topic can include MQTT wildcards, "+" for one level, "#" for multiple levels.

3. Save and execute the flow.

Result

The specified topic is successfully subscribed.

Example for MQTT OUT

The following example shows to publish the messages using the MQTT OUT node:

1. Create the flow as shown below:



2. Edit inject node properties:

- Payload (string): VFC MQTT

3. Edit mqtt out node properties and enter the details:

Delete Cancel Done

node properties

Topic vfc/khe4/ MQTT

QoS 0 Retain false

User Properties

Response topic

Correlation Data

Content Type

Expiry (secs)

Name Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

port labels

4. Save and execute the flow.

Result

The output is displayed in the message payload:

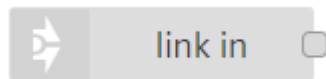
```
6/15/2021, 3:10:08.109 PM node: bb02f11d.acca7
vfc/khe4/MQTT: msg: string[13]
"vfc/khe4/MQTT"

6/15/2021, 3:10:08.109 PM node: f852092e.1bd728
vfc/khe4/MQTT: msg: string[8]
"VFC MQTT"
```

6.17 Link Nodes

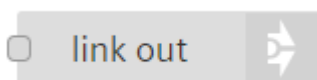
Link nodes allow you to create a virtual connection between the flows which exist in any flow tab.

Link in



The Link in node allows you to connect with other link out nodes which are connected with other flows.

Link out



Based on the mode selected in the Link out node properties, you can configure it to send messages to all connected Link in nodes or to send a response back to the link call node that triggers the flow. There are two types of modes for Link out node:

- Send to all connected link nodes
- Return to calling link node

Link call



The Link call node allows you to call a flow that starts with a Link in node and sends a response.

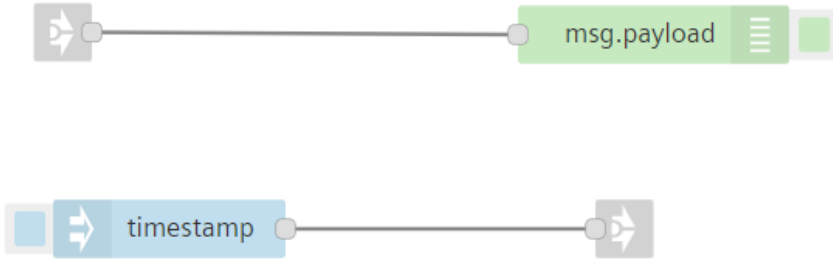
Example 1

The below given flows are an example for Link in, Link out, and link call nodes are configured with each other to send messages to all connected link nodes and return the message through link call node based on the mode selected.

Send to all connected link nodes:

To execute the example flow using link nodes for sending messages to all connected link nodes, follow these steps:

1. Create the flows as shown below:



2. Edit link out node properties and configure with link in node to send message:

Edit link out node

Delete Cancel Done

node properties

Name

Mode

Flow 2 *

- 3b30935d.e23d4c

> port labels

3. Edit link call node properties and configure with link in node to send message:

Edit link call node

Delete Cancel Done

node properties

Name

Timeout seconds

Link Type

Flow 2 *

- 3b30935d.e23d4c

> port labels

4. Save and execute the link out and link call node flows.

Result

You can view results in the message payload for sending messages to all connected link nodes.

Result for Link in and link out configuration flow:

debug

i
📊
🔍
▼

⏸

🔍 all nodes
🗑

1/30/2023, 11:41:41.105 AM node:
a9a6b68.43a7048
msg: number
1675059101104

Result for Link call and link in configuration flow:

debug

i
📊
🔍
▼

⏸

🔍 all nodes
🗑

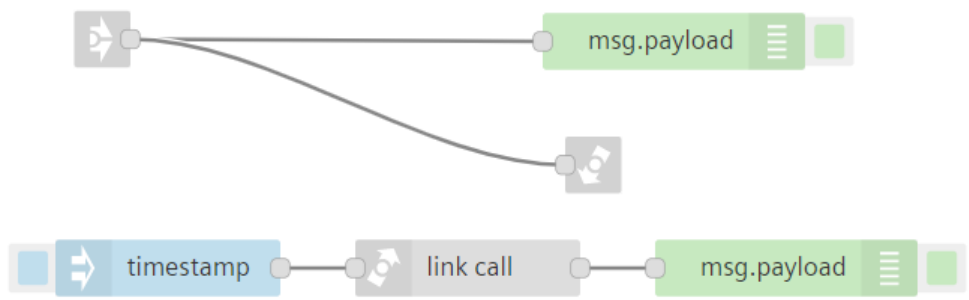
1/30/2023, 11:59:01.584 AM node:
a9a6b68.43a7048
msg: number
1675060141583

Example 2

Return to calling link node:

To execute the example flow to return message from the calling link node, follow these steps:

1. Create the flows as shown below:



2. Edit link out node properties and configure the mode to "Return to calling link node":

Edit link out node

Delete Cancel Done

node properties

Name Name

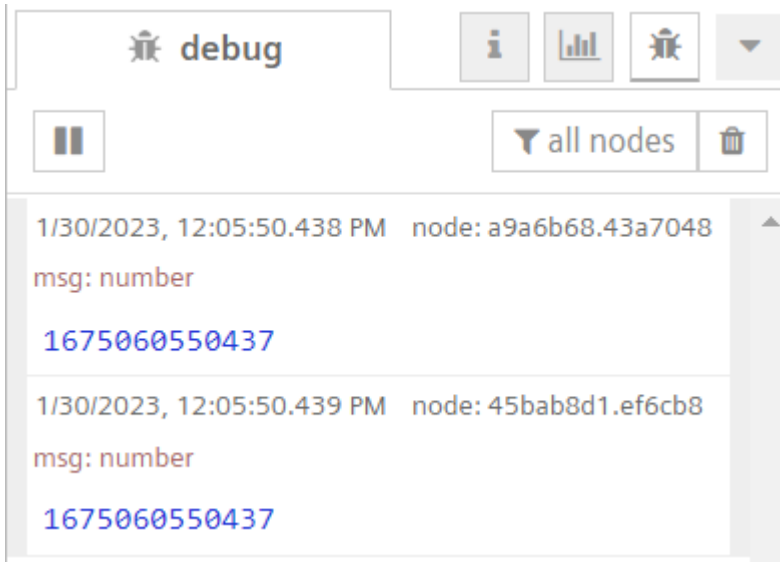
Mode Return to calling link node

port labels

3. Save and execute the link call node flow.

Result

You can view both results in the message payload when you return the message to the calling node.



6.18 Usage of Business Intelligence nodes

Business Intelligence is a powerful add-on application, that allows creating custom visualizations of your data. You can integrate data from different assets, files or queries in a single dashboard and apply custom calculations, filters or aggregations.

Business Intelligence nodes allows you to list, delete and create or update the datasources in Business Intelligence. For more information, refer to [Business Intelligence Getting Started documentation](#).

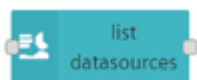


Business Intelligence nodes allows you to list, delete and create or update the datasources in Business Intelligence. For more information, refer to [Insights Hub Business Intelligence documentation](#).



To enable these Business Intelligence nodes in Visual Flow Creator, you need to log in Business Intelligence application at least once.

List datasources

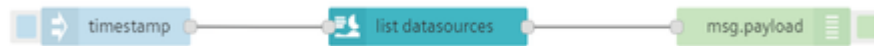


The "list datasources" node allows you to list all the datasources from Business Intelligence and display the data in the message payload.

Example

To execute the example flow using list datasources node, follow these steps:

1. Create the flow as shown below:



2. Save and execute the flow.

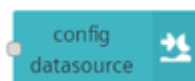
Output in the message payload

The list of the datasources are displayed in the message payload successfully.

```

2/7/2022, 3:22:33.068 PM node: 39604c28.6276f4
msg: array[2]
  array[2]
    0: object
      resources: object
      acquisition: object
      projectId: "610a2341-8940-4927-93f0-5297c808afe1"
      tags: array[0]
      name: "MyNewDatasource2"
      id: "e2252d29-4e49-4fb7-bfbd-9392c7702d99"
      etag: "d60d41a8762516490cf9a032d134375887fc28e432325a772d000628a741a567"
      created: "2022-02-04T08:38:26.426Z"
      lastModified: "2022-02-04T08:38:26.426Z"
      nextAcquisition: "2022-02-07T09:55Z"
      processingStatus: "pending"
      lastResult: object
      runLogs: array[0]
    1: object
  
```

Config datasource



The "config datasource" node allows you to create or update a datasource in Business Intelligence.

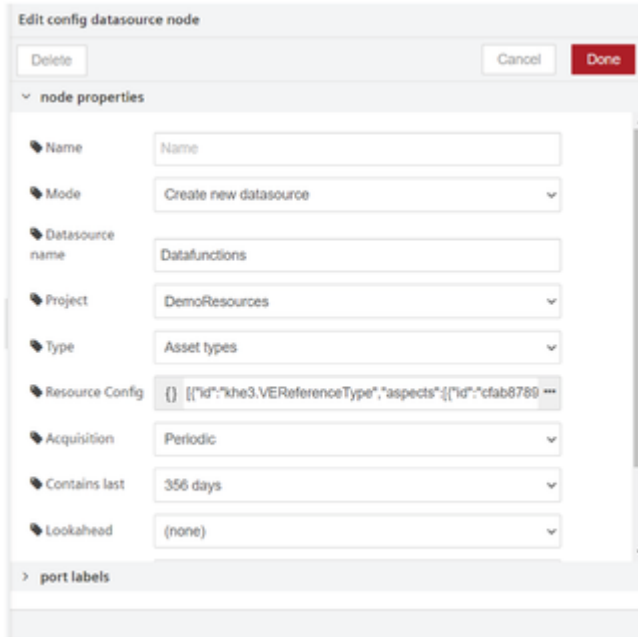
Example

To execute the example flow using config datasource node, follow these steps:

1. Create the flow as shown below:



2. Edit config datasource node to create or update a datasource.

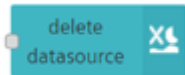


3. Save and execute the flow.

Output

The datasource is created or updated successfully.

Delete datasource



The "delete datasource" node allows you to delete a datasource from Business Intelligence.

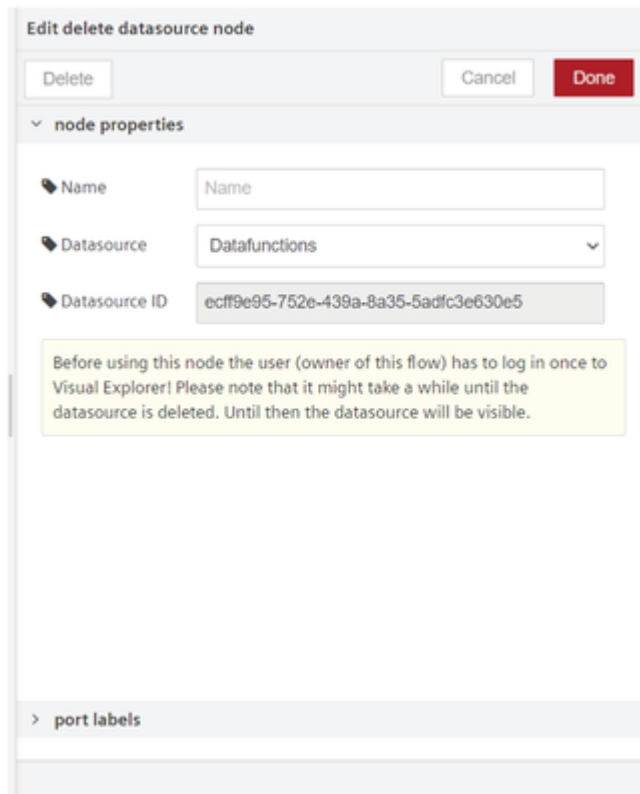
Example

To execute the example flow using delete datasource node, follow these steps:

1. Create the flow as shown below:



2. Edit delete datasource node to delete a datasource.



3. Save and execute the flow.

Output

The datasource is deleted successfully.

6.19 Simatic Notifier node

Usage of Simatic Notifier node

Simatic Notifier nodes in Visual Flow Creator are integrated with Simatic Notifier application. With these nodes you can raise, accept and clear the events in Simatic Notifier application.

Simatic Notifier nodes will notify the users with following event types based on the event priority:

- Alert
- Warning
- Information

For more information about the Simatic Notifier, refer to [SIMATIC Notifier](#) documentation.



To use these nodes, Simatic Notifier app should be subscribed and available on the Launchpad.

Raise event



The "Raise event" node allows you to raise the event in Simatic Notifier application.

Example

To execute the example flow using raise event node, follow these steps:

1. Create the flow as shown below:



2. Edit inject node properties:

- Payload (string): Pump overheating

3. Edit raise event node properties to select the asset and event type:

Edit raise event node

Delete Cancel Done

node properties

Name


Asset ...

Type ▾

Description

1

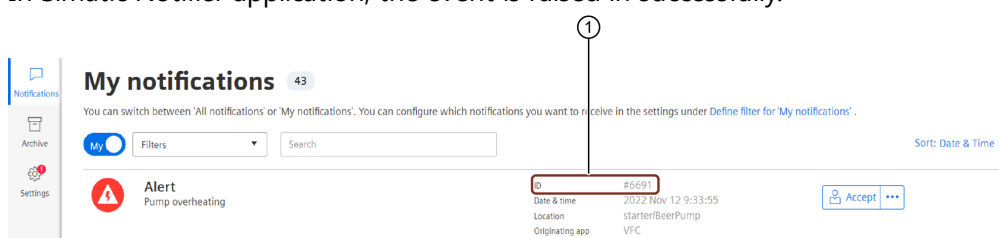
> port labels

4. Click  to select the asset and event type.

5. Save and execute the flow.

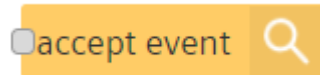
Output

In Simatic Notifier application, the event is raised in successfully.



① Notification ID

Accept event



After the event is raised, "Accept event" node allows to accept the raised event using "Notification ID".

Example

To execute the example flow using accept event node, follow these steps:

1. Create the flow as shown below:



2. Edit inject node properties:

- Payload (string): Pump overheating

3. Edit accept event node properties to enter the "Notification ID":

Edit accept event node

Delete Cancel Done

node properties

Name

Notification ID

> port labels

4. Save and execute the flow.

Output

In Simatic Notifier application, the event is accepted successfully.

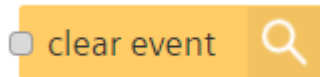
My notifications 43

You can switch between 'All notifications' or 'My notifications'. You can configure which notifications you want to receive in the settings under [Define filter for 'My notifications'](#).

My Filters Search Sort: Date & Time

	Alert Pump overheating	ID Date & time Location Originating app	#6691 2022 Nov 12 9:33:55 starter/BeerPump VFC	Accepted by me Accepted	
--	----------------------------------	--------------------------------------------------	---------------------------------------------------------	----------------------------	--

Clear event



After the event is resolved, "Clear event" node allows you to clear and archive the accepted event using "Notification ID".

Example

To execute the example flow using clear event node, follow these steps:

1. Create the flow as shown below:



2. Edit inject node properties:

- Payload (string): Pump overheating

3. Edit clear event node properties to enter the "Notification ID":

Edit clear event node

Delete Cancel Done

node properties

Name

Notification ID

> port labels

4. Save and execute the flow.

Output

In Simatic Notifier application, the event is cleared and archived successfully. To view the cleared and archived event, navigate to "Archive" tab.

My archived notifications

You can switch views between 'My archived notifications' or 'All archived notifications'. Under 'Manage my notifications' in the navigation under 'Settings' menu, you can configure which notifications you want to receive.

My Filters Search Sort: Date & Time

Alert
Pump overheating

ID	#6691	✓ Resolved by me Resolved
Date & time	2022 Nov 12 9:33:55	
Location	starter/BeerPump	
Originating app	VFC	

You can download Simatic Notifier app from [Google Play](#) store and [Apple App Store](#) to receive the event notification on your mobile phone by accessing the app with the permission.

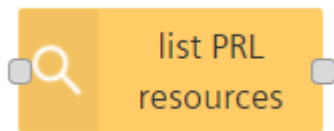
6.20 Predictive Learning (PRL) nodes

Usage of Predictive Learning nodes

Predictive Learning combines analytics, statistics and machine learning algorithms to provide unmatched insight into trends in your data. Combining predictive technology with IoT, service, field, and other data streams allows you to generate a deeper impact on the customer experience. Leveraging as-used data to identify patterns and sequences of events, companies can engage with customers and resolve potential issues before problems arise.

In Visual Flow Creator, Predictive Learning (PRL) nodes allows you to create, list and re-run the jobs by integrating with Predictive Learning application.

List PRL Resources



The "list prl resources" node lists all the available resources in Predictive Learning and stores the data in the message payload.

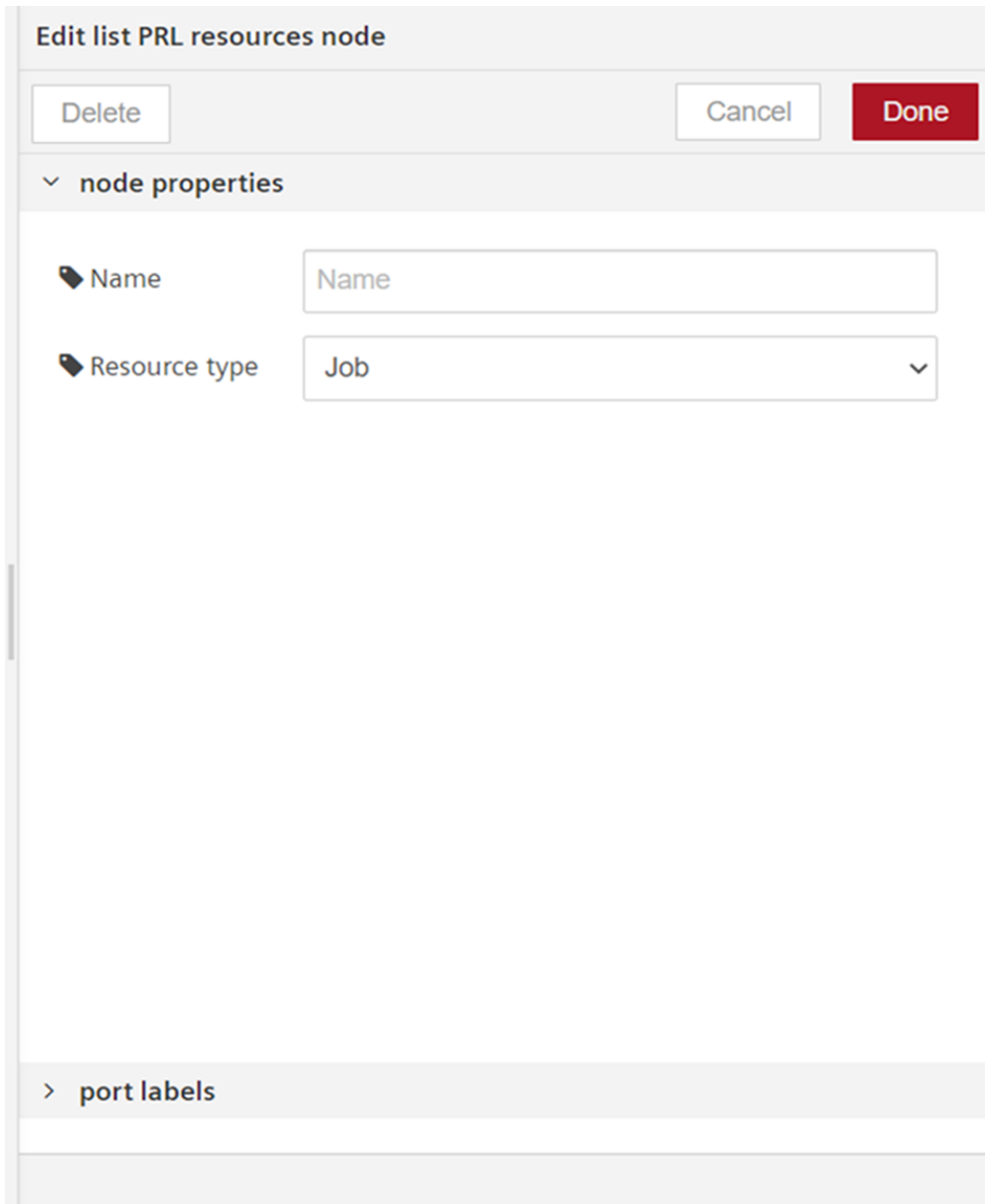
Example

To list all the available resources in Predictive Learning, follow these steps:

1. Create the flow as shown below:



2. Edit List PRL resource node and click :



Edit list PRL resources node

Delete Cancel Done

node properties

Name

Resource type

port labels

- Select "Resource type" and click "Done".

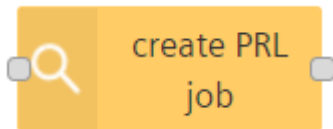
3. Save the flow.

Result

The following graphics shows in the message payload:

```
7/7/2022, 9:22:47.215 AM node: 177fdc8e.2432f3
msg: array[100]
▼ array[100]
▼ [0 ... 9]
  ▶ 0: object
  ▶ 1: object
  ▶ 2: object
  ▼ 3: object
      id: "2988de80-5f85-48bb-bc98-87213e0386e9"
      modelId: "26efd355-c371-416c-bdac-dffd90318e13"
      configurationId: "eecbe202-1e61-4014-afb4-75c2547a6f44"
      environmentId: "69f98701-fe2b-4984-87e1-9c9d23329abf"
      maxExecutionTime: 7200
      status: "SUCCEEDED"
      creationDate: "2022-06-30T07:54:26.944Z"
      createdBy: "prlteam"
```

Create PRL Job

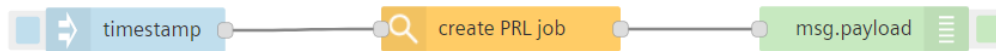



The "create PRL job" node allows you to create a new job in Predictive Learning through VFC and stores the data in the message payload. The status of the created job can be monitored via debug messages and Predictive Learning application. After the job is completed, object will be sent with all the internal information about completed job details.

Example

To create a new PRL job, follow these steps:

1. Create the flow as shown below:



Edit create PRL job node and click  to configure the node properties and then click

2. "Done":

Edit create PRL job node

Delete
Cancel
Done

▼ node properties

Name	<input type="text" value="Name"/>
Model	<input type="text" value="idlmodelv1"/> ...
Configuration	<input type="text" value="docker 2-120547dc-ad00-4dfc-b164-87"/> ...
Storage Type	<input type="text" value="DataLake"/> ▼
Input Folder	<input type="text" value="JunIDL050.3090059254505144"/> ...
Output Folder	<input type="text" value="JunIDL050.23440764335869735"/> ...
Start Command	<input type="text" value="Start Command"/>
Payload	▼ a_z
Description	<input type="text" value="Description"/>
Environment	+ Add new environment variable

> port labels

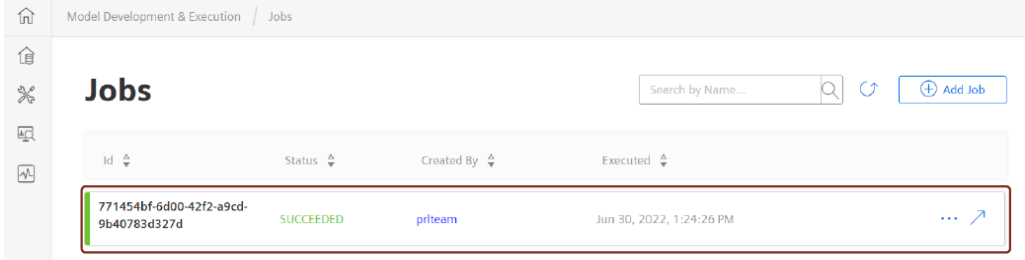
3. Save the flow.

Result

The following graphics shows the new PRL job created in the message payload and monitor the progress of the created job via debug messages:



You can also check the progress of the newly created job in Predictive Learning application.

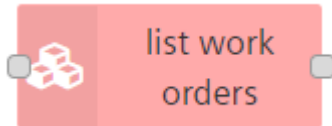


6.21 Work Order Nodes

With “Work orders” nodes, a basic workflow to digitize work requests (such as maintenance, repair, inspection and incident handling) that are essential to monitor asset health and to detect technical issues before they lead to asset failure and downtime. You can create, track and update work orders in multiple ways with Visual Flow Creator application. For more information about “Work orders”, refer to the chapter [Work orders](#) in Insights Hub Monitor documentation. In “Work orders” node palette, the following nodes are available:

- List Work Order
- Create Work Order
- Update Work Order
- Delete Work Order

List Work Order

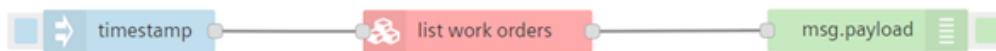


The "list work order" node allows you to list all the work orders created in the environment. If no handle information is given, you can read the specific work order by using handle ID. Handle ID is for user environment that describes the work order for further updates.

Example

To execute the example flow using list work order node to list the available work orders, follow these steps:

1. Create the flows as shown below:



2. Configure "Edit list-work-orders node" properties to list the available work orders:

The screenshot shows a configuration window for the "Edit list-work-orders node". At the top, there are three buttons: "Delete", "Cancel", and "Done". Below these is a section titled "node properties" with a dropdown arrow. This section contains two input fields: "Name" with the value "object" and "Handle" with the value "AA-003". At the bottom of the window, there is a section titled "port labels" with a right-pointing arrow.

3. Save and execute the flow.

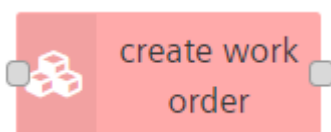
Result

You can view the results in the message payload.

```

2/6/2023, 8:05:26.631 PM node: 423d3aae.e4aa24
msg: array[2]
  array[2]
    0: object
      woHandle: "AA-001"
      dueDate: "2023-02-02"
      notifyAssignee: false
      title: "Title"
      type: "PLANNED"
      status: "OPEN"
      description: "just a test work order"
      priority: "MEDIUM"
      createdBy: "vfc-tu"
      createdAt: "2023-02-01T15:28:31.372Z"
      modifiedBy: "vfc-tu"
      modifiedDate: "2023-02-01T15:28:31.372Z"
      overdue: true
    1: object
      woHandle: "AA-000"
      dueDate: "2023-02-02"
      notifyAssignee: false
  
```

Create Work Order



The "create work order" node allows you to create the work order created in the environment.

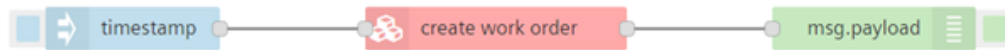
Node Properties	Description
Title	Define the title for the task or expectation.
Description	Enter the detailed information about the task or expectation.
Assigned to	Assign the task or expectation to the responsible assignee to work on the task. Assignee will be notified with an email.
Due date	Set the date as deadline for the task or expectation to be completed.

Node Properties	Description
Status	Update the task or expectation status while creating the work order. Open, In Progress, On Hold, Overdue and Done are the different statuses of the task or expectation can be assigned.
Priority	Prioritize the task or expectation based on the level of priority. Low, Medium, High and Emergency are the available priority options.
Type	Define the reason behind the initiation of the task or expectation. Planned and Incident are the options available types.
Associated asset	Associate the work order with an asset.
Attachment	Attach the document to capture or work to be done on the task or expectation.

Example

To execute the example flow using create work order node, follow these steps:

1. Create the flows as shown below:



2. configure "Edit create-work-order node" properties to create the work order:

Edit create-work-order node

Delete Cancel Done

node properties

Name: object

Title: MyNewWorkOrder

Description: First work order to be create for the asset

Assigned to: [blurred]

Due date: 2023/02/15

Status: Open

Priority: Low

Type: Incident

Associated asset: 2JZ-GTE (ebdeb4028b664e2bae:)

Attachment: ebdeb4028b664e2baeab95a546f:

port labels

3. Save and execute the flow.

Result

You can view the results in the message payload.

debug

2/6/2023, 9:22:00.491 PM node: dfc7b8db.f54278

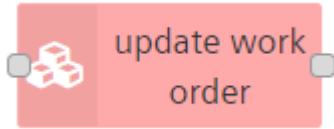
msg: Object

object

message: "Work order created with wo-handle AA-004"

woHandle: "AA-004"

Update Work Order



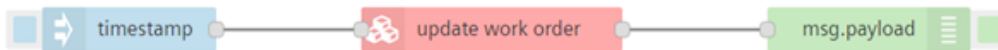
The "update work order" node allows you to update the created work order in the environment.

Node Properties	Description
Handle	Enter the work order ID to which describes work order for the user environment.
Title	Define the title for the task or expectation.
Description	Enter the detailed information about the task or expectation.
Assigned to	Assign the task or expectation to the responsible assignee to work on the task. Assignee will be notified with an email.
Due date	Set the date as deadline for the task or expectation to be completed.
Status	Update the task or expectation status while creating the work order. Open, In Progress, On Hold, Overdue and Done are the different statuses of the task or expectation can be assigned.
Priority	Prioritize the task or expectation based on the level of priority. Low, Medium, High and Emergency are the available priority options.
Type	Define the reason behind the initiation of the task or expectation. Planned and Incident are the options available types.
Associated asset	Associate the work order with an asset.
Attachment	Attach the document to capture or work to be done on the task or expectation.

Example

To execute the example flow using update work order node, follow these steps:

1. Create the flows as shown below:



2. Configure "Edit update-work-order node" properties to update the created work order:

Edit update-work-order node

Delete Cancel Done

node properties

Name object

Handle AA-004

Title MyNewWorkOrder

Description First work order to be create for the asset

Assigned to [blurred]

Due date 2023/02/15

Status Open

Priority Low

Type Incident

Associated asset 2JZ-GTE (ebdeb4028b664e2ba) ... x

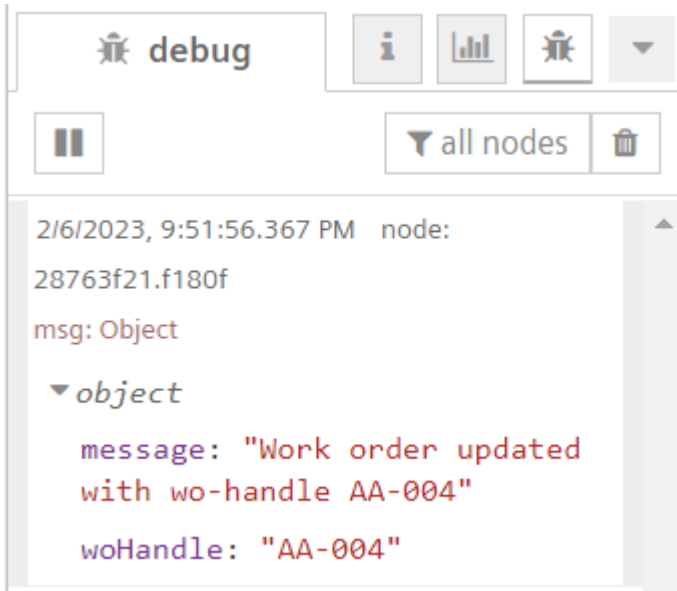
Attachment ebdeb4028b664e2baeab95a54 ... x

port labels

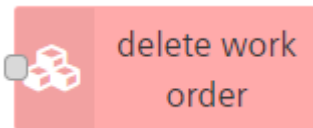
3. Save and execute the flow.

Result

You can view the results in the message payload.



Delete Work Order



The "delete work order" node allows you to delete the existing work order from the environment.

Example

To execute the example flow using delete work order node, follow these steps:

1. Create the flows as shown below:



Configure "Edit delete-work-order node" properties to delete the created work order using 2. "Handle" ID:

Dialog: Edit delete-work-order node

Buttons: Delete, Cancel, Done

node properties

Name: object

Handle: AA-004

port labels

3. Save and execute the flow.

Result

The created work order is successfully deleted.

6.22 MindConnect nodes

With "MindConnect" nodes, you can write the values of the datapoint, send a command to MindConnect MQTT device and check the status of the sent command. For more information about MindConnect Elements, refer to [Connectivity](#).

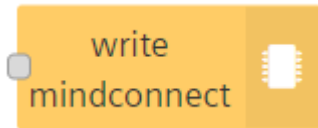


Ensure your MindConnect device is successfully onboarded before using these nodes.

In the "MindConnect" node palette, the following nodes are available:

- Write MindConnect
- Command MindConnect
- Command MindConnect Status

Write MindConnect



The "write mindconnect" node allows you to write a plain value to a datapoint of the onboarded MindConnect device.

Example

To execute the example flow using the write mindconnect node, follow these steps:

1. Create the flows as shown below:



2. Edit the inject node properties to enter the values:
 - Payload (Number): 127

3. Edit write mindconnect node properties:

Edit write mindconnect node

Delete Cancel Done

node properties

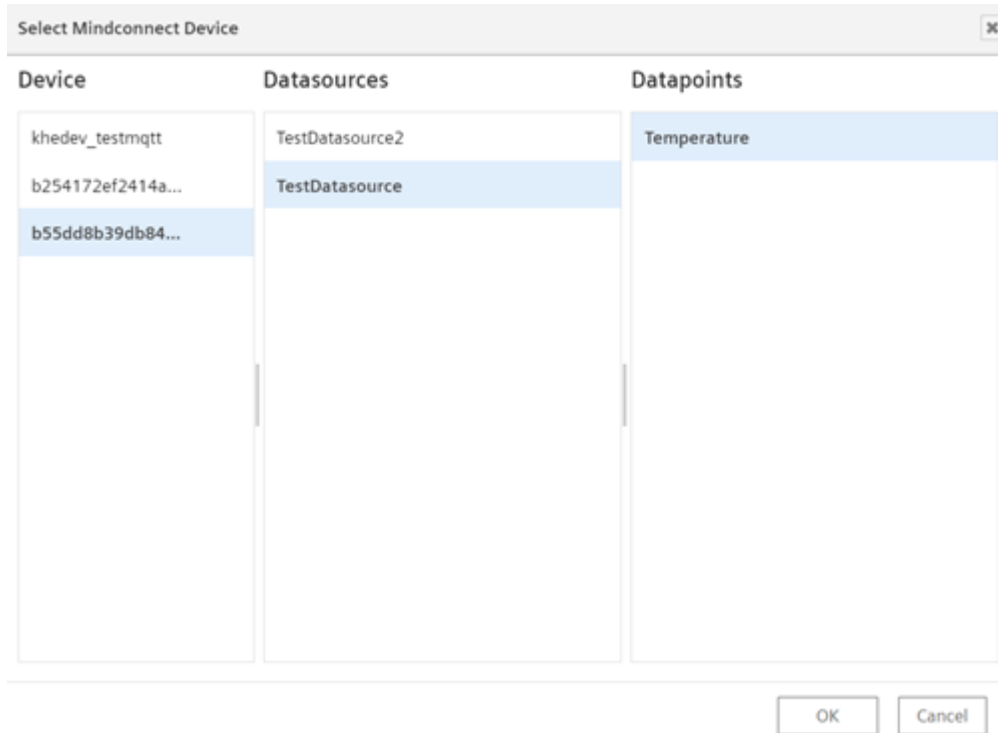
Name

Topic ...

> port labels

4. Click .

5. Select the "Device", "Datasources" and "Datapoints" to update the value.



6. Save and execute the flow.

Result

The datapoint value is successfully updated.

Command MindConnect



The "command mindconnect" node allows you to send a command to a MindConnect MQTT device.

Example

To execute the example flow using the command mindconnect node, follow these steps:

1. Create the flows as shown below:



2. Edit the function node properties to enter the values:

- Code:
`msg.payload = {"testprop5": "first"}; return msg;`

3. Edit the command mindconnect node properties:

The screenshot shows a dialog box titled "Edit command mindconnect node". At the top, there are three buttons: "Delete", "Cancel", and "Done". Below this is a section titled "node properties" with a dropdown arrow. It contains two fields: "Name" with a text input field containing "Name", and "MQTT Id" with a dropdown menu showing "MQTT Id" and a three-dot menu button. At the bottom, there is a section titled "port labels" with a right-pointing arrow.

4. Click  to select the MQTT device.

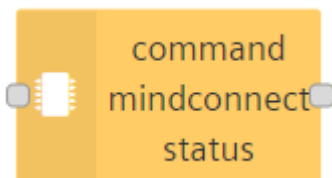
5. Save and execute the flow.

Result

You can view the results in the message payload. The command is successfully executed for the selected device.



Command MindConnect Status

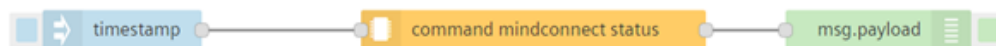


The “command mindconnect status” node allows you to get the status of a commanding job (MindConnect MQTT device).

Example

To execute the example flow using command mindconnect node, follow these steps:

1. Create the flows as shown below:



2. Edit command mindconnect status node properties:

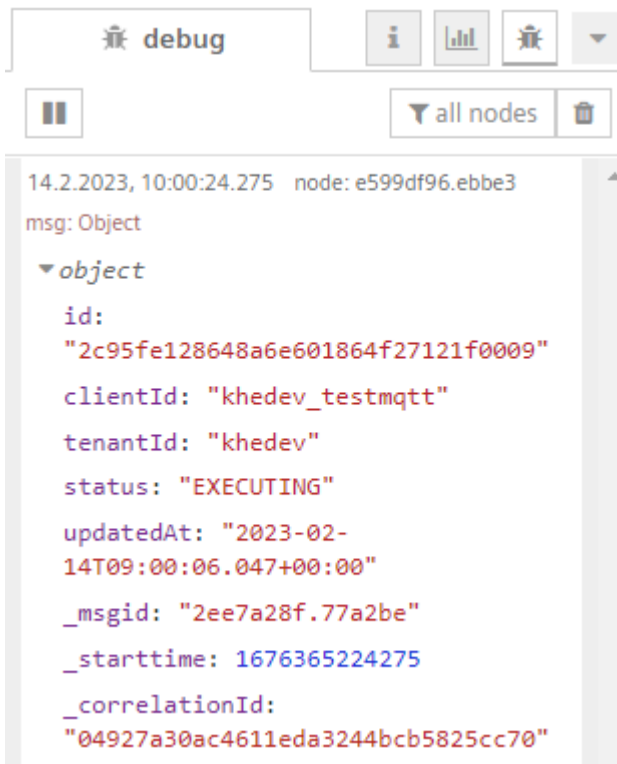
The screenshot shows a dialog box titled "Edit command mindconnect status node". At the top, there are three buttons: "Delete", "Cancel", and "Done". Below the buttons is a section titled "node properties" with a downward arrow. This section contains two input fields: "Name" and "JobId". Below the "node properties" section is a section titled "port labels" with a rightward arrow. The "Name" field contains the text "Name" and the "JobId" field contains the text "JobId".

3. Enter "JobId".

4. Save and execute the flow.

Result

You can view the results in the message payload. The status of the command is displayed successfully.



The screenshot shows a debug window with a toolbar at the top containing icons for information, a chart, a refresh symbol, and a dropdown arrow. Below the toolbar is a pause button and a filter button labeled 'all nodes' with a trash icon. The main area displays a log entry: '14.2.2023, 10:00:24.275 node: e599df96.ebbe3' followed by 'msg: Object'. A collapsed 'object' section is expanded to show the following JSON-like structure:

```
id: "2c95fe128648a6e601864f27121f0009"  
clientId: "kheDev_testmqtt"  
tenantId: "kheDev"  
status: "EXECUTING"  
updatedAt: "2023-02-14T09:00:06.047+00:00"  
_msgid: "2ee7a28f.77a2be"  
_starttime: 1676365224275  
_correlationId: "04927a30ac4611eda3244bcb5825cc70"
```

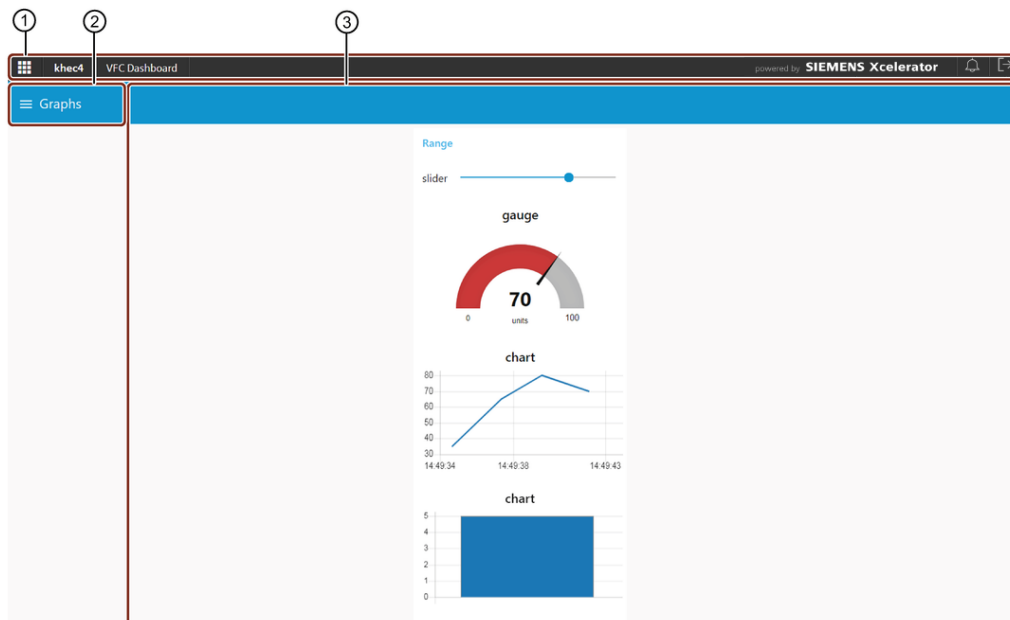
Dashboard in Visual Flow Creator

7

7.1 User Interface of Dashboard nodes output

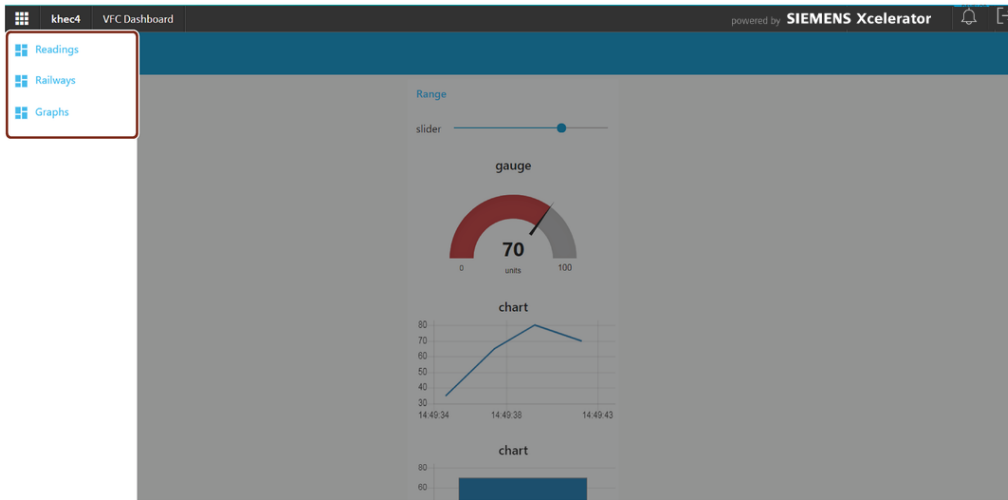
The dashboard nodes provide a set of nodes for reading, visualizing and analyzing data on a live dashboard.

Flow of dashboard nodes output screen



- ① OS bar
- ② Navigation drawer for tabs
- ③ Dashboard

User interface of group dashboard section



Select the required tab in the left panel to view the dashboard flow you have created. To learn more about tab and group creation, refer to [Create dashboard groups](#) in the "Node properties" section.

For more information about how to get started with a dashboard in VFC, refer to [VFC Dashboard - A Getting Started Example](#).

7.2 Usages of standard dashboard nodes

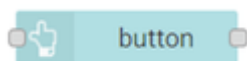
This section describes the dashboard nodes available in Visual Flow Creator.

You can edit the node properties as per requirements. You can set the state of dashboard widgets by sending messages to the corresponding nodes. You can read or react to values set by users of the dashboard by appending nodes to the outputs of the dashboard nodes.



Industrial IoT is not responsible for the complex dashboard nodes adopted from third party like Vega, Heat map, etc. For more information about the third party dashboard nodes, refer to their official websites.

Button



The node will generate a message with payload and topic defined in node settings. By default, the button sends a string with message id as payload. Clicking the button results in generating the message loaded in the payload.

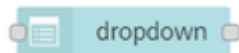
Output in the dashboard window:



CLICK TO DISPLAY INFORMATION

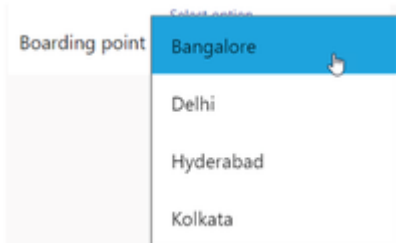
Here the size, color and label of the button has been edited to design a useful flow.

Dropdown



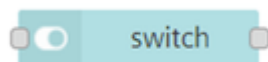
You can use the "dropdown" node to define multiple possible data values. Supported data types are string, number and boolean.

Output in the dashboard window:



You can customize the dropdown menu by editing the node properties. The "options" field in the node properties allows you to add the required options to design the dropdown menu.

Switch



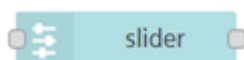
The switch node returns any kind of value (string/ number/ boolean/ json) when it changes state.

Output in the dashboard window:



The first switch shown is in the "off" mode, and the second switch is in the "on" mode.

Slider



You can set the value of the horizontal slider by sending a message to the node. You can react to a user's change to a slider by listening to the output message of the slider node. In order to both set the value and react to the new value, you can select "If msg arrives on input, pass through to output" in the node properties. A slider is defined in terms of the following parameters:

- A minimum value

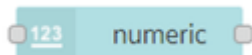
- A maximum value
- A step value

Input messages are converted to a number. The minimum value is used if conversion fails, and this will update the user interface or dashboard window. If the value changes, it will be passed to the output.

Output in the dashboard window:



Numeric

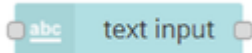


Like the slider, you can use the numeric node to define a range of values in steps with a minimum value and a maximum value.

Output in the dashboard window:



Text input



The node adds a text input to the dashboard window. The text input node supports the following modes:

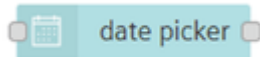
- Regular text
- Email address
- Password
- Number
- Telephone input
- Color picker
- Time picker
- Week picker
- Month picker

Output in the dashboard window:



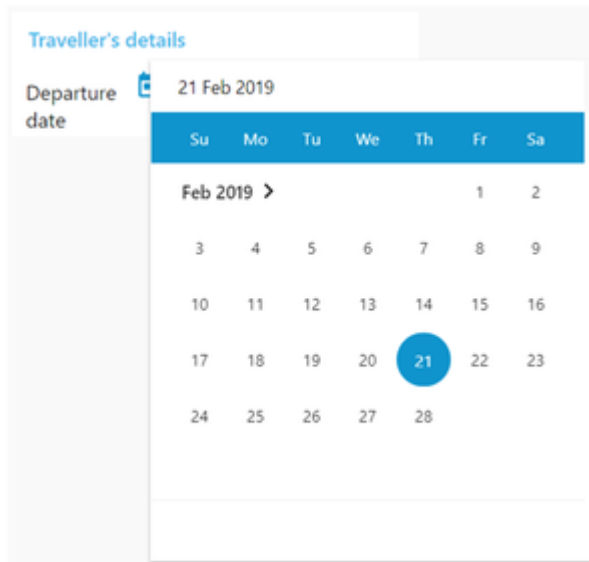
Here, the text input label has been changed to "Enter your name here".

Data picker



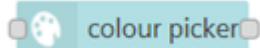
The date picker node allows you to pick dates from the widget.

Output in the dashboard window:



Here, the date picker label has been changed to "Departure date".

Colour picker



This node uses the color selector of the system and returns the color in rgb, hex, hex8, hsv or hsl format. Transparency is also supported for all except hex.

Output in the dashboard window:



The node label has been edited to "Add colour". Click the color box and a dropdown to choose color will appear. You can then choose the required color.

Form



The form node helps you to accumulate multiple form elements and then submit them on a single click as an object.

Output in the dashboard window:

Traveller's details

Registration form

Enter your name *

Type your email *

Choose a password *

Type your password again *

Address

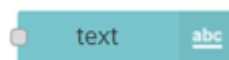
Contact number

Single lady

REGISTER **CANCEL**

You can edit the node properties of the "form" and design the form in the "form elements" section according to requirements.

Text



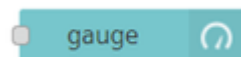
The text node is a read-only widget where you can configure the label and value.

Output in the dashboard window:

Write your address here

You can customize the text box label for a read-only message.

Gauge

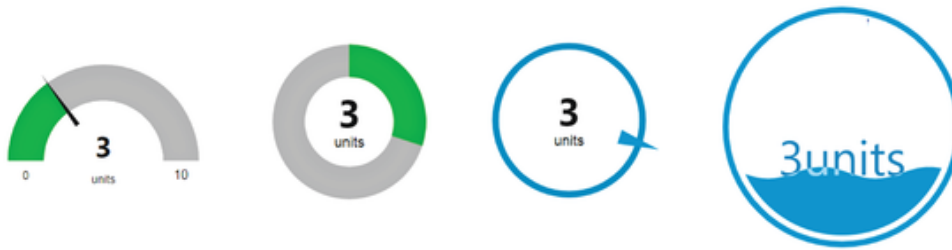


The gauge node is of four types:

- Standard gauge
- Donut
- Compass
- Level

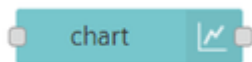
You can define the ranges with different colors for the gauge nodes.

Output in the dashboard window:



The "gauge" dashboard widget can be customized to appear as one of four possible types.

Chart

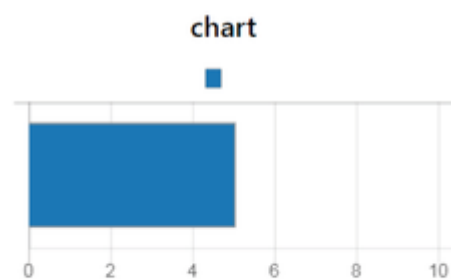
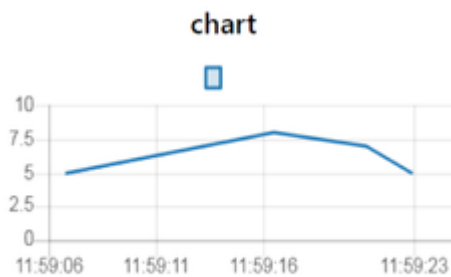


The chart node can be designed in the following formats:

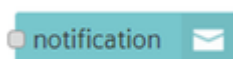
- Line
- Bar
- Pie

Chart nodes are helpful for multiple data readings. Charts are useful for representations and comparisons of numeric and quantitative information.

Output in the dashboard window:

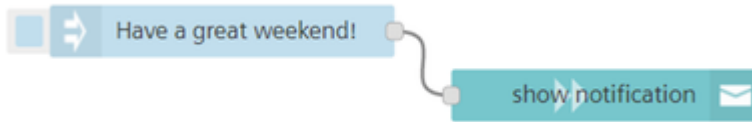


Notification



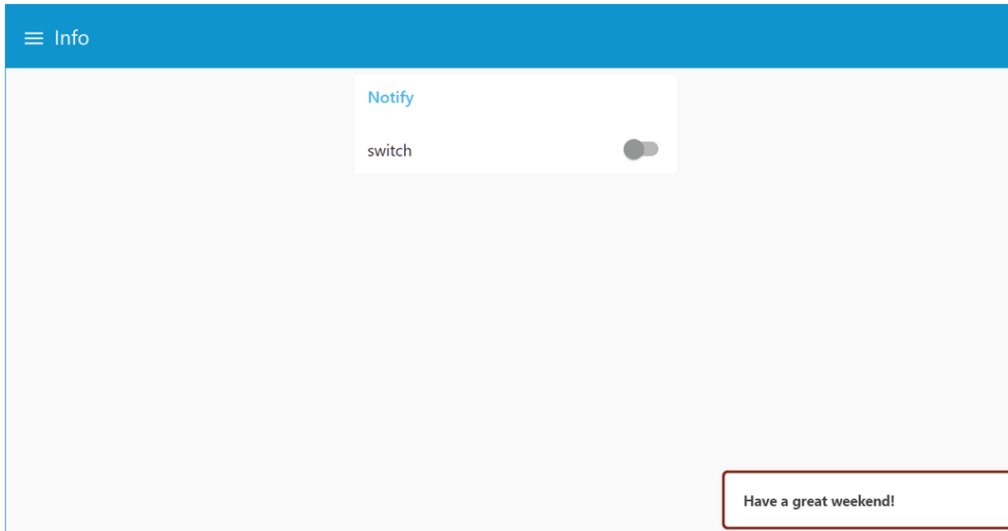
The node creates user alerts as popup notifications.

Output in the dashboard window:

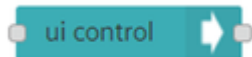


Let us consider the following example:

The notification is configured to be displayed on the bottom right. You get the following message in the dashboard window:

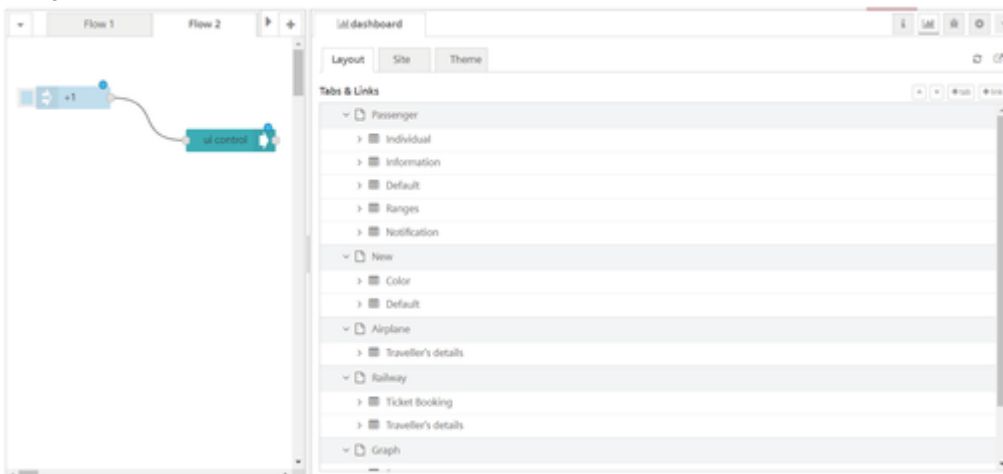


UI control



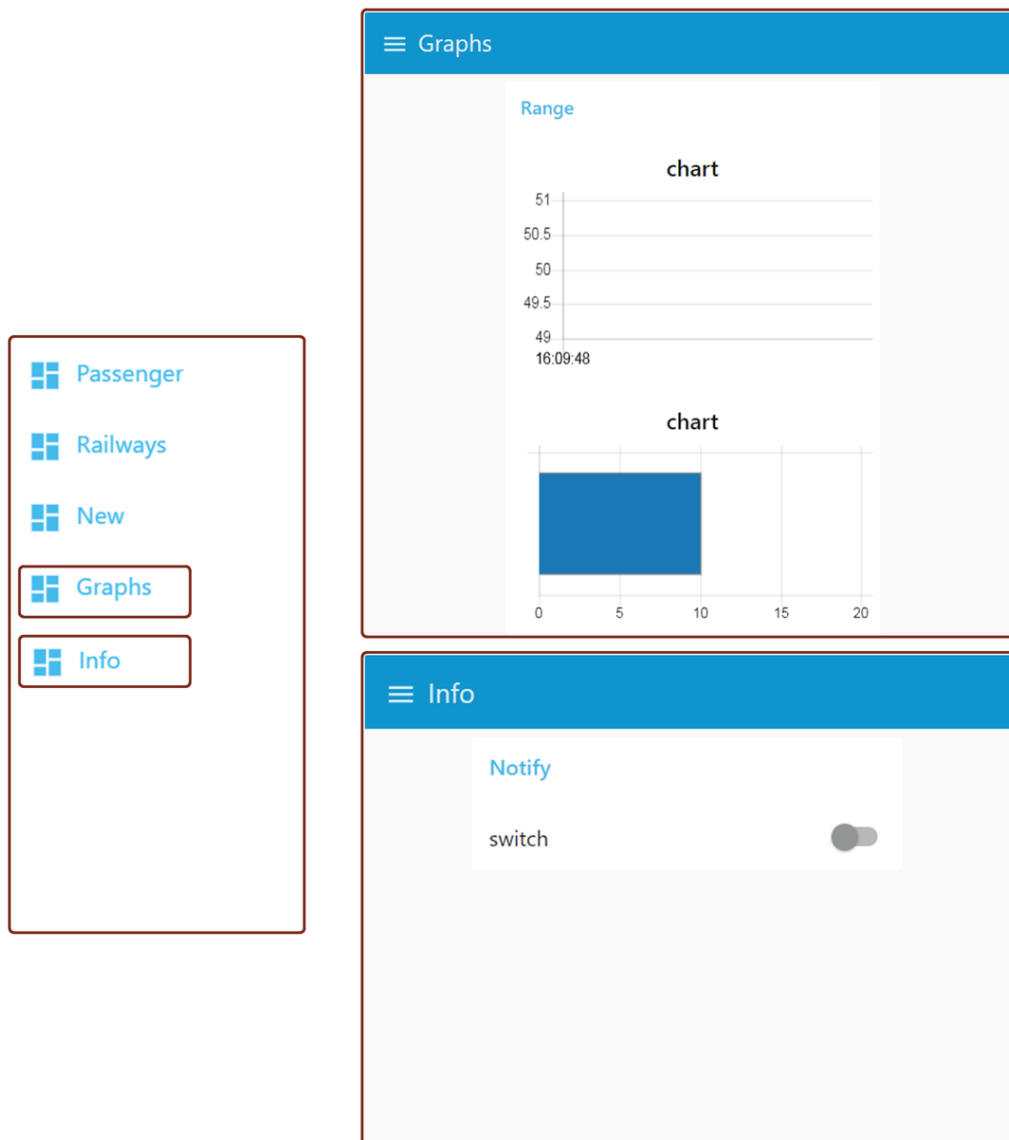
The ui control node allows some dynamic control of the dashboard. Sending a msg.payload of the tab number (from 0) or tab_name will switch to that tab. It refreshes the current opened page if the message contains an empty or unset "tab" property.

Output in the dashboard window:



Let us consider the following flow with the defined dashboard groups in the sidebar:

The input node is defined with string "+1". This opens the next tab incrementally in the dashboard window.

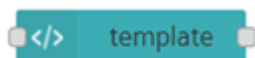


Similarly, if you define the input node with string "-1", the dashboard window will change to the previous tab, in a reverse order.

When you perform some activity in the dashboard window, this node connected with debug node will display the message of payload and `loggedInUser` details in the message payload.

```
8/16/2022, 11:07:33.492 AM  node:
4d0de465.3489cc
msg: Object
  ▾ object
    tab: "a4655911.4318c8"
    ▶ query: object
      socketid: "E5URVgcfvVFCd9aIAANm"
      name: "Railways"
    ▶ payload: object
      loggedInUser:
        "brown.muller@siemens.com"
      _msgid: "8c721e8a.f53bf"
      _starttime: 1660628253491
      _correlationId:
        "8709b8301d2511edb8b17f94d3b9cc75"
```

Template



The template node allows you to create and define your own widgets. You can design the widgets within the framework using HTML, JavaScript. You can also override builds in CSS styles. You can create a template and save it in the library for re-use.

Output in the dashboard window:

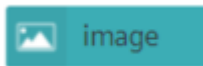
Let us consider the simple code given below:

[simple code](#)

You get the following message in the dashboard window:

Have a nice day!

Image



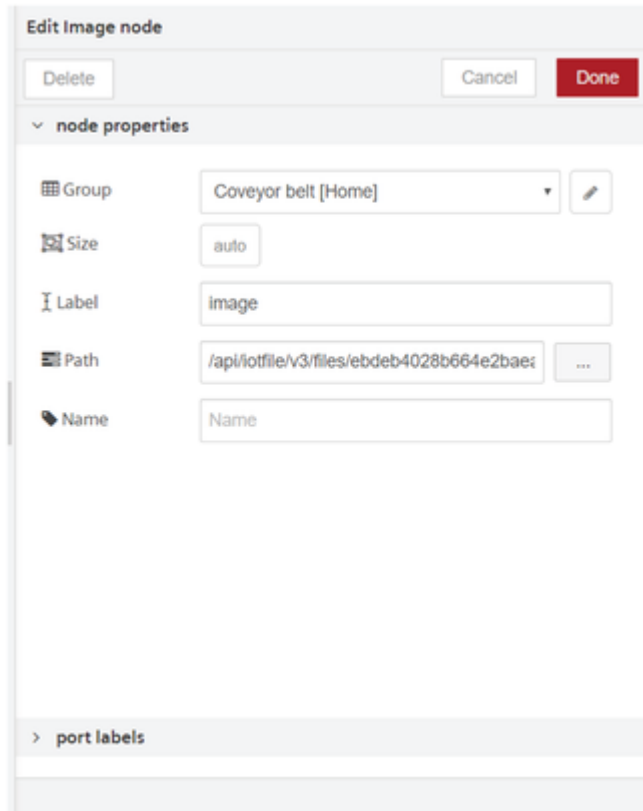
The image node allows you to add an image to the dashboard in the user interface. It supports different types of image formats. The image file can be uploaded using Insights Hub Monitor.


The path represents the File API path to image resource.

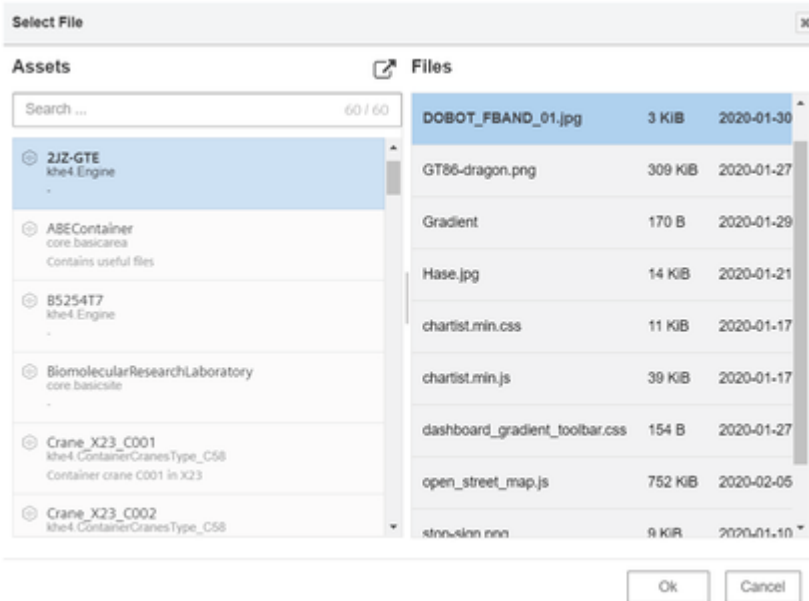
Procedure to upload the image

To upload the image file to image node, follow these steps:

1. Double click the image node.



2. Click  to add the image from the "Select an Image" dialog box.

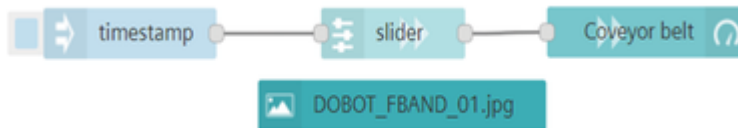


3. Select file and click "Ok".

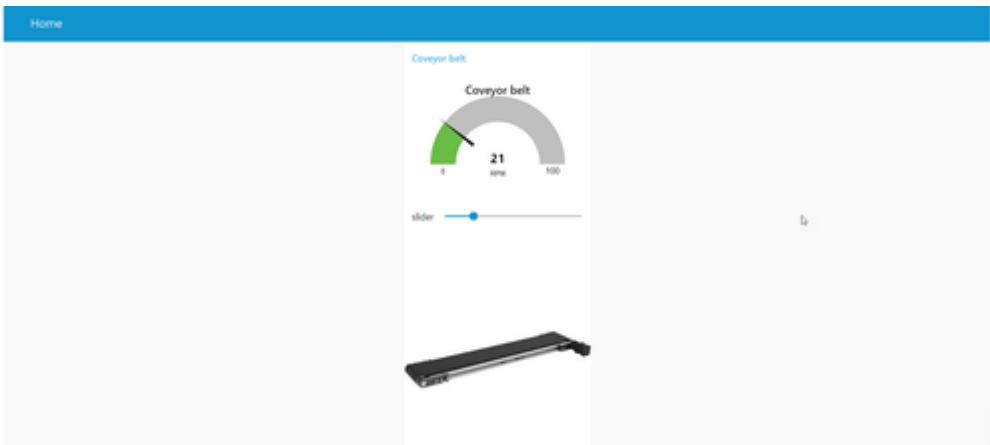
4. Click "Done".


Output in the dashboard window

Let us consider the following flow as an example:



Open Visual Flow Creator dashboard to view the output.





The image should be available in Insights Hub Monitor application before using it to Image node.

JS/CSS

JS JS/CSS

The JS or CSS node allows you to embed custom JS and/or CSS files to the VFC Dashboard. All JS/CSS files will be embedded according to their sequence numbers. The .js or .css file can be uploaded using Insights Hub Monitor.

Procedure to upload the .js or .css file

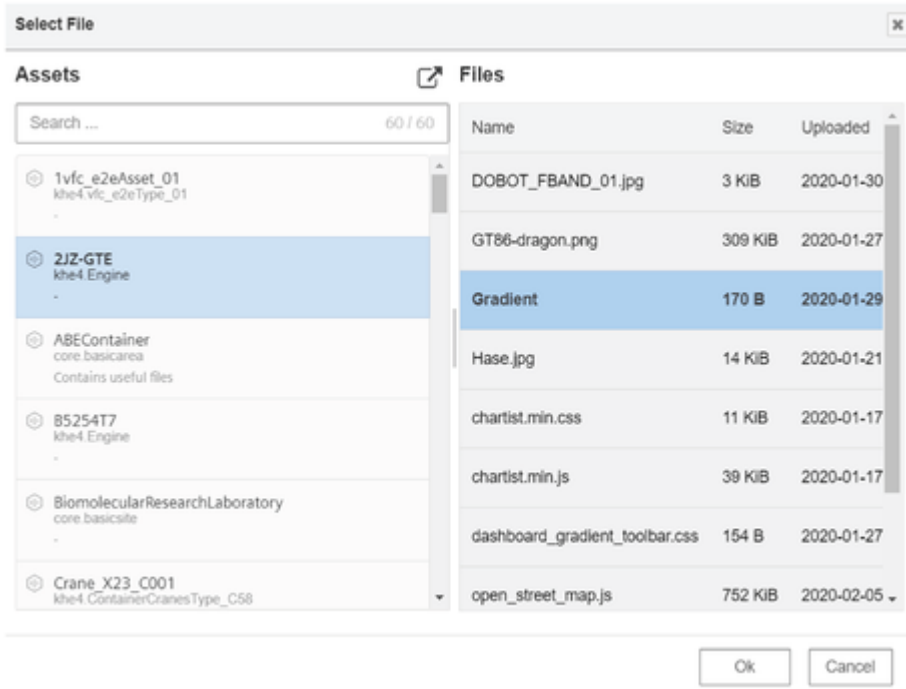
To upload the .js or .css file to JS/CSS node, follow these steps:

1. Double click the JS/CSS node.

① Insights Hub Monitor plugin

② Operations Insight plugin

2. Click "Select File" button.



3. Select file and click "Ok".

4. Select the sequence number.

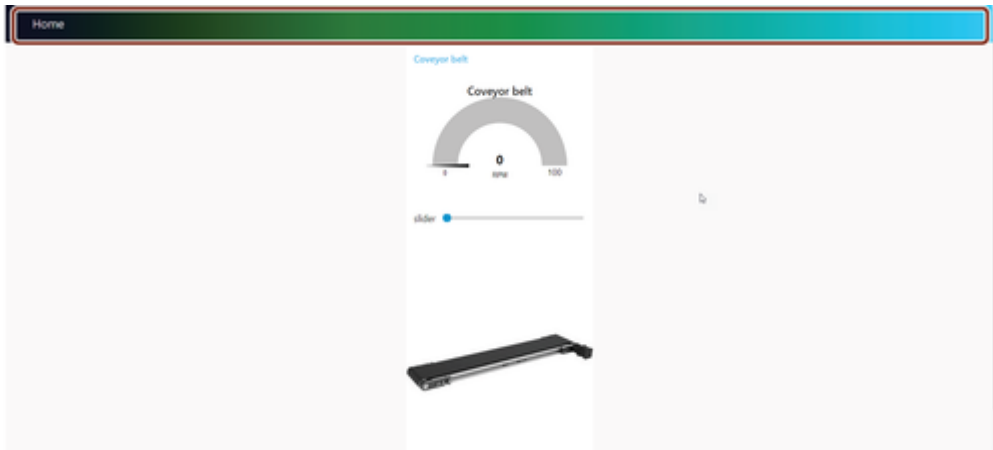
5. Click "Done".

Output in the dashboard window:

Let us save the below code in .css file and upload the file to Insights Hub Monitor.

```
#nr-dashboard-toolbar {
    background: rgb(2,0,36);
    background: linear-gradient(90deg, rgba(2,0,36,1) 0%, rgba(41,121,9,1) 35%, rgba(0,206,255,1) 100%);
}
```

For example, the above code will change the tab ribbon color from blue to gradient as shown below:





- The file should be available in Insights Hub Monitor application before using it to JS/CSS node.
- Define the sequence number to order the files which will be embedded into the HTML page.

Use Insights Hub Monitor and Operations Insight plugin to load the dashboard for all the users. Now, it is possible to create the dashboards from one user to another user.



The scripts are loaded with a delay after the page loads, you can use the below example code to load the page with a delay.

- Example code:

```
html <script> setTimeout (() => {$('#myID').html('Hi')}; }, 100);  
</script> <div ng-bind-html="msg.payload" id="myID"> </div>
```

IFrame



The IFrame node allows you to embed an external webpage in Visual Flow Creator dashboard.

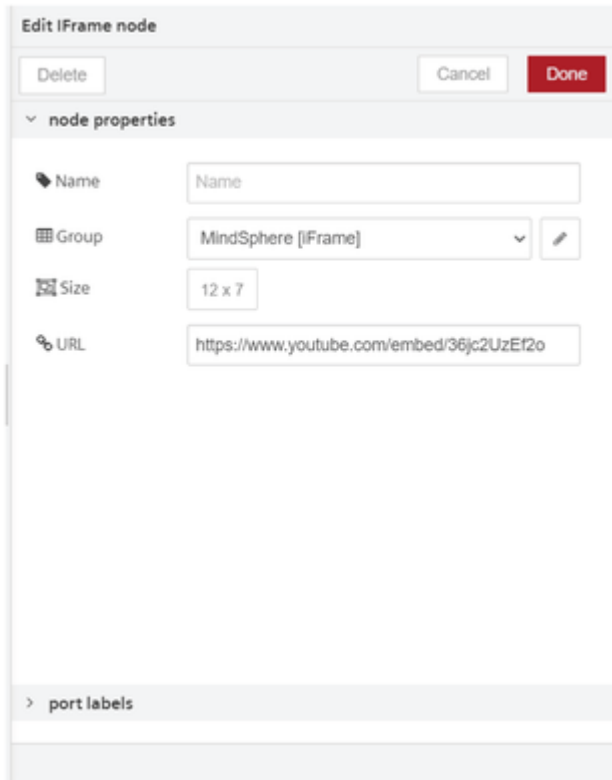


Adhere to the Content Security Policy (CSP) of the external websites for the protection against attacks, malicious content and other mitigates issues.

Example to embed an external webpage**

To embed the external webpage in Visual Flow Creator dashboard using iFrame node, follow these steps:

1. Double click the iFrame node.

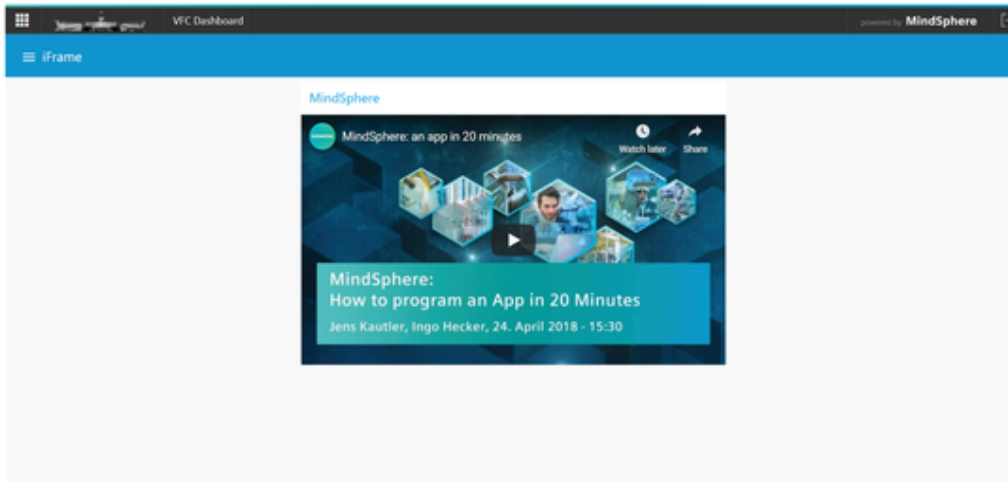


2. Enter the URL.

3. Click Done.

Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output.



Minerva



The Minerva node allows you to add the web widgets to the dashboard. It includes the following widgets:

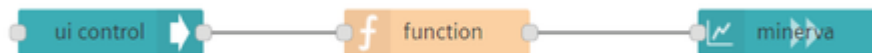
- Bullet Chart
- Reference Band
- Recursive Pattern
- Cyclic Angle Time Spiral
- Smart Color Designer
- DBSCAN Param
- DBSCAN Scoring
- Ranked Stream Graph
- Confusion Matrix

Each widget will be configured using data taken from msg.payload. For more information about the widgets, refer to the info tab of minerva node.

Example

To add the web widgets to the dashboard, follow these steps:

1. Create the flow as shown below:



2. Edit function node:

- Code:

??? Code

```

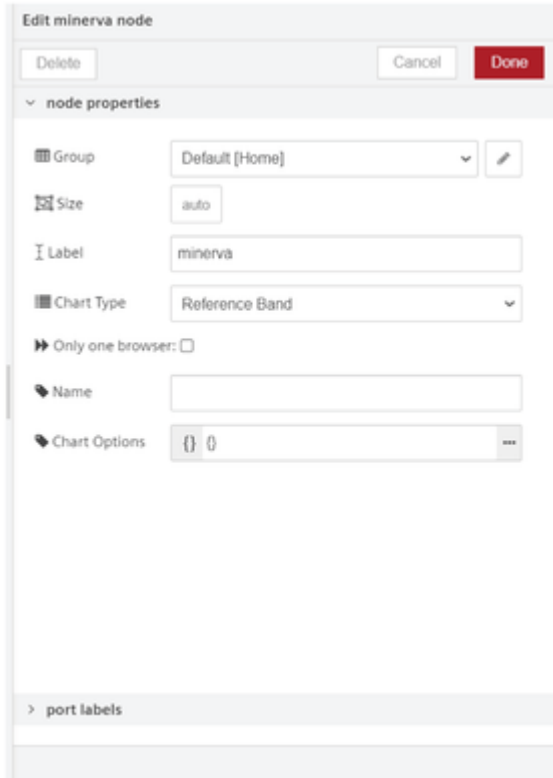
```json
{
 "data": [
 {
 "timestamp": "2016-04-05 00:00:30.976",
 "id": "671121446"
 },
 {
 "timestamp": "2016-04-05 00:00:33.984",
 "id": "1000005"
 },
 {
 "timestamp": "2016-04-05 00:00:33.988",
 "id": "1000005"
 },
 {
 "timestamp": "2016-04-05 00:00:35.98",
 "id": "671121446"
 }
]
}
```

```

```
    },  
    {  
      "timestamp": "2016-04-05 00:00:39.992",  
      "id": "671121446"  
    },  
    {  
      "timestamp": "2016-04-05 00:00:42.996",  
      "id": "679477920"  
    },  
    {  
      "timestamp": "2016-04-05 00:00:44",  
      "id": "679477924"  
    },  
    {  
      "timestamp": "2016-04-05 00:01:50.004",  
      "id": "679477929"  
    },  
    {  
      "timestamp": "2016-04-05 00:01:57.008",  
      "id": "679477921"  
    },  
    {  
      "timestamp": "2016-04-05 00:02:38.012",  
      "id": "679477928"  
    },  
    {  
      "timestamp": "2016-04-05 00:03:29.016",  
      "id": "687964883"  
    },  
    {  
      "timestamp": "2016-04-05 00:04:02.02",  
      "id": "1000005"  
    },  
    {  
      "timestamp": "2016-04-05 00:04:02.024",  
      "id": "1000005"  
    }  
  ],  
  "options": {  
    "height": "400px",  
    "variableStreamWidth": false,  
    "nrOfPeriods": 7,  
    "nrOfRanks": 6,  
  }
```

```
"periodWidth": 70,  
"averageStreamWidth": 60,  
"extensionsHeight": 100,  
"colorscheme": [  
  "#33a02c",  
  "#a6cee3",  
  "#fb9a99",  
  "#b2df8a",  
  "#fdbf6f",  
  "#1f78b4",  
  "#e31a1c",  
  "#ffff99",  
  "#cab2d6",  
  "#ff7f00",  
  "#b15928",  
  "#6a3d9a",  
  "#8dd3c7",  
  "#ffffb3",  
  "#bebada",  
  "#fb8072",  
  "#80b1d3",  
  "#fdb462",  
  "#fccde5",  
  "#b3de69",  
  "#ffed6f",  
  "#d9d9d9",  
  "#bc80bd",  
  "#ccebc5"  
]  
}  
}  
...
```

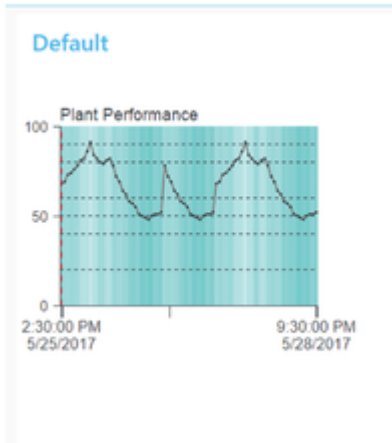
3. Edit minerva node:



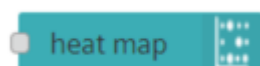
4. Save the flow.

Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output.



Heat map



The Heat map node allows you to create a graphical representation of the data, where the individual values contained in a matrix are represented as colors. For more information, refer to the info tab of heat map node.

Example

To execute the example flow using heatmap node, follow these steps:

1. Create the flow as shown below:



2. Edit function node properties:

- Name: Generate random matrix
- Code:

```

var len = 200;
msg.payload = [];
while (len--){
var value = Math.floor(Math.random()*100);
msg.payload.push(value);
}
return msg;
  
```

3. Edit heat map node properties.

Delete
Cancel
Done

▼ node properties

Group Data [Heatmap] ✎

Size 6 x 5

Grid size Rows Columns

Specify minimum and maximum value

Grid values Keys ▼

Legend None ▼

Background None (transparent) ▼

Radius

Opacity

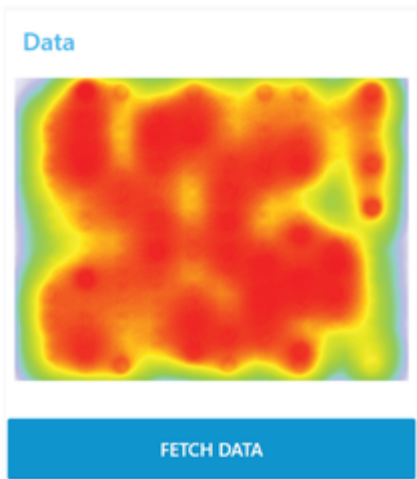
Blur

Name

> port labels

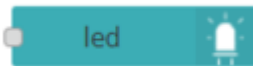
4. Save and execute the flow.

Output in the dashboard window:



Open Visual Flow Creator dashboard to view the output.

LED



The LED node allows you to create a simple LED status indicator for the Dashboard. For more information, refer to the info tab of LED node.

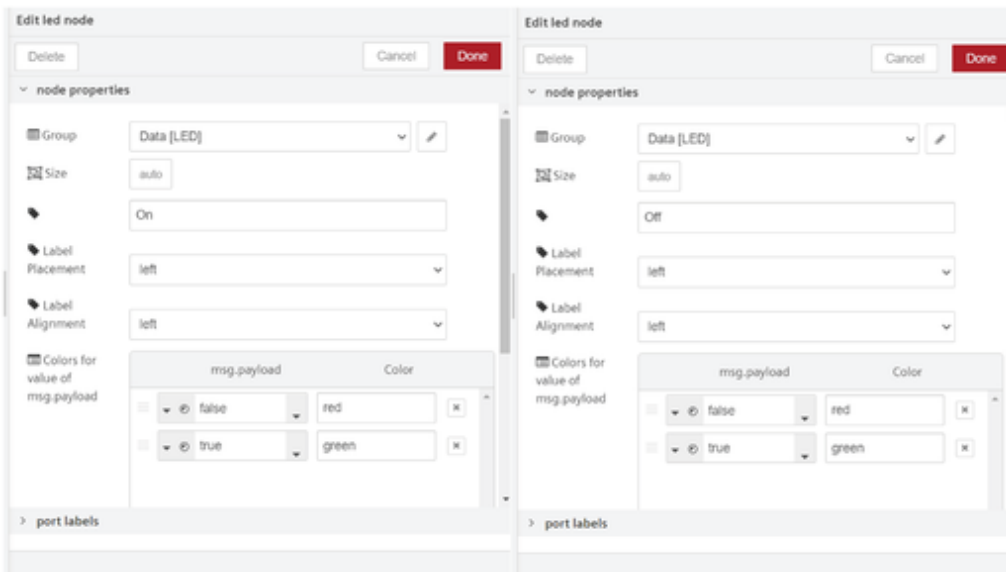
Example

To execute the example flow using LED node, follow these steps:

1. Create the flow as shown below:



2. Edit LED node properties.





To add more colors to the LED node, click "+Color".

3. Save and execute the flow.

Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output.

Data

On



Data

Off



Linear gauge

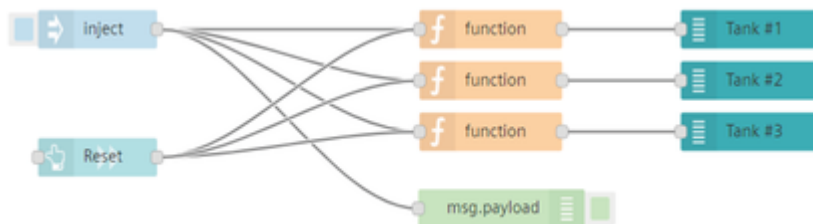


The Linear gauge node allows you to create a linear gauge with high or low limits and animated sliding pointer. For more information, refer to the info tab of linear gauge node.

Example

To execute the example flow using linear gauge node, follow these steps:

1. Create the flow as shown below:



2. Edit function (Tank #1) node properties:

◦ Code:

```
msg.payload = 80;  
msg.highlimit = 150;  
msg.lowlimit = 10;  
msg.setpoint = 75;  
return msg;
```

3. Edit function (Tank #2) node properties:

◦ Code:

```
msg.payload = 35;  
msg.highlimit = 50;  
msg.lowlimit = 12;  
msg.setpoint = 45;  
return msg;
```

4. Edit function (Tank #3) node properties:

◦ Code:

```
msg.payload = 56;  
msg.highlimit = 250;  
msg.lowlimit = 5;  
msg.setpoint = 88;  
return msg;
```

5. Edit linear gauge (Tank #1) node properties.

The screenshot shows the 'Edit linear gauge node' dialog box. It has a title bar 'Edit linear gauge node' and buttons for 'Delete', 'Cancel', and 'Done'. The 'node properties' section is expanded and contains the following fields:

- Group:** Data [Linear Gauge]
- Size:** 2 x 5
- High Area Color:** Red
- Middle Area Color:** Green
- Low Area Color:** Yellow
- Unit:** °C
- Name:** Tank #1

At the bottom, there is a section for 'port labels' which is currently collapsed.

6. Edit linear gauge (Tank #2) node properties.

Dialog: Edit linear gauge node

Buttons: Delete, Cancel, Done

node properties

- Group: Data [Linear Gauge]
- Size: 2 x 5
- High Area Color: red
- Middle Area Color: purple
- Low Area Color: green
- Unit: *C
- Name: Tank #2

> port labels

7. Edit linear gauge (Tank #3) node properties.

Dialog: Edit linear gauge node

Buttons: Delete, Cancel, Done

node properties

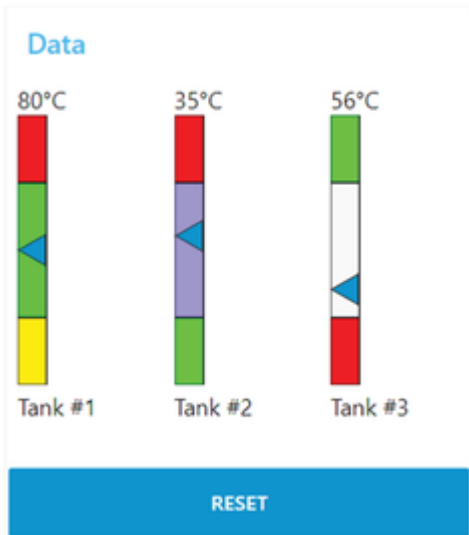
- Group: Data [Linear Gauge]
- Size: 2 x 5
- High Area Color: green
- Middle Area Color: white
- Low Area Color: red
- Unit: *C
- Name: Tank #3

> port labels

8. Save and execute the flow.

Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output.



Microphone



The Microphone node allows you to create an audio to be recorded and allows speech recognition from the dashboard. For more information, refer to the info tab of microphone node.

Example

To execute the example flow using microphone node, follow these steps:

1. Create the flow as shown below:



2. Save and execute the flow.

Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output and start recording from the dashboard.



Statechart



The Statechart node allows you to create a bar chart to visualize numeric values in relation, together with state represented by color. For more information, refer to the info tab of statechart node.

Example

To execute the example flow using statechart node, follow these steps:

1. Create the flow as shown below:

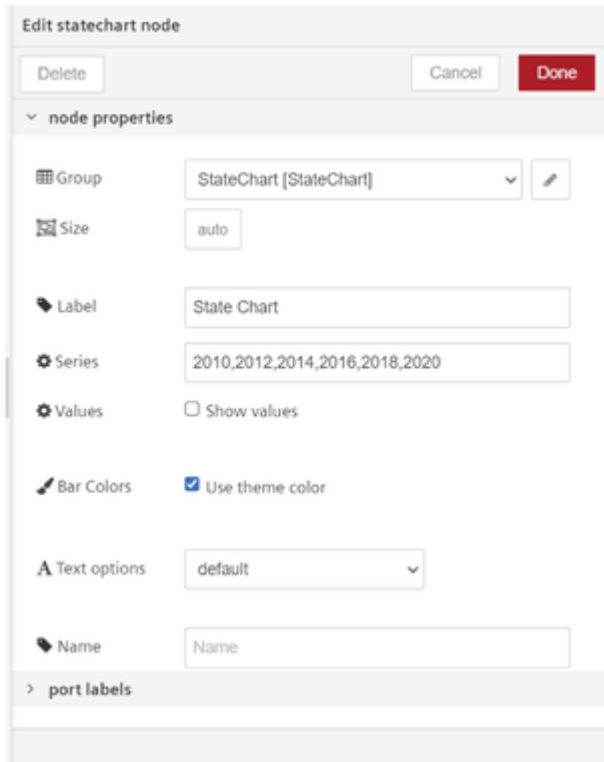


2. Edit function node properties:

- Code:

```
msg.payload = [{
  series: "2010",
  value: 300000,
  state: true
},
{
  series: "2012",
  value: 400000,
  state: true
},
{
  series: "2014",
  value: 500000,
  state: true
},
{
  series: "2016",
  value: 600000,
  state: true
},
{
  series: "2018",
  value: 700000,
  state: true
},
{
  series: "2020",
  value: 800000,
  state: true
}]
return msg;
```

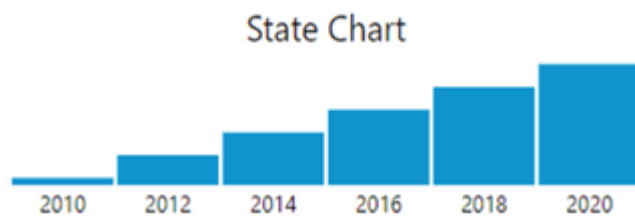
3. Edit statechart node properties:




4. Save and execute the flow.

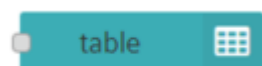
Output in the dashboard window

Open Visual Flow Creator dashboard to view the output.



 The default color theme can be changed using "Bar Colors" and "Text options" in "State chart" node properties.

Table



The Table node allows you to display the data as a table in the dashboard. For more information, refer to the info tab of table node.

Example

To execute the example flow using table node, follow these steps:

1. Create the flow as shown below:



2. Edit function node properties:

◦ Code:

```

msg.payload = [
  {
    "Name": "Steve",
    "Age": "33",
    "Place": "Paris",
    "Country": "France"
  },
  {
    "Name": "Sofia",
    "Age": "30",
    "Place": "Barcelona",
    "Country": "Spain"
  }
];
return msg;
  
```

3. Save and execute the flow.

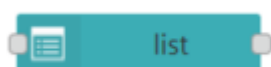
Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output.

Table

| Name ▲ | Age ▲ | Place ▲ | Country |
|--------|-------|-----------|---------|
| Steve | 33 | Paris | France |
| Sofia | 30 | Barcelona | Spain |

List



The List node allows you to create a list of the items. For more information, refer to the info tab of list node.

Example

To execute the example flow using list node, follow these steps:

1. Create the flow as shown below:



2. Edit inject node properties:

- Name: Insights Hub services
 - Payload (JSON):

```

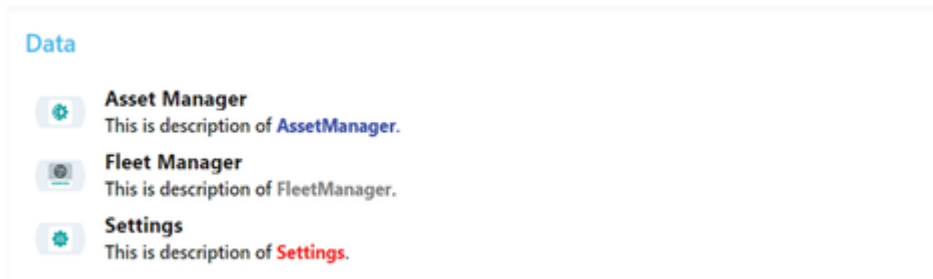
[
  {
    "title": "<b>Asset Manager</b>",
    "description": "This is description of <fontcolor=\"blue\"><b>AssetManager</b></b>.",
    "icon": "https://siemens.mindsphere.io/en/docs/mindaccess/_jcr_content/root/container/division/content/dynamiccard/head/image.coreimg.svg/1596115731456/asset-manager-docsteaserkiosk-640x384.svg"
  },
  {
    "title": "<b>Insights Hub Monitor</b>",
    "description": "This is description of <font color=\"Grey\"><b>FleetManager</b></font>.",
    "icon": "https://siemens.mindsphere.io/en/docs/mindaccess/_jcr_content/root/container/division/content/dynamiccard_copy_920548260/head/image.coreimg.svg/1597391606440/fleet-manager-docsteaserkiosk-640x384.svg"
  },
  {
    "title": "<b>Settings</b>",
    "description": "This is description of <font color=\"red\"><b>Settings</b></font>.",
    "icon": "https://siemens.mindsphere.io/en/docs/mindaccess/_jcr_content/root/container/division/content/dynamiccard_copy/head/image.coreimg.svg/1596115743466/settings-docsteaserkiosk-640x384-2.svg"
  }
]
  
```


3. Edit list node properties:

4. Save and execute the flow.

Output in the dashboard window:

Open Visual Flow Creator dashboard to view the output.



Vega



The Vega node allows you to create a declarative data visualization using Vega visualization grammar. It also supports the Vega-lite specification. For more information, refer to the info tab of vega node.



If you select "Only one browser" for a node in a flow, the node will handle each of the dashboards separately. For each browser, the communications originating from the node will be exclusively handled. If "Only one browser" is not selected, the node will share all the dashboard communication with every open browser from that user. For more information about an example on usage of one and multiple browsers, refer to [Application examples on node properties](#).

Example

To execute the example flow using vega node, follow these steps:

1. Create the flow as shown below:



2. Edit inject node properties:

- Payload (JSON):

```

??? Code ``json { "$schema": "https://vega.github.io/schema/vega/v5.json", "description": "An annotated line chart of the population of Falkensee, Germany.", "width": 500, "height": 250, "padding": 5, "config": { "title": { "fontSize": 16 } }, "title": { "text": { "signal": "Population of Falkensee from ' + years[0] + ' to ' + years[1]" } },
  
```

```

    "data": [
      {
        "name": "table",
        "values": [
          {"year": 1875, "population": 1309},
          {"year": 1890, "population": 1558},
          {"year": 1910, "population": 4512},
          {"year": 1925, "population": 8180},
          {"year": 1933, "population": 15915},
          {"year": 1939, "population": 24824},
          {"year": 1946, "population": 28275},
          {"year": 1950, "population": 29189},
          {"year": 1964, "population": 29881},
          {"year": 1971, "population": 26007},
          {"year": 1981, "population": 24029},
          {"year": 1985, "population": 23340},
          {"year": 1989, "population": 22307},
          {"year": 1990, "population": 22087},
          {"year": 1991, "population": 22139},
          {"year": 1992, "population": 22105},
        ]
      }
    ]
  
```

```

        {"year": 1993, "population": 22242},
        {"year": 1994, "population": 22801},
        {"year": 1995, "population": 24273},
        {"year": 1996, "population": 25640},
        {"year": 1997, "population": 27393},
        {"year": 1998, "population": 29505},
        {"year": 1999, "population": 32124},
        {"year": 2000, "population": 33791},
        {"year": 2001, "population": 35297},
        {"year": 2002, "population": 36179},
        {"year": 2003, "population": 36829},
        {"year": 2004, "population": 37493},
        {"year": 2005, "population": 38376},
        {"year": 2006, "population": 39008},
        {"year": 2007, "population": 39366},
        {"year": 2008, "population": 39821},
        {"year": 2009, "population": 40179},
        {"year": 2010, "population": 40511},
        {"year": 2011, "population": 40465},
        {"year": 2012, "population": 40905},
        {"year": 2013, "population": 41258},
        {"year": 2014, "population": 41777}
    ],
    "transform": [
        {
            "type": "extent",
            "field": "year",
            "signal": "years"
        }
    ]
},
{
    "name": "annotation",
    "values": [
        {
            "start": 1933,
            "end": 1945,
            "text": "Nazi Rule"
        },
        {
            "start": 1948,
            "end": 1989,
            "text": "GDR (East Germany)"
        }
    ]
}

```

```
        }
      ]
    }
  ],

  "scales": [
    {
      "name": "x",
      "type": "linear",
      "range": "width",
      "zero": false,
      "domain": {"data": "table", "field": "year"}
    },
    {
      "name": "y",
      "type": "linear",
      "range": "height",
      "nice": true,
      "zero": true,
      "domain": {"data": "table", "field": "population"}
    },
    {
      "name": "color",
      "type": "ordinal",
      "domain": {"data": "annotation", "field": "text"},
      "range": ["black", "red"]
    }
  ],

  "axes": [
    {
      "orient": "left",
      "scale": "y",
      "title": "Population",
      "titlePadding": 10,
      "grid": true
    },
    {
      "orient": "bottom",
      "scale": "x",
      "format": "d",
      "title": "Year",
      "tickCount": 15
    }
  ]
}
```

```

    }
  ],

  "marks": [
    {
      "type": "rect",
      "from": {"data": "annotation"},
      "encode": {
        "enter": {
          "x": {"scale": "x", "field": "start"},
          "x2": {"scale": "x", "field": "end"},
          "y": {"value": 0},
          "y2": {"signal": "height"},
          "fill": {"scale": "color", "field": "text"},
          "opacity": {"value": 0.2}
        }
      }
    },
    {
      "type": "line",
      "from": {"data": "table"},
      "encode": {
        "enter": {
          "interpolate": {"value": "monotone"},
          "x": {"scale": "x", "field": "year"},
          "y": {"scale": "y", "field": "population"},
          "stroke": {"value": "steelblue"},
          "strokeWidth": {"value": 3}
        }
      }
    },
    {
      "type": "symbol",
      "from": {"data": "table"},
      "encode": {
        "enter": {
          "x": {"scale": "x", "field": "year"},
          "y": {"scale": "y", "field": "population"},
          "stroke": {"value": "steelblue"},
          "strokeWidth": {"value": 1.5},
          "fill": {"value": "white"},
          "size": {"value": 30}
        }
      }
    }
  ]
}

```

```

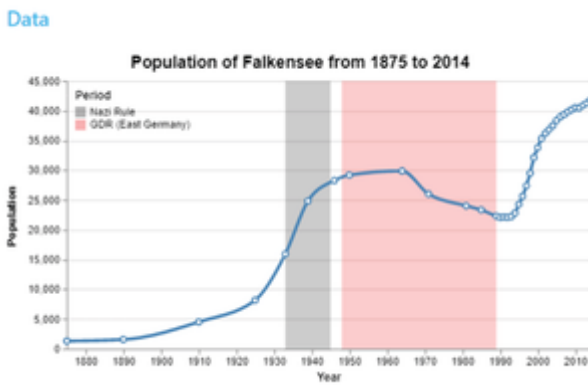
    }
  }
],
"legends": [
  {
    "fill": "color",
    "title": "Period",
    "orient": "top-left",
    "offset": 8,
    "encode": {
      "symbols": {
        "update": {
          "strokeWidth": {"value": 0},
          "shape": {"value": "square"},
          "opacity": {"value": 0.3}
        }
      }
    }
  }
]
}
...

```

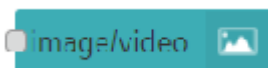
3. Save and execute the flow.

Output in the dashboard window

Open Visual Flow Creator dashboard to view the output.



Image/Video



The Image/Video node allows you to add an image/video to the dashboard. You can select the image/video source file from the “Select file” dialog or define using message property path.

Example

To execute the example flow using image/video node, follow these steps:

1. Create the flow as shown below:



2. Edit function node properties:

- Message property path (FUN):

```
msg.path = '/api/iotfile/v3/files/78ae02303cd448b997a08cdfd83e00d4/ForBiggerFun.mp4';  
return msg;
```

- Message property path (Blazes):

```
msg.path = '/api/iotfile/v3/files/78ae02303cd448b997a08cdfd83e00d4/ForBiggerBlazes.mp4';  
return msg;
```

3. Edit image/video node properties:



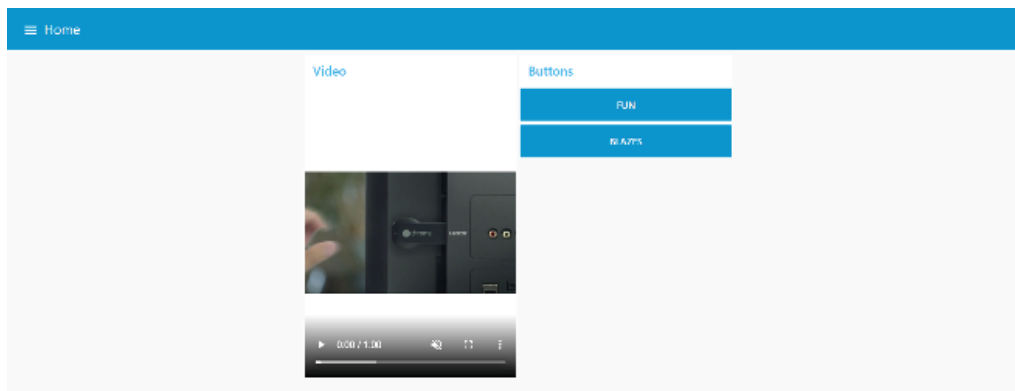
If the "Video" type is selected:

- Controls: It allows you to switch "On" or "Off" the controls of the audio file on the VFC dashboard. If you switch to "Off", the controls will be not visible on VFC dashboard.
- Autoplay: It allows you to switch "On" or "Off" the autoplay of the video file on the VFC dashboard.

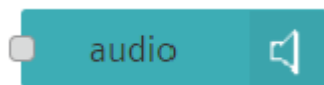
4. Save and execute the flow.

Output in the dashboard window

Open Visual Flow Creator dashboard to view the output. Click the "FUN" or "BLAZES" buttons to play the videos from the source.



Audio

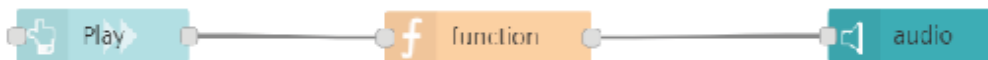


The Audio node allows you to add an audio player to the dashboard. You can select the audio source file from the “Select file” dialog or define using message property path.

Example

To execute the example flow using audio node, follow these steps:

1. Create the flow as shown below:



2. Edit function node properties:

- Message property path (PLAY):


```
msg.path = '/api/iotfile/v3/files/ebdeb4028b664e2baeab95a546f8b
84f/file_example_MP3_1MG.mp3';
return msg;
```



3. Edit audio node properties:


Edit audio node


Delete
Cancel
Done


▼ **node properties**


 Name


 Group ▼ 


 Size


 Label

 Do not cache audio:

 Controls ▼

 Autoplay ▼

 Path

[Open Fleet Manager](#) 

> **port labels**

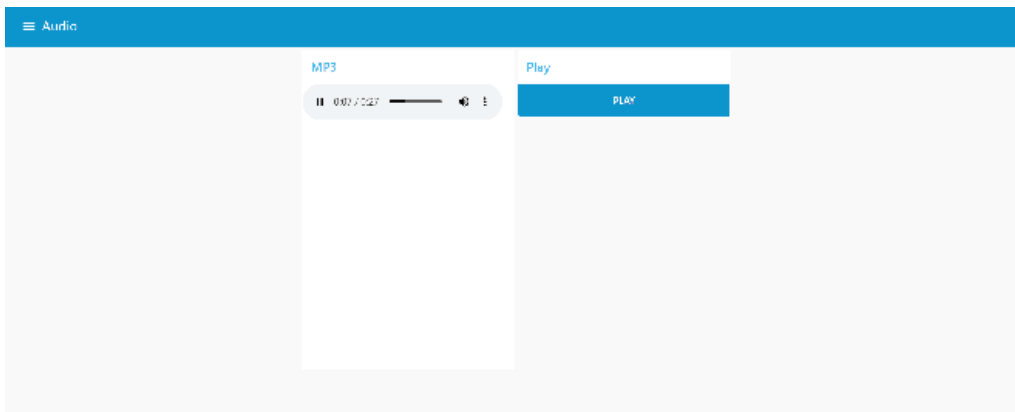


- **Controls:** It allows you to switch "On" or "Off" the controls of the audio file on the VFC dashboard. If you switch to "Off", the controls will be not visible on VFC dashboard.
- **Autoplay:** It allows you to switch "On" or "Off" the autoplay of the audio file on the VFC dashboard.

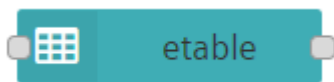
4. Save and execute the flow.

Output in the dashboard window

Open Visual Flow Creator dashboard to view the output. Click the "PLAY" button to play the audio from the source.



etable

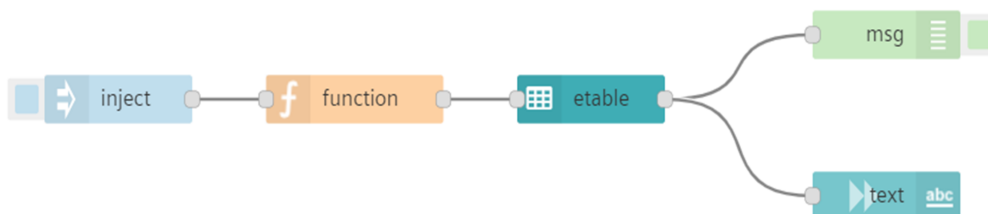


The etable node allows you to display the data in an interactive table format on Visual Flow Creator dashboard. This etable node displays the table data based on [Tabulator library](#).

Example

To execute the example flow using etable node, follow these steps:

1. Create the flow as shown below:



2. Edit function node properties:

In "function" node, enter the below JavaScript code in the "Code" field.

◦ Code:

```
var machine1 = 'Generator';  
var machine2 = 'Pump';  
  
msg.payload = [  
  {"AssetType":machine1, progress:95, dom:"14/01/2022", rating:4},  
  {"AssetType":machine2, progress:90, dom:"15/02/2022", rating:3}];  
return msg;
```

3. Edit etable node properties:

Edit etable node

Delete Cancel Done

node properties

Group test [Home]

Size 20 x 7

Name Name

Options { "movableColumns":true,"layout":"fitColour ...

PayloadColumns [{"title":"PushMe","field":"PushMe","width' ...

port labels

In "etable" node, enter the below json code in the "Options" field.

o Options:

```
{
  "movableColumns": true,
  "movableRows": true,
  "layout": "fitColumns",
  "pagination": "local",
  "height": "700px"
}
```

In "etable" node, enter the below json code in the "PayloadColumns" field.

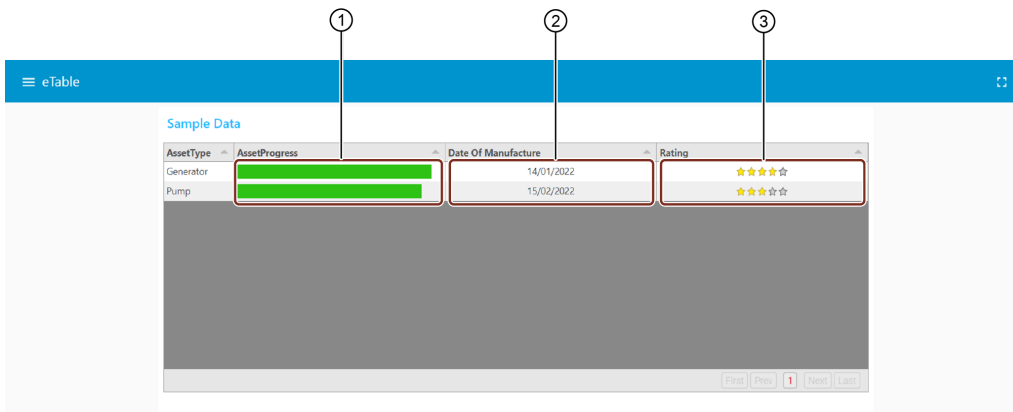
o PayloadColumns:

```
[
  {
    "title": "AssetType",
    "field": "AssetType",
    "width": "10%",
    "editor": false
  },
  {
    "title": "AssetProgress",
    "field": "progress",
    "formatter": "progress",
    "sorter": "number"
  },
  {
    "title": "Date Of Manufacture",
    "field": "dom",
    "sorter": "date",
    "hozAlign": "center",
    "width": "30%",
    "editor": false
  },
  {
    "title": "Rating",
    "field": "rating",
    "formatter": "star",
    "hozAlign": "center",
    "width": "30%",
    "editor": true
  }
]
```

4. Save and execute the flow.

Output in the dashboard window

Open Visual Flow Creator dashboard to view the output.



- ① Allows editing or updating the asset progress
- ② Allows editing or updating the manufacturing date of the asset
- ③ Allows editing or updating the rating of the Asset

You can edit or update the etable data on Visual Flow Creator dashboard directly.

7.3 Dashboard layouts

Tabs and groups

To configure the dashboard nodes, select the node to drag to the working area and double click the respective node to configure.

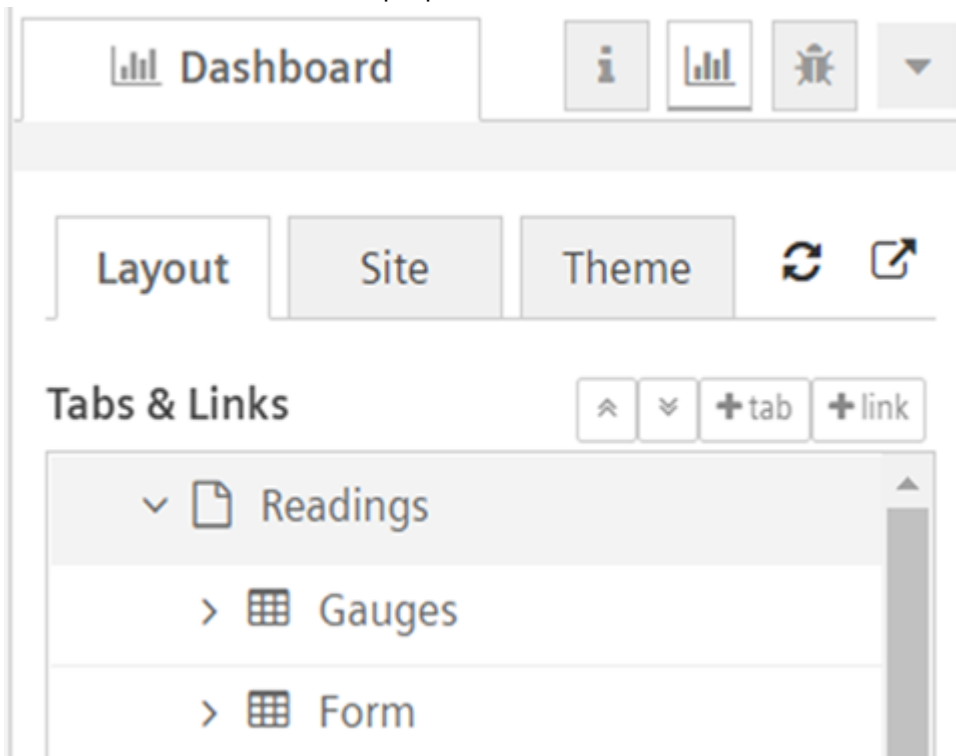
The tabs in the sidebar help you to re-order the groups and widgets. Also, you can add and edit their properties as per requirements. The detailed explanation of dashboard nodes configuration is given below.

Node properties

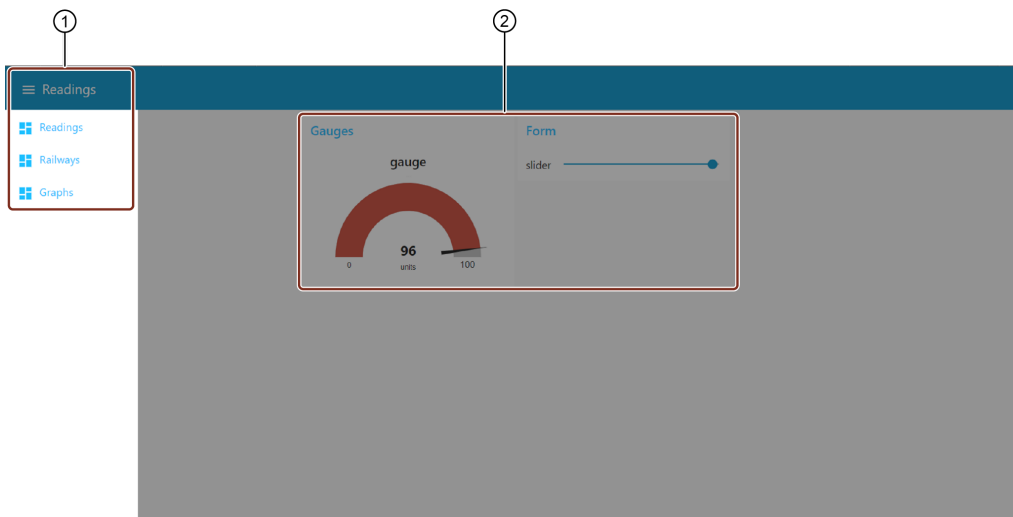
Some of the important node properties are described below:

Tab: Tab is created under dashboard tab. You can add and edit groups and edit the group properties under the tab. You can add or edit multiple groups in the tab. The following user interface shows all the node properties of VFC dashboard related node:

1. interface shows all the node properties of VFC dashboard related node:

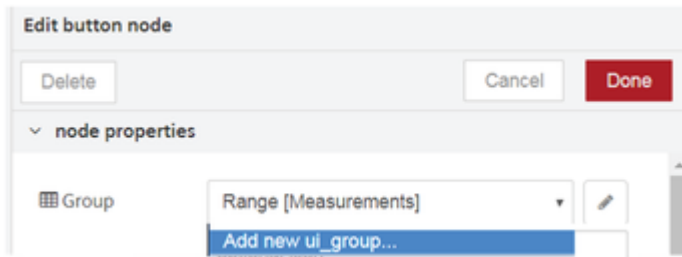


- In "Readings" tab, you can add one or more "Groups" like "Gauges" and "Form".
- Each "Group" can be configured with multiple dashboard nodes.
- Tabs provides the optional navigation level, as shown below:

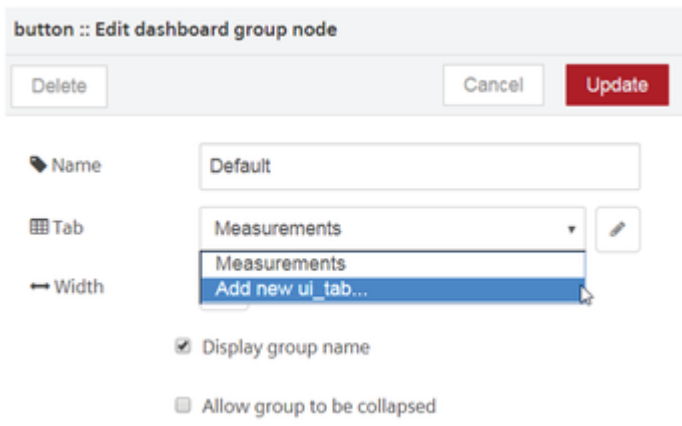


- ① Tabs on VFC dashboard
- ② Groups on VFC dashboard
 - Groups are displayed next to each other (in case more than 1 group is configured)

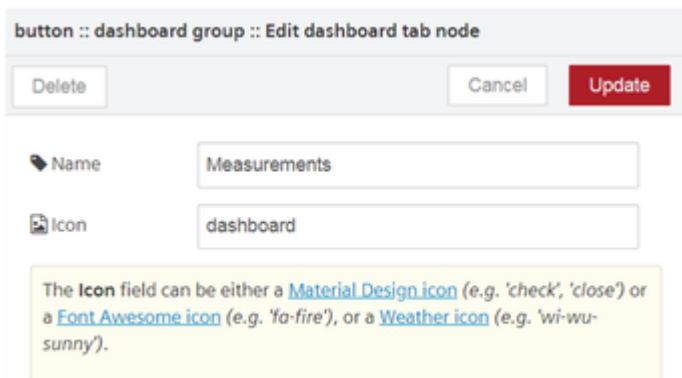
Group: A group is a container for multiple dashboard elements. To create a new group,
 2. select "Add new ui_group" from the shown drop-down menu.



Name the UI group. Thereafter configure the tab by selecting "Add new ui_tab" from the drop-down menu.

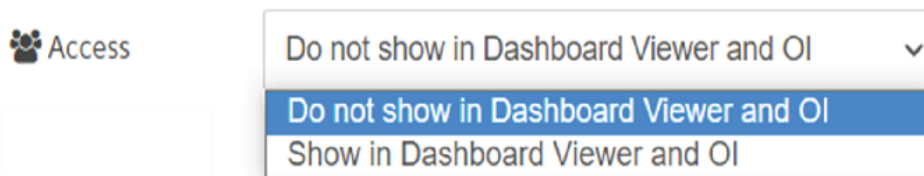


Configure the UI tab width and name the UI tab.



The tab and group of the node will be now configured. You can also create groups in the sidebar. For more information, refer to [Create groups in sidebar](#).

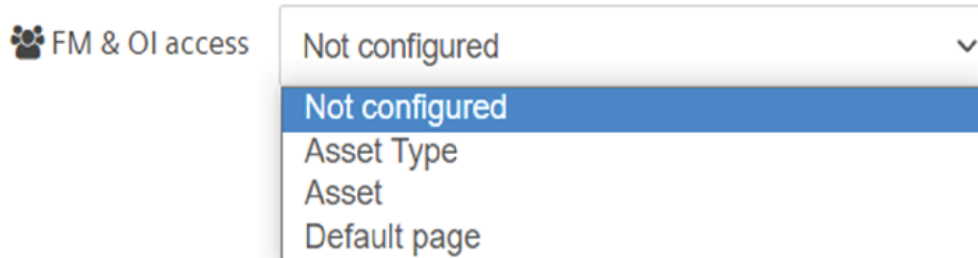
Access: Choose the option "Show in Dashboard Viewer and OI" or "Do not show in Dashboard Viewer and OI":
 3. Dashboard Viewer and OI":



- Show in Dashboard Viewer and OI: The dashboard will be displayed in "Dashboard Viewer" and "Insights Hub Monitor".

- Do not show in Dashboard Viewer and OI: The dashboard will not be displayed in "Dashboard Viewer" and "Insights Hub Monitor".

OI access: It is allowed to display the dashboard for an asset or asset type in "Insights Hub Monitor". For more information, refer to [Dashboards](#) chapter in Insights Hub Monitor.



4.



The "Default page" will display the dashboard on the selected tab within the tenant.

5. **Size:** The grid layout defines the height and width of the widgets as the size of the group. By default, 'auto' is defined as the group size, but you can always set the size to a fixed number of units according to requirements. To understand the size configuration refer to [Example on size configuration](#).

6. **Type:** This is an optional property wherein you can select the working type to design a node.

7. **Icon:** The node's icon is defined by its property.

8. **Label/ Name:** This is an optional property. The label can be defined with a new name which will be displayed as the name of the node to the user.

9. **Value format:** Define data format if necessary.

10. **Units:** Define a unit if applicable.

11. **Color:** This is an optional property. You can set the icon color that will be displayed in the output.

12. **Background:** This is an optional property. You can set the background color as per requirements.

13. **Range:** This is an optional property. You can define a range by setting minimum and maximum values.

14. **Topic:** The topic field can be used to set the msg.topic property that is output.

15. **Check boxes:**

- If message arrives on input, pass through to output: Selecting this will set the value to true. You can refer to [Example on message input passing to output](#) for a clear understanding.
- Only one browser: If this is selected for a node in a flow, the node will handle each of the dashboards separately. For each browser, the communications originating from the node will be exclusively handled.

If "Only one browser" is not selected, the node will share all the dashboard communication with every open browser from that user. For more information, refer to [Example on usage of one and multiple browsers](#).

Link and its node properties

The link in the dashboard tab helps you to add the links to the dashboard. Also, you can edit the properties to assign a link to display or access from the dashboard. Some of the important link node properties are described below:


1. **Name:** Enter the name of the link node.
2. **Link:** Enter the link to open from the dashboard.
3. **Icon:** Enter the name for the icon of the link node properties
4. **Open in:** Choose "New Tab" or "This tab" to open the link in the new or current browser.
5. **Show in the sidebar of dashboard opened from Dashboard Viewer:** It is a check box option, to access the link from the sidebar of the dashboard while opening from "Dashboard Viewer".

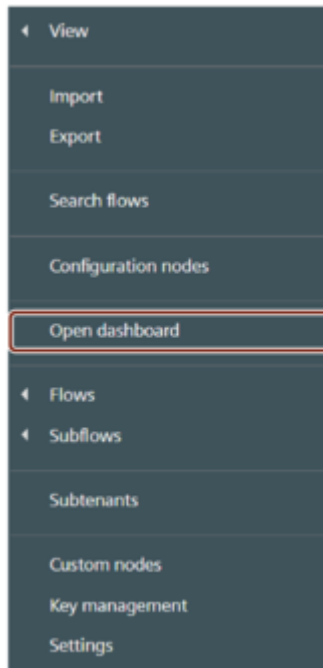


If you uncheck the box, the link from the sidebar of the dashboard while opening from "Dashboard Viewer" will not be visible. But it will be visible in the list of sidebar while accessing from Visual Flow Creator dashboard.

6. **Access:** Choose the option to "Show in Dashboard Viewer" or "Do not show in Dashboard Viewer".
7. **Icon:** Upload an image for the link node to display it in "Dashboard Viewer", if required.
8. **Description:** Enter the description for the link node, if required.

View dashboard nodes output

Once a flow is created, save it to deploy. Click the menu button  and then select "Open dashboard".



A separate browser tab will open where you will be able to view the dashboard.

You will be redirected to "<https://visualflowcreatorURL.yyy.mindsphere.io/ui/dash/?user=username@email.com#/0>".

You can open the dashboard of another user in his environment by defining user=[username@email.com](#).

The tab menu will be visible on the left of the dashboard UI.



Select the configured tab to view your dashboard screen.



- You can view the dashboards of other users in the tenant.
- The user's dashboards are automatically available for another tenant's user.

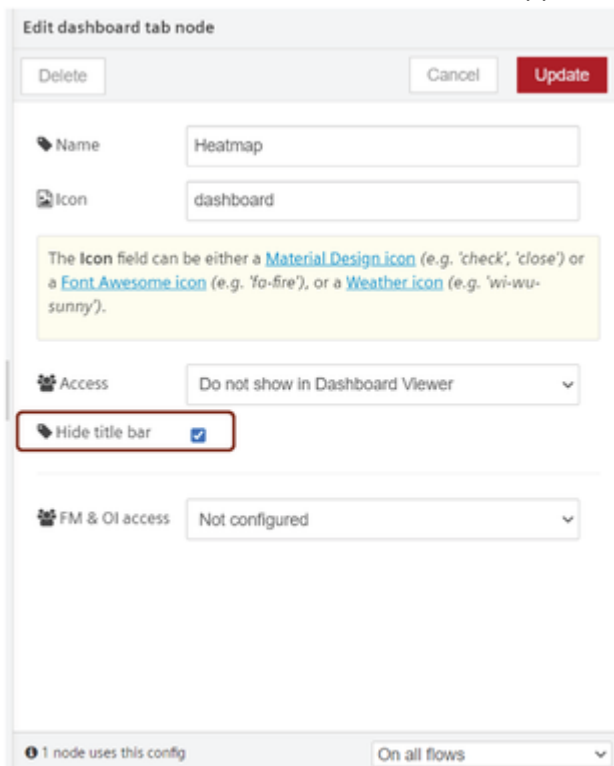
Hide dashboard application title bar

You can hide the application title bar to increase dashboard view.



- ① Dashboard tab menu
- ② Application title bar

You can select "Hide title bar" to hide the application title bar in dashboard tab node properties.



In this case, dashboard tab menu is not available to change the tabs.

Other methods to configure dashboard nodes

You can configure the layout of the dashboard by using the "dashboard" view on the sidebar. To learn more about configuring tabs and groups using sidebar, refer to [Configuration of dashboard nodes in sidebar](#).

7.4 Using standard dashboard nodes

Example scenario

A linear gauge needs to display a specific data point, using a pointer. The pointer is required to move on a colored scale to indicate whether the monitored data is within the defined limits.

Objective

The gauge scale gives values between predefined minimum and maximum values. Within that scale, you can create various ranges to classify the data.

Requirements

- Standard dashboard gauge node
- Standard dashboard slider node

Procedure

Select the slider node from the standard dashboard nodes. Configure the group and range

1. of the node:

- Group name: Range [Measurements]
- Type: Gauge
- Label: Linear scale
- Range: Min 0, max 10
- Color gradient: Green, yellow, red
- Sectors: 0-3-7-10

Select the gauge node from the same dashboard group. Configure the group, range and

2. color gradient of the node:


- Group name: Range [Measurements]
- Label: Pointer
- Range: Min 0, max 10; Step 1

3. Connect the slider to the gauge.

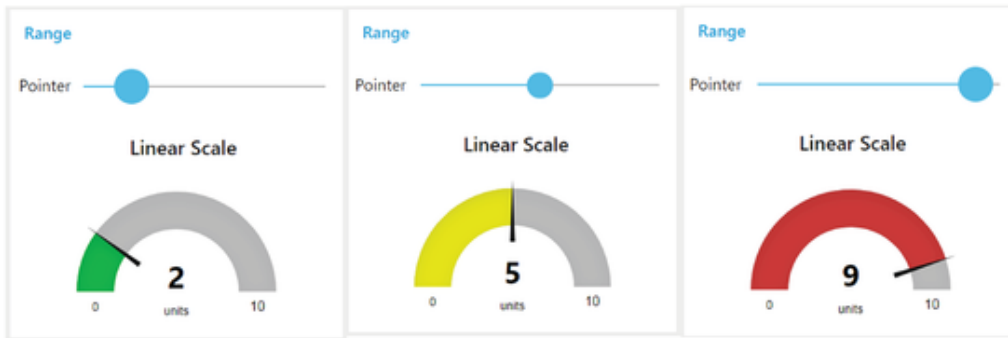
4. Save and deploy.



Result

To view the results, click the  icon and select "Open dashboard" from the menu. You will be redirected to "<https://visualflowcreatorURL.yyy.mindsphere.io/ui/dash/?user=username@email.com#/0>" in a new tab.

Select the configured group left navigation window of the dashboard UI. The required dashboard flow will now be visible on the dashboard screen. You can slide the pointer and get the desired results:



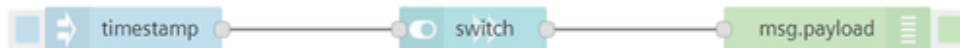
7.5 Application examples on node properties

Some examples have been described below to explain the dashboard node properties.

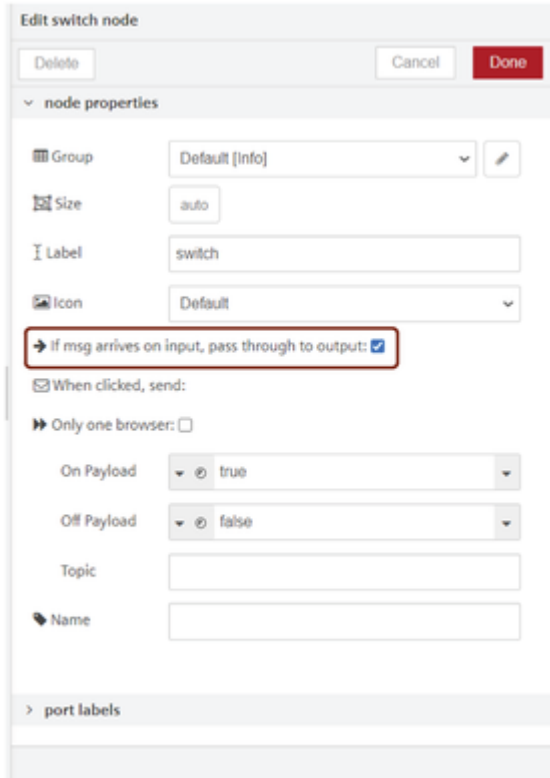
Example 1 - Message input passes to output

Let us consider a simple example considering the switch node.

1. Inject the timestamp node to the switch node and connect it to a msg.payload output.



2. Select the "If msg arrives on input, pass through to output:" check box.



Result

The "If msg arrives on input, pass through to output:" check box is selected: The output will be displayed in the debug tab depending on the toggling of the switch in the dashboard window browser.



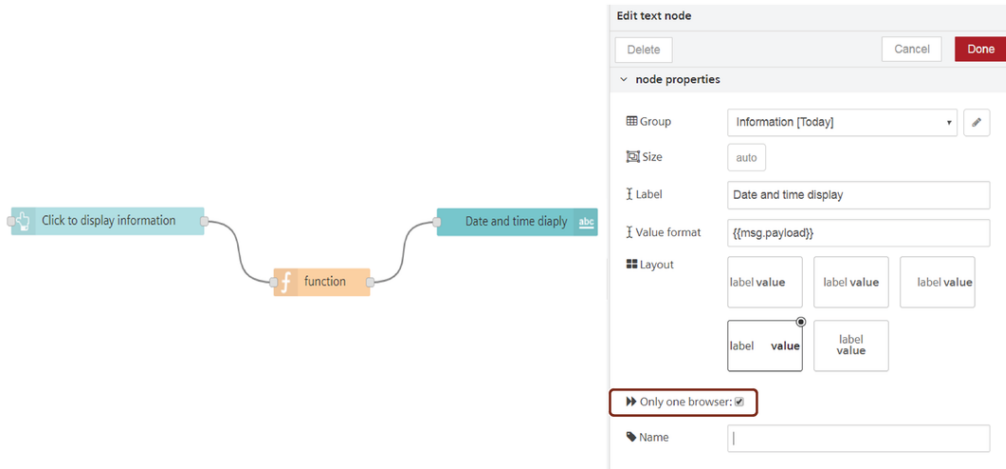
The "If msg arrives on input, pass through to output:" check box is not selected: The output will not be displayed in the debug tab when you toggle the switch in the dashboard window browser.



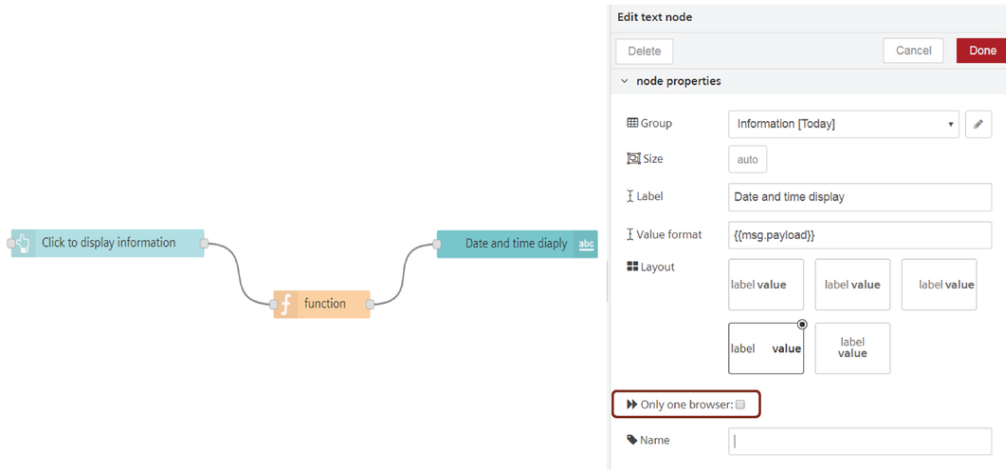
Example 2 - Flows on usage of a single and multiple browsers

Define a function which generates today's date. Given below are the examples which differentiates the single and multiple browser concept.

1. Select the "only one browser" check box of the last node (text node) from the given flow:



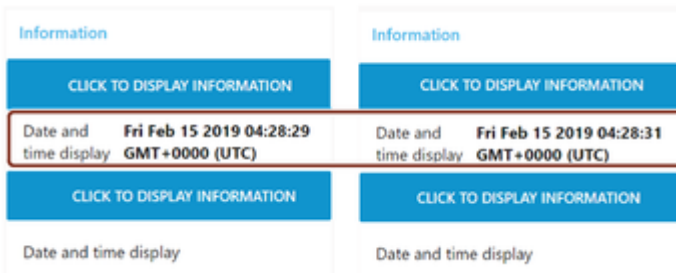
2. Clear the "only one browser" check box of the last node (text node) from the given flow:



3. Open the dashboard from the menu icon.
4. Open two dashboard browsers to compare the flows.

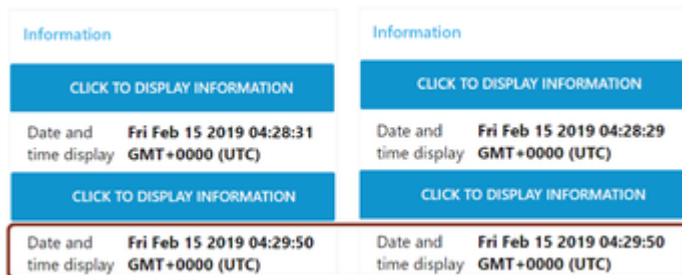
Result

The "Only one browser" check box is selected: When you click the button (click to display information) node in the first browser, you get the output only for the clicked respective browser. You have to click the button node of the other browser separately to get its output.



Note that the date and time display has seconds difference in time because the buttons in both the browsers were clicked individually.

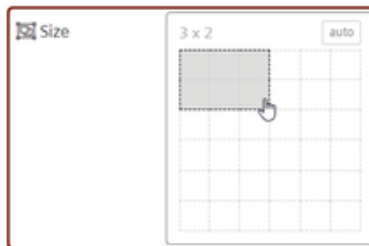
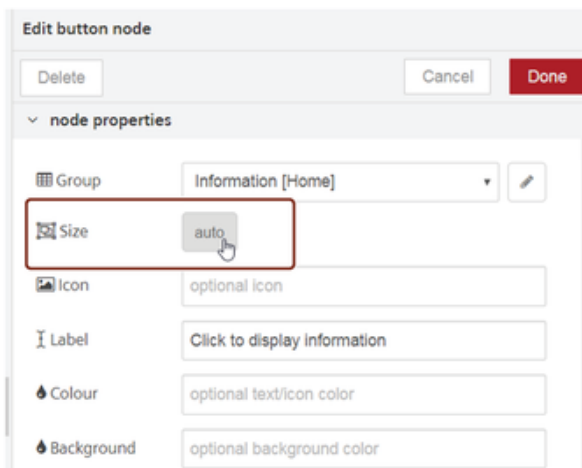
The "Only one browser" check box is not selected: When you click the button (click to display information) node in any browser, output of all the opened browsers will be displayed simultaneously.



Here, the date and time display has no time difference because only one button click in any of the browsers displayed the output at the same time.

Example 3 - Configure size

Considering the size of dashboard nodes in "Example 2", by default the button node size is set to "auto". To configure the button node to different size, open properties and select the size to different number, for example "3x2" as shown below:



Result

The size of the button will be changed and the button will be displayed with the size set in the properties.



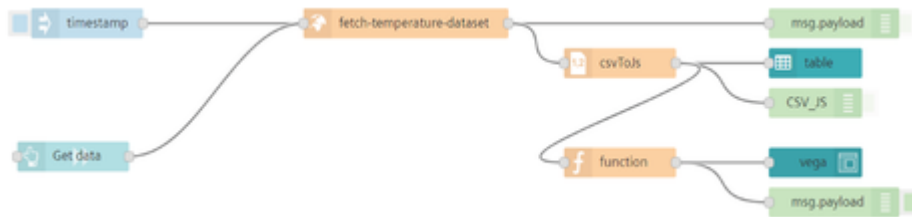
7.6 Example for using Vega dashboard node

In this example, you can read the weather report in the CSV format and convert to JSON and then the output is displayed on the dashboard using the Vega dashboard node.

Example

Let us consider an example to read CSV formatted data and convert to JSON and then display the output on the dashboard:

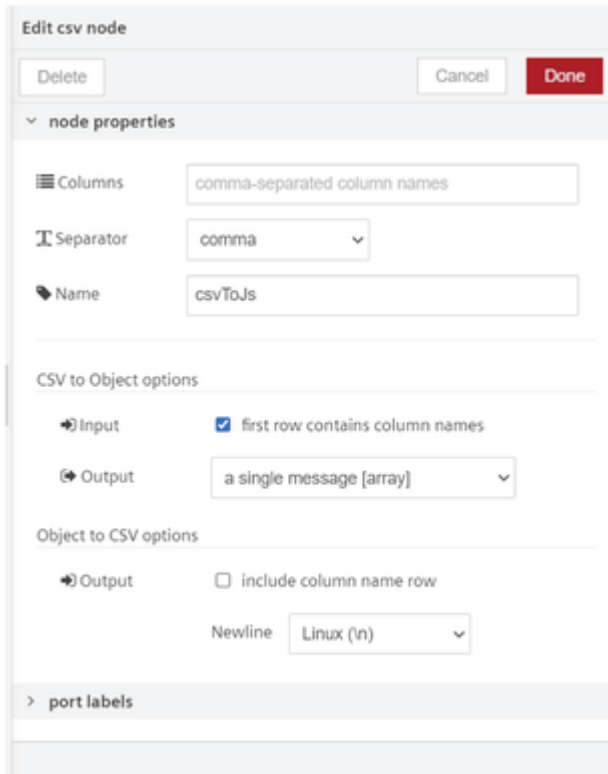
1. Create the flow as shown below:



2. Edit http request node properties.

- URL: "<https://cdn.jsdelivr.net/npm/vega-datasets@2.2.0/data/seattle-weather-hourly-normals.csv>"

3. Edit csv node properties.



4. Edit function node properties.

- Code:

```
??? Code ```javascript // store msg let mytmpdata = msg.payload; msg.payload = {  
"$schema": "https://vega.github.io/schema/vega/v5.json", "description": "A heatmap showing  
average daily temperatures in Seattle for each hour of the day.", "width": 800, "height": 500,  
"padding": 5, "title": { "text": "Seattle Annual Temperatures", "anchor": "middle", "fontSize": 16,  
"frame": "group", "offset": 4 },
```

```
"signals": [  
  {  
    "name": "palette", "value": "Viridis",  
    "bind": {  
      "input": "select",  
      "options": [  
        "Turbo",  
        "Viridis",  
        "Magma",  
        "Inferno",  
        "Plasma",  
        "Cividis",  
        "DarkBlue",  
        "DarkGold",  
        "DarkGreen",  
        "DarkMulti",  
        "DarkRed",  
        "LightGreyRed",  
        "LightGreyTeal",  
        "LightMulti",  
        "LightOrange",  
        "LightTealBlue",  
        "Blues",  
        "Browns",  
        "Greens",  
        "Greys",  
        "Oranges",  
        "Purples",  
        "Reds",  
        "TealBlues",  
        "Teals",  
        "WarmGreys",  
        "BlueOrange",  
        "BrownBlueGreen",  
        "PurpleGreen",  
        "PinkYellowGreen",  
        "PurpleOrange",  
        "RedBlue",  
        "RedGrey",  
        "RedYellowBlue",  
        "RedYellowGreen",  
        "BlueGreen",  
        "BluePurple",
```

```

        "GoldGreen",
        "GoldOrange",
        "GoldRed",
        "GreenBlue",
        "OrangeRed",
        "PurpleBlueGreen",
        "PurpleBlue",
        "PurpleRed",
        "RedPurple",
        "YellowGreenBlue",
        "YellowGreen",
        "YellowOrangeBrown",
        "YellowOrangeRed"
    ]
}
},
{
    "name": "reverse", "value": false, "bind": {"input": "checkbox"}
}
],

"data": [
    {
        "name": "temperature",
        "url": "data/seattle-weather-hourly-normals.csv",
        // "format": {"type": "csv", "parse": {"temperature": "number", "date": "date"}},
        "transform": [
            {"type": "formula", "as": "hour", "expr": "hours(datum.date)"},
            {"type": "formula", "as": "day", "expr": "datetime(year(datum.date), month(datum.date), date(datum.date))"}
        ]
    }
],

"scales": [
    {
        "name": "x",
        "type": "time",
        "domain": {"data": "temperature", "field": "day"},
    }
]

```

```

    "range": "width"
  },
  {
    "name": "y",
    "type": "band",
    "domain": [
      6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23,
      0, 1, 2, 3, 4, 5
    ],
    "range": "height"
  },
  {
    "name": "color",
    "type": "linear",
    "range": {"scheme": {"signal": "palette"}},
    "domain": {"data": "temperature", "field": "temperatur
e"},
    "reverse": {"signal": "reverse"},
    "zero": false, "nice": true
  }
],

"axes": [
  {"orient": "bottom", "scale": "x", "domain": false, "titl
e": "Month", "format": "%b"},
  {
    "orient": "left", "scale": "y", "domain": false, "titl
e": "Hour",
    "encode": {
      "labels": {
        "update": {
          "text": {"signal": "datum.value === 0 ? 'Mi
dnight' : datum.value === 12 ? 'Noon' : datum.value < 12 ? datum.value
+ ':00 am' : (datum.value - 12) + ':00 pm'"}
        }
      }
    }
  }
],

"legends": [
  {

```

```

        "fill": "color",
        "type": "gradient",
        "title": "Avg. Temp (°C)",
        "titleFontSize": 12,
        "titlePadding": 4,
        "gradientLength": {"signal": "height - 16"}
      }
    ],

    "marks": [
      {
        "type": "rect",
        "from": {"data": "temperature"},
        "encode": {
          "enter": {
            "x": {"scale": "x", "field": "day"},
            "y": {"scale": "y", "field": "hour"},
            "width": {"value": 5},
            "height": {"scale": "y", "band": 1},
            "tooltip": {"signal": "timeFormat(datum.date, '%b %d
            %I:00 %p') + ': ' + datum.temperature + '°'"}
          },
          "update": {
            "fill": {"scale": "color", "field": "temperature"}
          }
        }
      }
    ]
  };

```

```

const jsonString = JSON.stringify(mytmpdata);
console.log("-----");
console.log(jsonString);
console.log("===-----");
const myobj = JSON.parse(jsonString);

```

```

//ok const xdata = [{"date":"2010-12-30T11:00:00","pressure":1018.
3,"temperature":4.9,"wind":3.8},{ "date":"2010-12-30T12:00:00","pressur
e":1017.9,"temperature":5.6,"wind":3.8},{ "date":"2010-12-30T13:00:0
0","pressure":1017.5,"temperature":5.9,"wind":4},{ "date":"2010-12-30T1
4:00:00","pressure":1017.2,"temperature":6.2,"wind":4.1}];
//ok msg.payload.data[0].values = xdata;

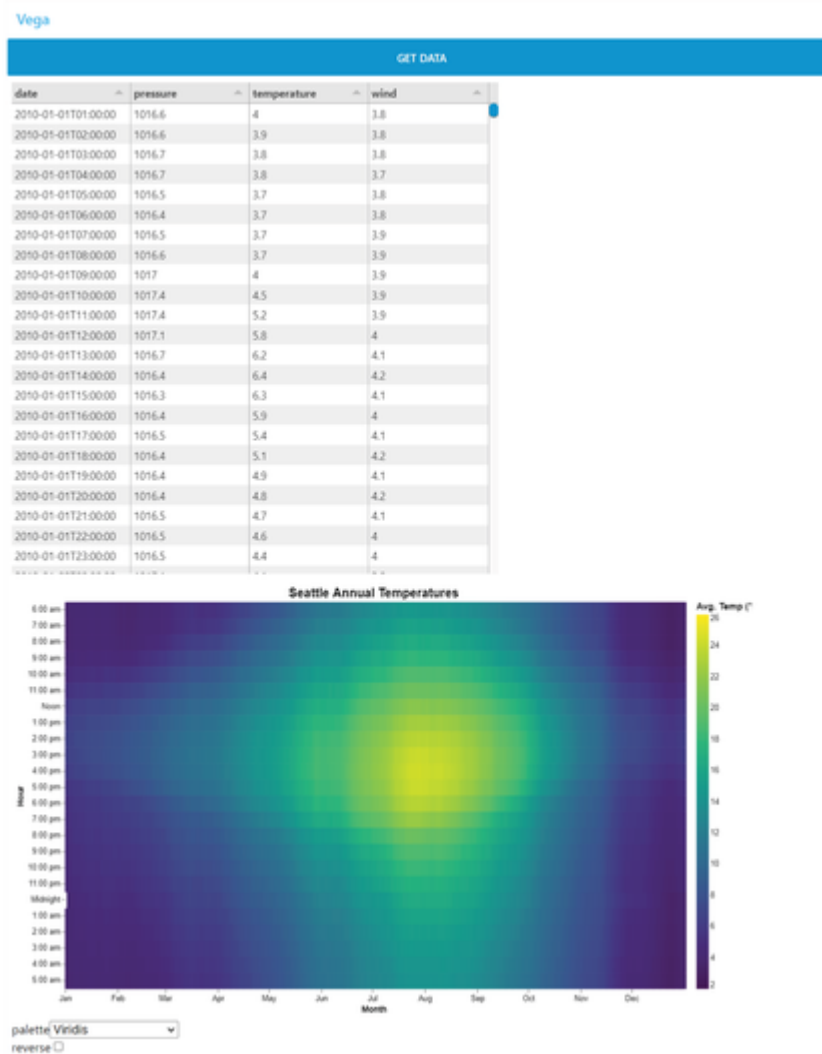
```

```
msg.payload.data[0].values = myobj;  
console.log("typecheck "+ typeof xdata);  
return msg;  
...  
...  
...
```

5. Save and execute the flow.

Result

Open Visual Flow Creator dashboard to view the output.

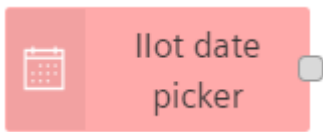


7.7 Usage of IIoT Dashboard nodes

The IIoT dashboard nodes allow you to build, create and interchange IIoT dashboard nodes and default dashboard nodes in one single dashboard.

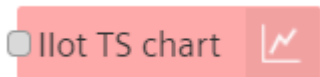
For more information, refer to [Example to use IIoT Dashboard nodes](#).

IIoT date picker



The node enables you to select a date range with a start date and an end date.

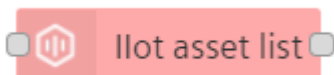
IIoT TS chart



The node aggregate values of a certain aspect for a defined date range. The display will be a line chart UI in the dashboard.

| Variables | Description |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Show none if msg.topic has no variables | If this option is selected, no aggregate variable values of an aspect will be displayed in the dashboard.
Note: Refresh the dashboard page to view the result. |
| Show all if msg.topic has no variables | If this option is selected, all aggregate variable values of an aspect will be displayed in the dashboard.
Note: Refresh the dashboard page to view the result. |

IIoT asset list

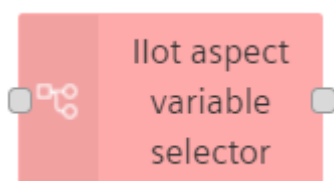


The asset list can be initialized by sending a payload with a JSON object containing the assetId that should be selected. If selected asset has changed the following message will be sent:

```
{assetId: 'asset id', name: 'name of asset'}
```

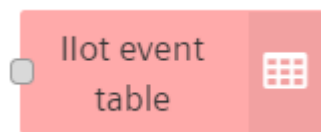
Example: {assetId: '12345', name: '2JX5D'}

IIoT variable selector



The node allows you to select an aspect variable from the asset list provided. It sends a topic property defined both in msg and msg.payload objects.

IIoT event table



The node allows to add IIoT Event Table to the VFC Dashboard. The node provides events which have been emitted within a date range. IIoT Event Table needs to be configured either with Asset, Date range (From and To) properties using configuration page or receiving a message with "From" and "To" properties. You can customize the event table columns from "Show columns" configuration and you can add custom event table fields using "Custom event type ID" and "Custom fields".

Delete
Cancel Done

▼ node properties

display [Button] ▼

auto

event table

Asset ...

▶▶ Only one browser:

Severity
 EntityId
 Timestamp
 Description
 Source
 Acknowledged
 CorrelationId
 TypeId
 Code

+ Add custom field

| Field | Type |
|-----------------------|------|
| No fields defined yet | |

> port labels

| Properties | Description |
|------------|-------------------------------------|
| Asset | Select the asset to list the files. |

| Properties | Description |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| From | Select the start date. |
| To | Select the end date. |
| Only one browser | Select for a node in a flow, the node will handle each of the dashboards separately. For each browser, the communications originating from the node will be exclusively handled. If "Only one browser" is not selected, the node will share all the dashboard communication with every open browser from that user. Refer to Example on usage of one and multiple browsers for better understanding. |
| Name | Select the asset to list the files. |
| Show columns | Select the properties to customize the event table columns. |
| Custom event type ID | Enter the custom event ID to display the event properties. |
| Custom fields | Add the custom fields for the custom event type. |

IIoT map



The node allows you to configure IIoT map for an asset and displays a geographic map based on OpenStreetMap. It needs to be configured with Latitude, Longitude and Zoom level by using node properties.

7.8 Using IIoT Dashboard nodes

Example scenario

An event needs to be organized in Berlin. The event is a demo show on turbo engines of Siemens. It is required to design an event flow to integrate to the event management application.

Objective

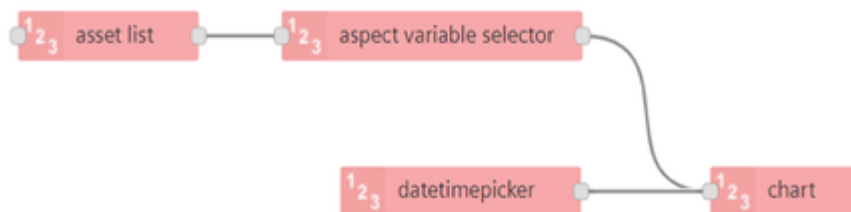
To display the data of an asset for a date range.

Requirements

- A timestamp input
- IIoT date picker node
- IIoT asset list node
- IIoT aspect selector node
- IIoT TS chart node
- Message payload

Procedure

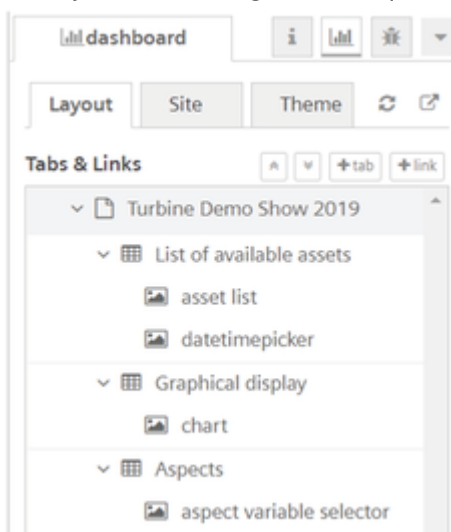
1. Design the event flow as shown below:



2. Configure the tabs in groups:

- IIoT date picker node
- IIoT asset list node
- IIoT aspect selector node
- IIoT TS chart node
- IIoT map To understand group and tabs configuration, refer to [Dashboard working layouts](#).


Swap, shift and configure groups and tabs from the sidebar "Dashboard" window under 3. "Layout" according to the requirements.



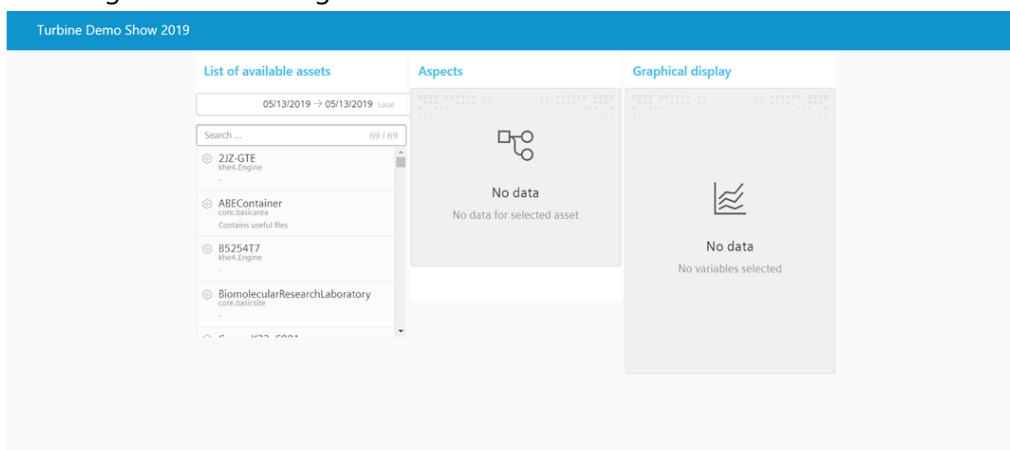
For more information, refer to [Dashboard layouts](#).

4. Save the flow.

Result 1

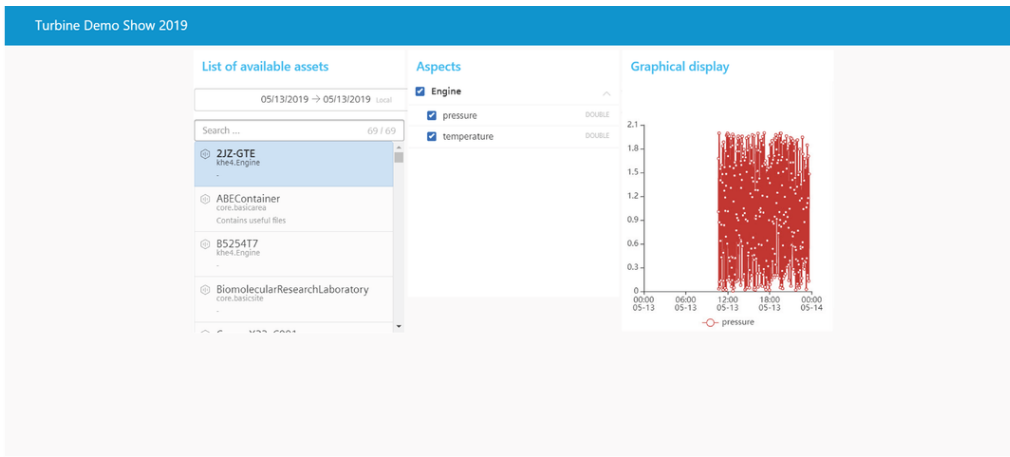
Click  in Visual Flow Creator application, and select "Open dashboard" to view the flow in Visual Flow Creator dashboard.

You will get a flow as designed:



Select an asset, an aspect/(s) and the time range. You will be the data analysis in the chart and the asset location:

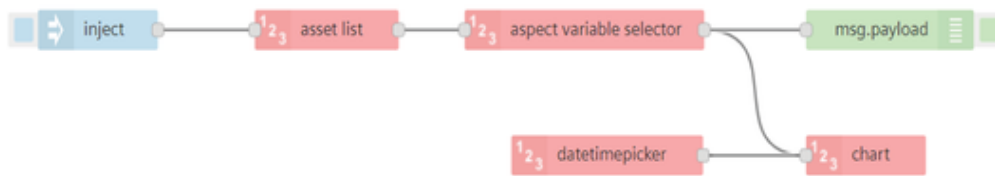
7.9 Implement IIoT Map node - Example



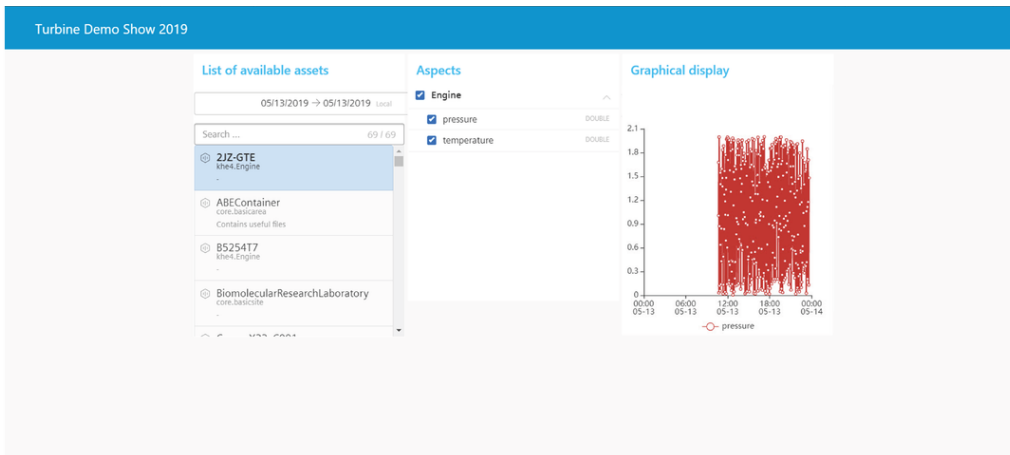
Result 2

Connect a time stamp and a message payload as shown below with the selected asset, aspect and date range as shown in Result 1:

Node flow design:



Flow in dashboard:



The required information will be displayed in the message payload:

```

6/29/2020, 3:19:34.854 PM node: 66272f25.78a79
msg: Object
  {
    topic:
      --
  }

6/29/2020, 3:19:35.443 PM node: 66272f25.78a79
ebdeb4028b664e2baeab95a546f8b84f/Engine/pressure_tem...
msg: Object
  {
    topic:
      "ebdeb4028b664e2baeab95a546f8b8..."
  }
    
```

7.9 Implement IIoT Map node - Example

Example scenario

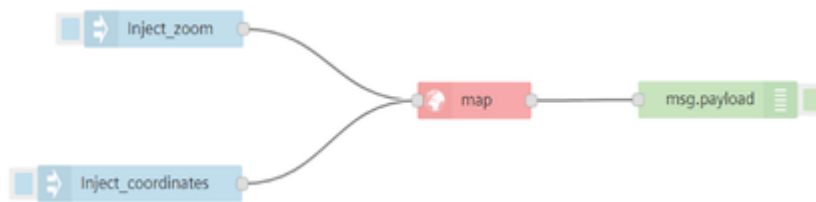
Create a flow with Map node.

Objective

To view the map in the Visual Flow Creator Dashboard with given coordinates.

Procedure

1. Design the event flow as shown below:



2. Configure the group in tab:

- IIoT map node

3. Edit IIoT map node properties:

- Group: Location [IIoT Map]
- Label: map
- Longitude: 8.352164
- Latitude: 49.027324
- Zoom level: 15

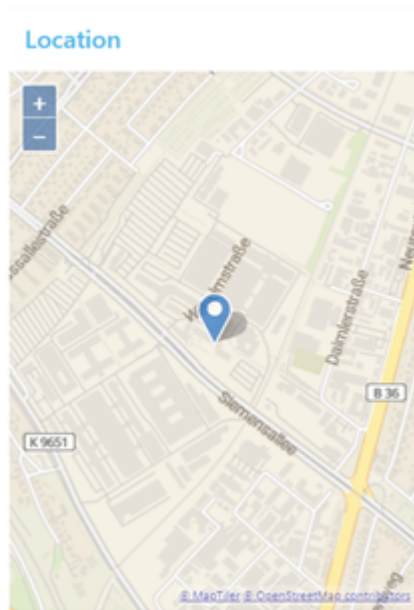


- To drag the map in the dashboard, check "Draggable".
- To set the coordinates to current location, click "Use current location".

4. Save the flow.

Result 1

Open the dashboard to display the result as per the coordinates set in IIoT map node:



Change zoom level using inject node

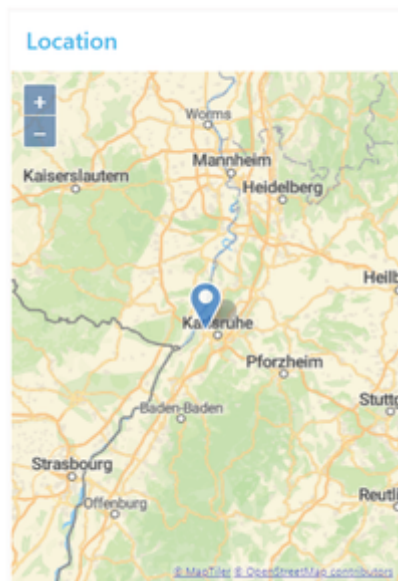
Edit the "Inject_zoom" node property and enter the below JSON code to change the zoom level:

```
{  
  
  "zoomLevel": 8  
  
}
```

Save the flow.

Result 2

Open the dashboard to display the result as per the zoom level set in Inject_zoom node:



Change zoom level and coordinates using inject node

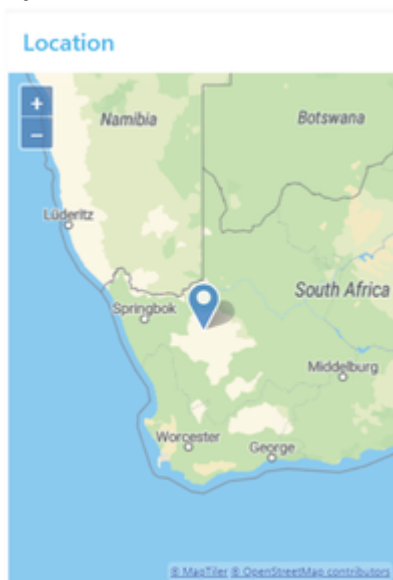
Edit the "Inject_coordinates" node property and enter the below JSON code to change the zoom level and coordinates:

```
{
  "latitude": -30,
  "longitude": 20,
  "zoomLevel": 5
}
```

Save the flow.

Result 3

Open the dashboard to display the result as per the zoom level and coordinates set in Inject_coordinates node:



7.10 Integrating Insights Hub Monitor plugin - Example

Example scenario

Access Visual Flow Creator dashboard by using Insights Hub Monitor plugin.

Objective

To display Visual Flow Creator dashboard in Insights Hub Monitor by using a plugin.

Procedure


1. Design the event flow as shown below:



2. Double click to edit "slider" and "gauge" node properties.

Swap, shift and configure groups and tabs from the sidebar "Dashboard" window under

3. "Layout" according to the requirements.

4. Click  to configure Insights Hub Monitor plugin:

Edit dashboard tab node

Delete Cancel Update

Name: FM tab

Icon: dashboard

The **Icon** field can be either a [Material Design icon](#) (e.g. 'check', 'close') or a [Font Awesome icon](#) (e.g. 'fa-fire'), or a [Weather icon](#) (e.g. 'wi-wu-sunny').

Access: Do not show in Dashboard Viewer

FM access: Asset

Access type: Show for all users

Asset: FA20 (af1a2af7695b40b2bc36955b179b5x)

1 node uses this config On all flows

- FM access: Asset
 - Access type: Show for all users
 - Asset: FA20 asset selected
5. Click "Update".
 6. Save the flow.
 7. Open Insights Hub Monitor.

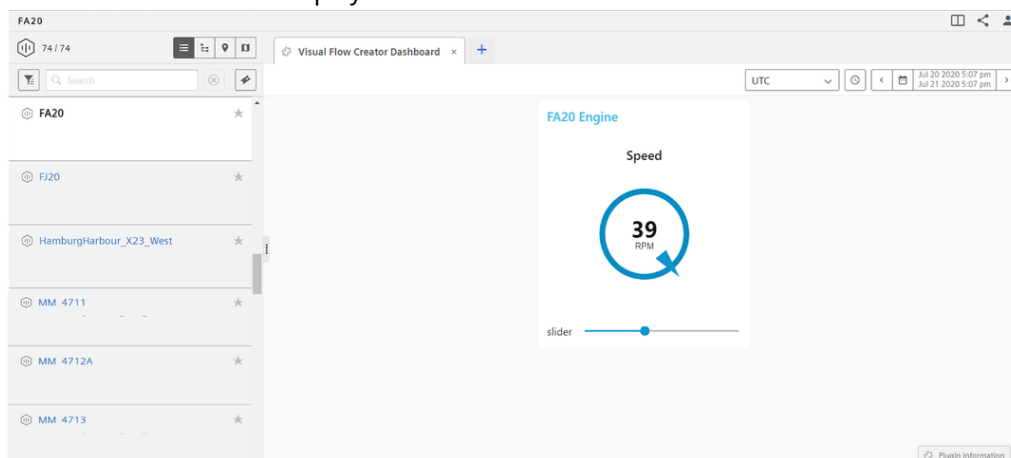
8. Select Visual Flow Creator tile.



9. Select FA20 asset.

Result

The dashboard will be displayed in Visual Flow Creator tab.



7.11 Usage of Dashboard Viewer

The "Dashboard Viewer" is an application on your Launchpad and can be used to access dashboards created by users in your environment. You can access the "Dashboard Viewer" with role environment user. To make dashboard accessible for other users of your environment, use the VFC to publish them.

Dashboard Viewer screen

The following screenshot shows the "Dashboard Viewer" in the Launchpad:



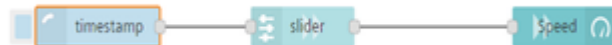
- ① Launchpad
- ② Dashboard Viewer

Procedure

You can publish the dashboard in the "Dashboard Viewer" through Visual Flow Creator. Consider a simple example to measure a conveyor belt speed.

Drag and drop the timestamp node, slider node and compass node from the dashboard nodes.

1. Interconnect the nodes:



Select the slider node from the standard dashboard nodes. Configure the group and range

2. of the node:

- Group name: Conveyor belt [Home]
- Label: slider
- Range: Min 0, max 100


Select the gauge node from the same dashboard group. Configure the group, label, units

3. and range of the node:

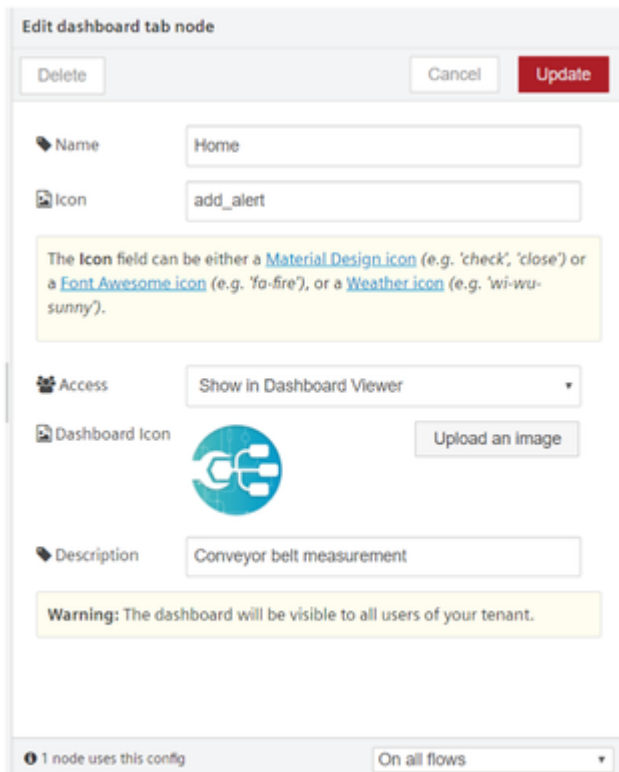
- Group name: Conveyor belt [Home]
- Type: Compass
- Label: speed

- Units: rpm
- Range: Min 0, max 100

4. In the dashboard tab, click  to create in the dashboard window.

Hover the mouse on the created tab, click .

5. Edit dashboard tab node screen appears:



- Name: Enter the dashboard name
- Icon: Select a textual key indexing from the standard catalogs
- Access: Select the "Show in Dashboard Viewer" to publish dashboard
- Dashboard Icon: Upload an image for the dashboard



Image size should be max 10Kb, so use SVG format for better results.

- Description: Enter the description about the dashboard

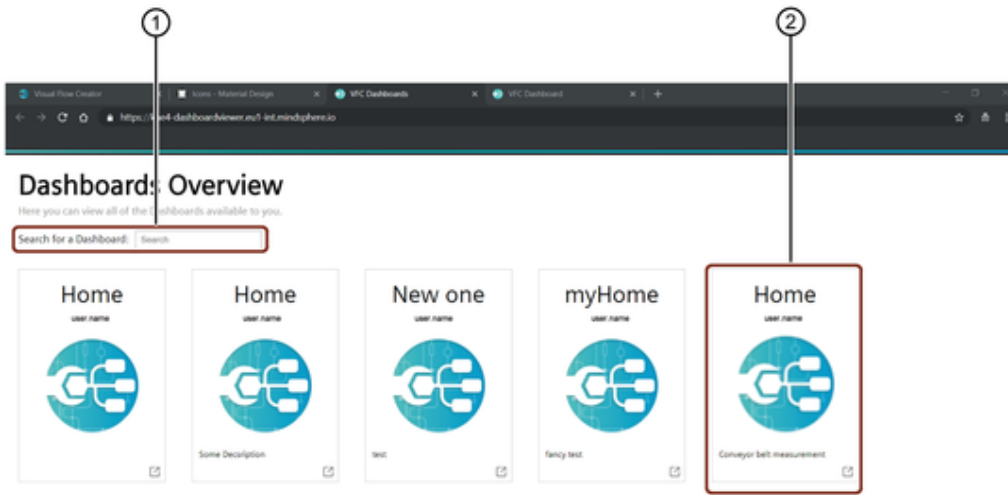
6. Click "Update".

7. Save and deploy.

Result

To view the results, click the "Dashboard Viewer" icon in the Launchpad. "Dashboard Viewer" screen is displayed.

7.12 Adding links to Dashboard Viewer

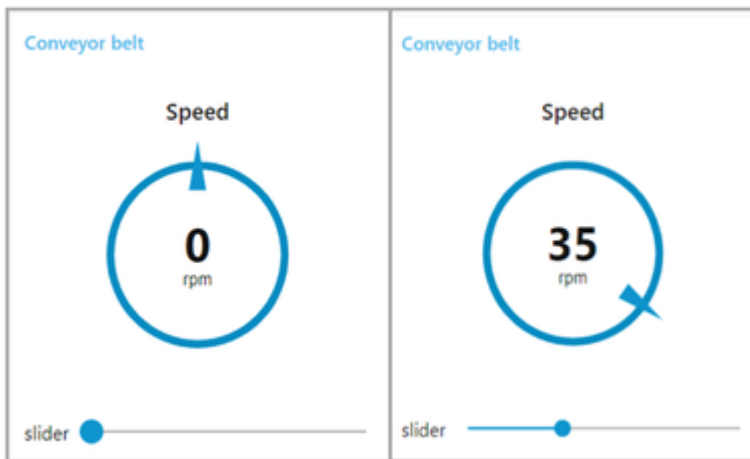


- ① Search for a dashboard
- ② Published dashboard

You can click on the published dashboard.

The required dashboard flow will now be visible on the Visual Flow Creator Dashboard screen.

You can slide the pointer and get the desired results:




7.12 Adding links to Dashboard Viewer

You can add the links to "Dashboard Viewer" and access the link from sidebar of Visual Flow Creator dashboard.

Procedure

To add the link to the dashboard, follow these steps:

1. In "dashboard" tab, click  .

2. Edit the link node properties:

Dialog box titled "Edit link node" with the following fields and options:

- Name: Visual Flow Creator tutorial
- Link: <https://siemens.mindsphere.io/en/docs/tutorials/vf...>
- Icon: open_in_browser
- Open in: New Tab, This Tab
- Show in the sidebar of dashboard opened from Dashboard Viewer
- Access: Do not show in Dashboard Viewer
- On all flows

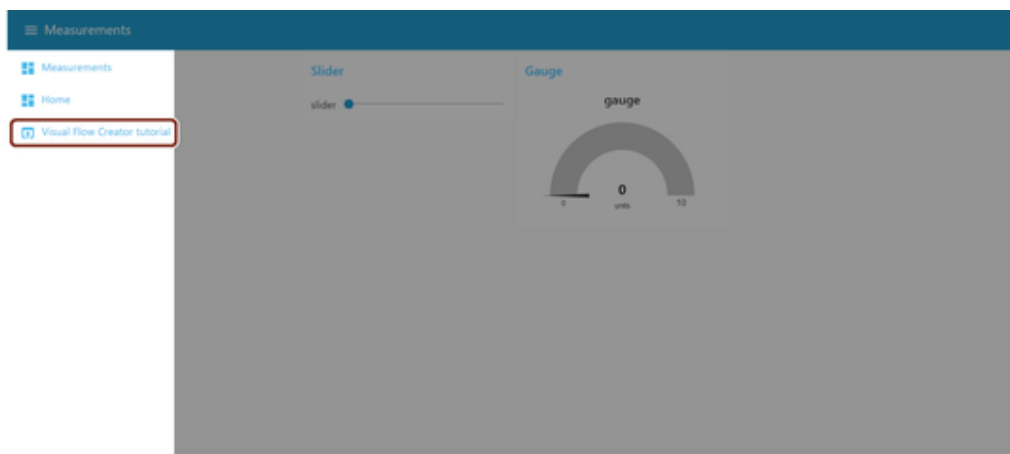
- Name: Visual Flow Creator tutorial
- Link: "<https://siemens.mindsphere.io/en/docs/tutorials/visual-flow-creator-KPI>"
- Show in the sidebar of dashboard opened from Dashboard Viewer: Check

3. Save and deploy.

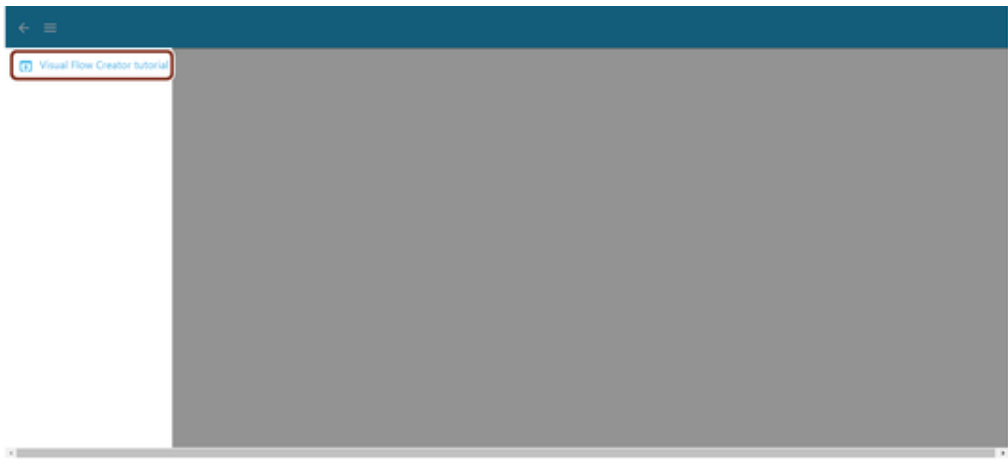
Result

To view the results, you can access the link from the dashboard:

Accessing link from VFC Dashboard



Accessing link from Dashboard Viewer

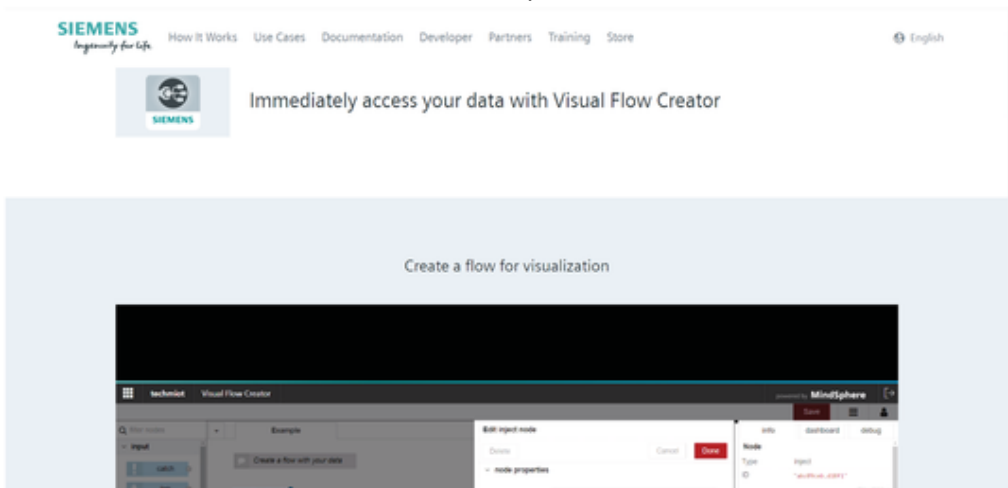


You can click any tile in "Dashboard Viewer".



If "Show in the sidebar of dashboard opened from Dashboard Viewer" is unchecked, then the link will not be accessible from "Dashboard Viewer".

Click "Visual Flow Creator tutorial" link to open the tutorial website.





Optimizing Visual Flow Creator flows

8

8.1 Optimizing Visual Flow Creator flows

You can optimize the Visual Flow Creator executions to avoid the excess consumption of compute hours.

| Avoid having sequential executions | Avoid using multiple inject nodes for parallel executions |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| When executing the nodes, try avoiding sequential flows. | When executing the parallel executions, try to execute with single |
| Recommended is to execute the flows in parallel. | inject node. |
|  |  |

Subflows

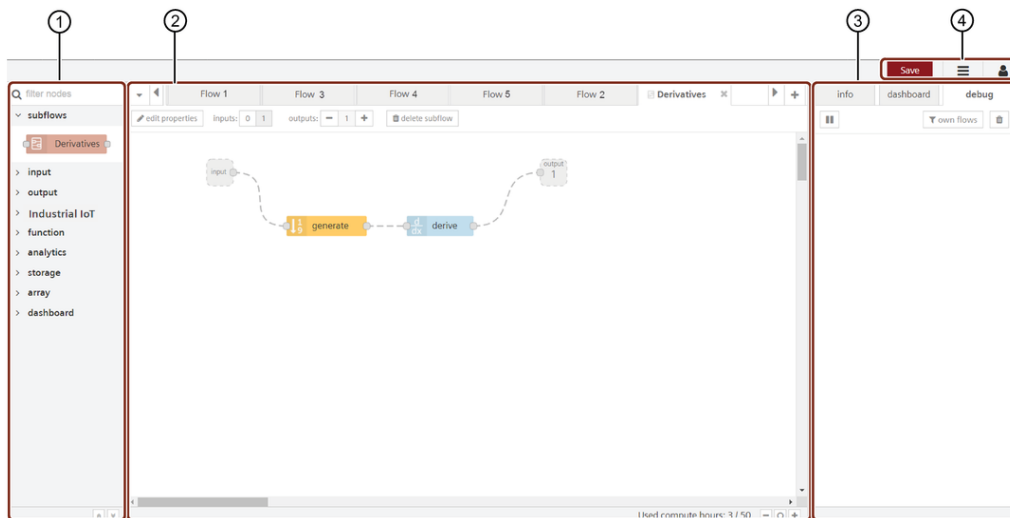
9.1 Subflows

Working of a subflow

A subflow is a set of nodes which can be bundled together for re-using later like a function. The subflows can be used in multiple larger flows as a single node which in turn reduces the complexity in a flow.

The subflow feature in VFC is an adaptation of the "subflow" in Node-RED. For more information, refer to [Node-RED subflows](#).

User interface of a subflow




- ① Node palette where subflow nodes are created under "subflows"
- ② Working area to design subflows
- ③ Sidebar
- ④ Tools menu

Create a subflow

You can design subflows using the two methods:

- Create a new subflow
- Selection to subflow

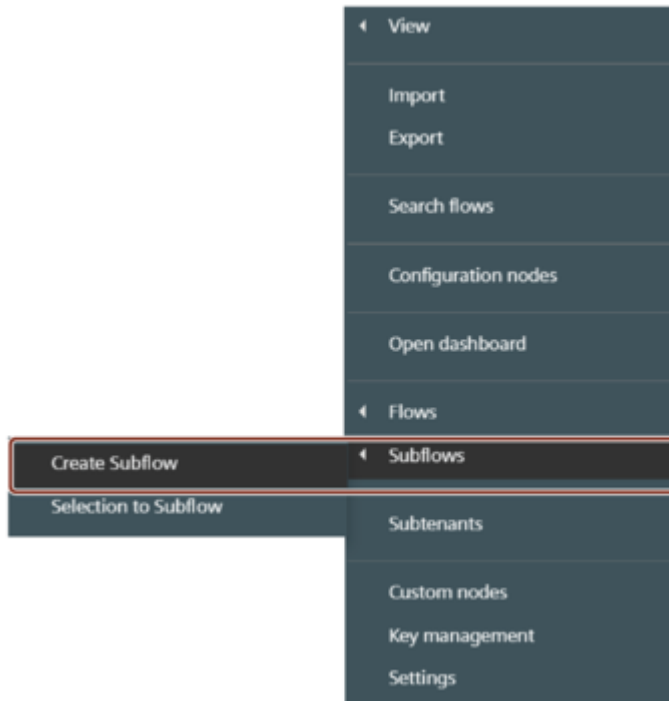
Both the methods are described below.

The subflow option is available in the menu bar  .

Create a subflow

To create a subflow, proceed with the following steps:

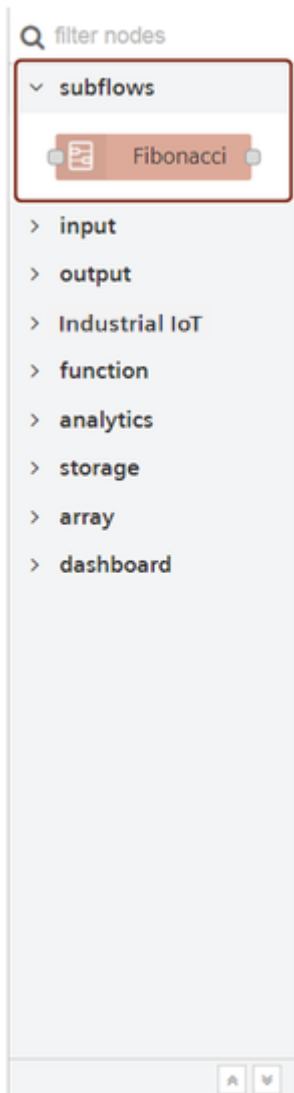
1. Click the menu bar option and choose the subflows option. Select "Create Subflow".



2. Create a new subflow as required.
3. Click  to set the name and description of the subflow. This is optional.

Result

A new subflow will be created and will be visible in the node palette under section "subflows".



You can now use the created subflow as a single node in other flows.

Selection to subflow

You can create a subflow while designing a flow.

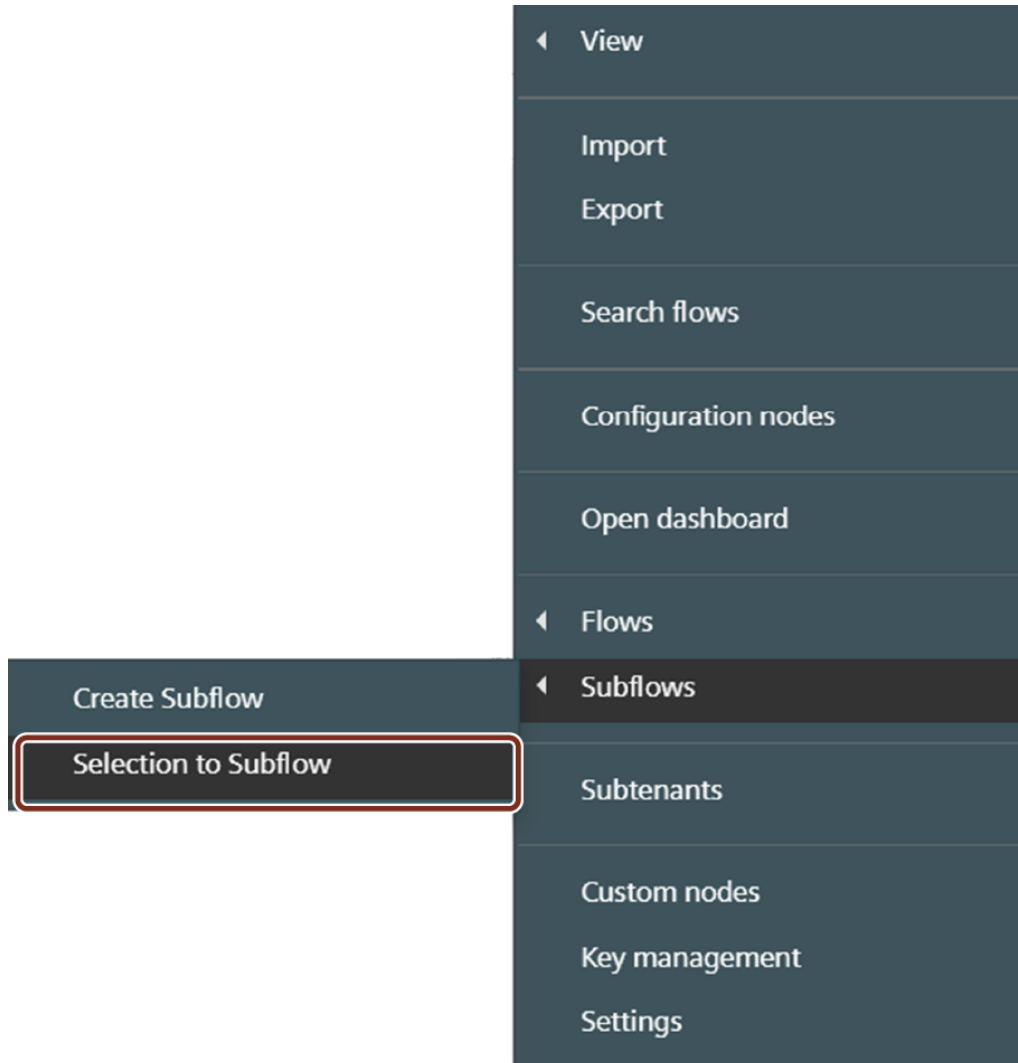
Let us consider the following flow:




Proceed with the following steps to create subflows while designing a current flow:

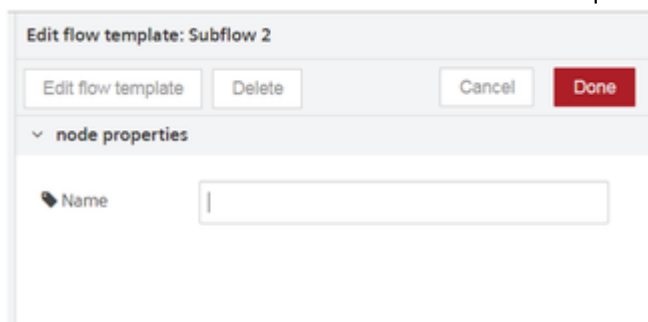
1. Select the "generate" and "filter" nodes.

2. Open the menu bar  and select the "Selection to Subflows" option.



 The "Selection to subflow" option will be visible only when you select the nodes in your working space.

3. Double click the created subflow and edit if required.





Clicking delete button removes the subflow from your current working space and not from the node palette.

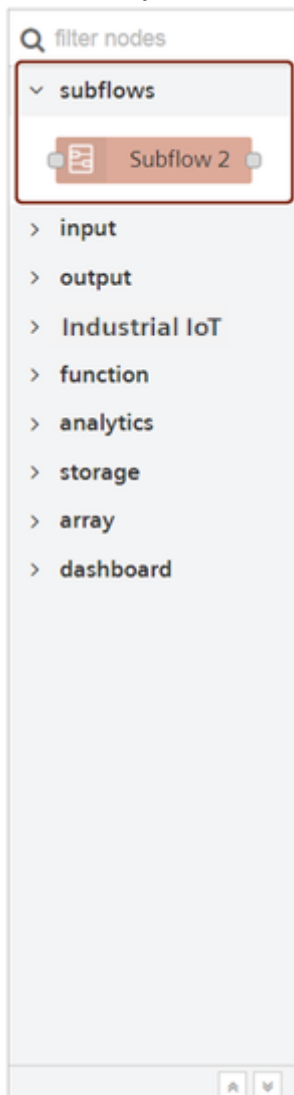
To delete subflows from the node palette, refer to [Delete subflows](#).

Result

1. The selected nodes will automatically be converted to a subflow.



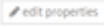
2. The newly created subflow will be visible in the node palette.



Edit a subflow

There are two ways to edit properties of a subflow:

Edit from Node Palette

Double click the node from the palette under "subflow" section. Click  and edit the subflow properties as required.

Edit from currently opened subflow

Open the working subflow and click on  to edit the subflow properties.

Inputs and outputs

You have the option of designing flows at most with one input

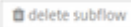
inputs: 0 1

But flows can be designed with as many outputs as required

outputs: - 1 +

Delete a subflow

You can delete a subflow when it is no longer required.

Double click the required subflow from the node palette and select  from the subflow working space. Once you delete the subflow, it will no longer be visible in the node palette.

10.1 Custom nodes

The custom nodes helps to design and deploy your own customized nodes if any of the required nodes are not available in Visual Flow Creator.

Custom nodes are initially not available in the node palette with all the nodes in VFC. You need to create them first and then deploy. Thereafter, they will be automatically integrated in the palette.



Custom nodes can access to context variables with limited for local and flow-context whereas global and environment context variables are not accessible.

Design custom nodes

Visual Flow Creator is composed into multiple components which are required for developing custom nodes. They are:

Editor component ("editor"): While deploying the custom node modules, files are populated to the editor. The frontend resources (HTML files, icons, language files) of the editor are delivered to the users' browser by integrating them in the requests, against

- which responses are generated.

Runtime component ("runtime"): The runtime is responsible for executing the flows by executing the backend code of each node. The runtime can access all provided files (HTML files, icons, Javascript files, etc.). When a node gets triggered, the message sent to the node is forwarded to the runtime which starts thereafter. The runtime executes the code and forwards the debug and status messages to the editor. It also forwards

- messages to the next connected nodes.



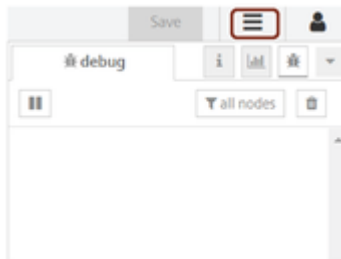
The number of messages which a custom node can send / forward is restricted to 20 messages per execution.

You can create custom nodes at your convenience and then can use the nodes to design flows in the Visual Flow Creator.

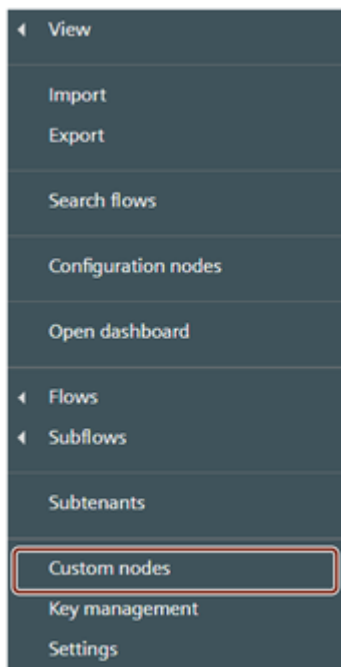
The nodes have the possibility to integrate both frontend and backend functionalities. In special cases, the custom nodes they can use backend functionality if the functionality is provided by the user but cannot call custom backend functionalities.

Create a custom node

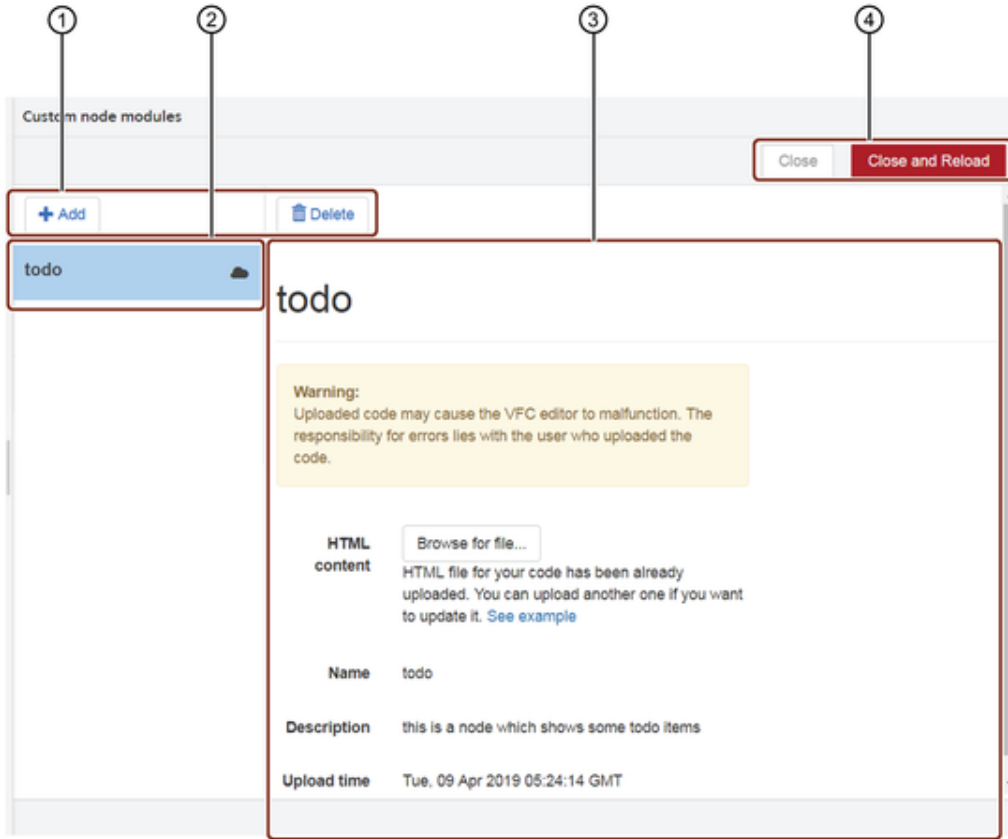
1. Open the menu button from the top right corner of the Visual Flow Creator screen.




2. Select "Open custom nodes page" from the menu.



3. The custom node screen dialog is displayed:



- ① Actions buttons
- ② Custom component window
- ③ Custom component details
- ④ Close / reload after uploads buttons



Only users with the `mdsp:core:vfc.admin` permissions are allowed to access the custom nodes configuration screen.

4. Browse to upload the relevant zipped custom node content file which can contain multiple nodes. The file should be zipped and should have two or more files.

Configurations for Custom Nodes:

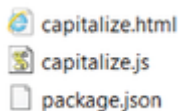
The zip archive should have three mandatory files - HTML node file, javascript file and the package.json file.

You can include other files in the zip archive like:

- Icon: The icons have to be in a directory called "icons".
- Language: The language files have to be in a directory called "locales/", for example "locales/en-US".

The zip archive can also have multiple HTML node files.

The zip archive structure looks as shown below:



Configurations for special Custom Nodes:

The zip archive must have two mandatory files - HTML node file and the package.json file. Absence of the package.json file will throw an error.

You can include other files in the zip archive like:

- Icon: The icons have to be in a directory called "icons".
- Language: The language files have to be in a directory called "locales/", for example "locales/en-US".

The zip archive can also have multiple HTML node files.



The archive is immediately uploaded to the server and validated.

The nodes are then extracted. The user always uploads "modules" which can contain multiple nodes. The following information is shown in the custom nodes overview:

- Name: Name of the new component.
- Description: Short summary of the component.
- Upload time: The field displays the current date and timestamp of the upload.

5.Click the "Close and reload" button at the top right corner to load the custom node. This displays the created node(s) to Visual Flow Creator node palette.


Some of the node-red nodes available on the internet may have dependencies.

The zip archive structure looks as shown below:



These dependencies should be installed manually as below:

1. Download and extract the .zip package file.
2. Run npm install to install.
3. Create a new .zip archive.



- Avoid the binary dependencies while creating custom nodes.
- Custom nodes can be created/ deployed maximum 12 times in 12 hours.

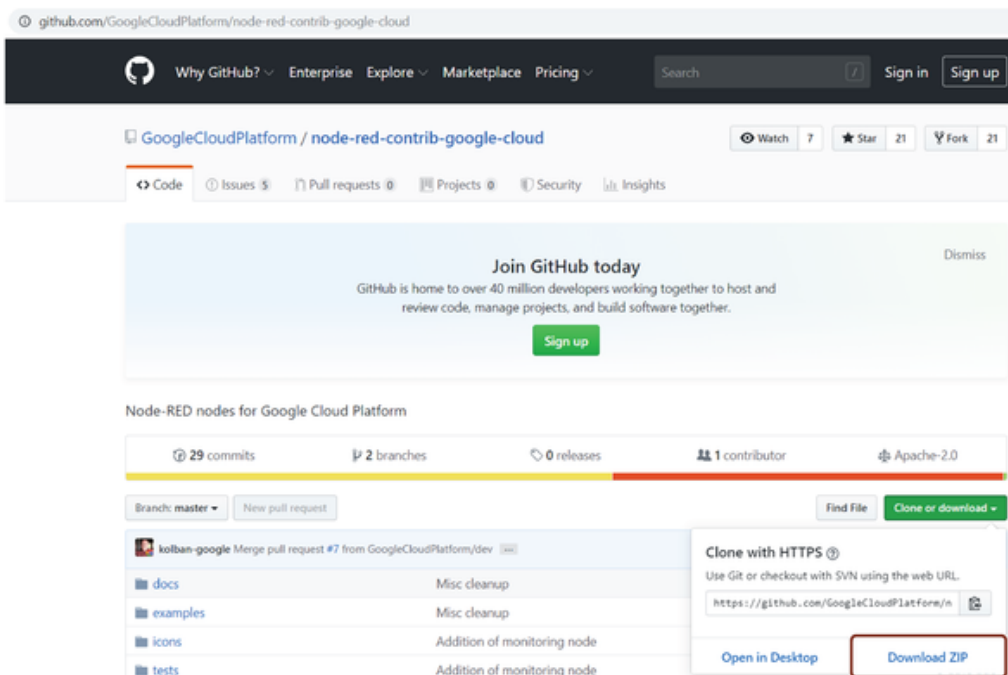
Result

The custom node is created with the chosen name under the designated section you have defined in the html file.

Ready-to-use nodes

The Ready-to-use nodes can be downloaded as described below:

- They can be downloaded via the CLI with npm command 'npm pack packetName' and then the packetName.tgz file could be uploaded directly. You can find the nodes in [Node-RED Library](#).
- They can be cloned or downloaded from Github. Download repo and the zip archive can be uploaded without modifications into custom nodes dialog. Example, [Node-RED nodes for Google Cloud Platform](#) and press on "Clone or download" and then "Download ZIP".



Custom node dependency

If the custom node has any external module dependencies, they must be installed in the dependencies section of its package.json file. You can follow below steps to install the dependency:

1. Download the external module .zip file from the internet.
2. Check for the dependencies in package.json file.
3. To install the dependencies, enter "npm install" in CLI.
4. After installing, zip the external module node file.

You can use this .zip file to create a custom node in Visual Flow Creator.



Avoid using binary node dependencies.

Implement custom nodes - Example 1

Example scenario

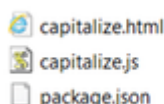
Create a custom node to design flows where text inputs are displayed in upper case.

Objective

To convert a message payload into upper case characters.

Pre-requisites

Create a directory having the following files:



The example for the capitalize.html file is given below:

```
??? Example html <script type="text/x-red" data-template-name="capitaliz
e"> <div class="form-row"> <label for="node-input-name"><i class="fa f
a-tag"></i> <span data-i18n="common.label.name">Name</span></label> <i
nput type="text" id="node-input-name" placeholder="Name" maxlength="6
4"> </div> </script> <script type="text/x-red" data-help-name="capital
ize"> <p>Capitalizes msg.payload</p> </script> <script type="text/java
script"> RED.nodes.registerType('capitalize',{ category: 'custom node
s', color:"#C0DEED", defaults: { name: {value: "", required: false} }
inputs: 1, outputs: 1, label: function() { return this.name || "capita
lize"; }, labelStyle: function() { return this.name ? "node_label_ital
ic":""; }, paletteLabel: "capitalize", icon: "watch.png" }); </script>
```

The example for the capitalize.js file is given below:

```
??? Example javascript module.exports = function(RED) { class CapitalizeN
ode { constructor(config) { RED.nodes.createNode(this, config); this.o
```

```
n('input', this.onInput); } onInput(message) { message.payload = message.payload.toUpperCase(); this.send(message); } } RED.nodes.registerType("capitalize", CapitalizeNode); }
```

The example for the package.json file is given below:

```
??? Example JSON { "name": "capitalize", "version": "1.0.0", "description": "", "main": "capitalize.js", "node-red": { "nodes": { "capitalize": "capitalize.js" } } }
```

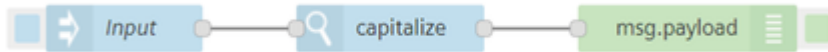
Upload the zipped file and create a custom node in the custom nodes page.

Procedure

1. Select the "Capitalize" node from the "custom nodes" section.

Connect an input node and input your string in the properties section. Connect the

2. "Capitalize" node with a message payload as shown below:



3. Inject the timestamp.

Result

The output is displayed in the message payload:



Implement custom nodes - Example 2

Example scenario

A management team wants to read data from AWS S3.

Objective

To view the detailed event information including the actions and resources which the developers require.

Pre-requisites

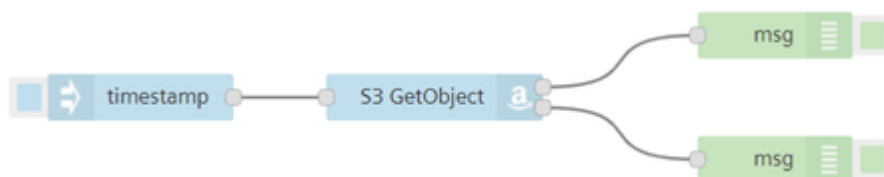
Go to the website: [Node-RED nodes for AWS](#).

Clone or download the repository and include all the files in the zip archive. Create a custom node "AWS" in the custom nodes page by browsing the zip archive.

Procedure

Select the "AWS S3" node from the "AWS" section. Edit the node in the properties if required
1. by setting its operation to "GetObject".

Connect the "S3 GetObject" node and attach the timestamp input and the message payload
2. outputs as shown below:



In this example, the node is configured in a way to pass correct data in one payload and throw error in the other if any. Hence, in the message payload, you get only one of the output results.

3. Inject the timestamp.

Result

The output will be visible in the message payload. As mentioned, the output would either be real data or an error.

①

```

15.7.2019, 11:08:11.585 node: b37cc4a2.52a0c8
msg: Object
  object
    _correlationId:
      "1177cda0a6e011e9b78701eb219b9854"
    topic: ""
    payload: object
      AcceptRanges: "bytes"
      LastModified: "2019-04-16T07:33:20.000Z"
      ContentLength: 1204
      ETag: ""7970288ce8df5ce7a507c67e3e34fbdf""
      ContentType: "application/octet-stream"
      ServerSideEncryption: "AES256"
      Metadata: object
      Body: buffer[1204]
    _msgid: "4518cdfd.6dd564"
    _starttime: 1563181690876
            
```

②

```

7/15/2019, 2:34:24.576 PM node: 57109def.fa9cf4
msg: Object
  object
    err: object
      message: "Access Denied"
      code: "AccessDenied"
      region: null
      time: "2019-07-15T09:04:24.443Z"
      requestId: "F282273298B693BD"
      extendedRequestId:
        "7213b0ARYzRY+0/iDPaPqdo83e17ngi3xj8z."
      statusCode: 403
      retryable: false
      retryDelay: 157.78742547203652
      _msgid: "137cd9a8.bcacc6"
      _starttime: 1563181464575
      _correlationId:
        "8aa8d8f0a6df11e9b8109f7816930362"
            
```

- ① Real time data
- ② Error data

Implement custom nodes - Remote backend functionality

Example scenario

Consider a new REST API example: TODO List. Click [Todo List Rest API](#) to view the complete list.

Objective

To get the TODO list from free REST API. This will display a list of users with their task list.

Pre-requisites

Create a custom node "todo" in the custom nodes page by browsing the zipped node file.

For your reference, the HTML code for "todo" is given below:

```

??? Code html <script type="text/x-red" data-template-name="todo"> <div
class="form-row"> <label for="node-input-todoId"><i class="fa fa-tag">
</i> <span data-i18n="common.label.name">Todo ID</span></label> <input
type="number" id="node-input-todoId" placeholder="Leave blank to get a
ll entries"> </div> </script> <script type="text/x-red" data-help-name
="todo"> <p>Gets todo list using the jsonplaceholder free API.</p> </s
cript> <script type="text/javascript"> RED.nodes.registerType('todo',
category: 'service nodes', color: '#C0DEED', icon: 'todo.png', default
s: { todoId: {value: '', required: false}, requestParams: { value: { u
rl: 'https://jsonplaceholder.typicode.com/todos' } } } inputs: 1, outp
uts: 1, icon: 'c.png', label: function() { return 'todo'; }, paletteLa
bel: 'todo', oneditprepare: function() { }, oneditprepare: function() { t
his.requestParams.url = 'https://jsonplaceholder.typicode.com/todos/'
+ todoId; } }); </script>

```



HTML files follows the Node-RED format, except for request params which is shown in the above HTML code.

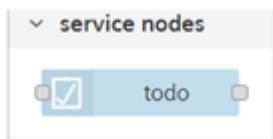
A sample package.json file is shown below:

```

{
  "name": "todo",
  "version": "1.0.0",
  "description": "this is a node which shows some todo items",
  "main": "index.js"
}

```

In this case, a custom node "todo" is created under the section "service nodes" in the node palette of Visual Flow Creator.



Procedure

1. Select the "todo" node from the "services nodes" section.

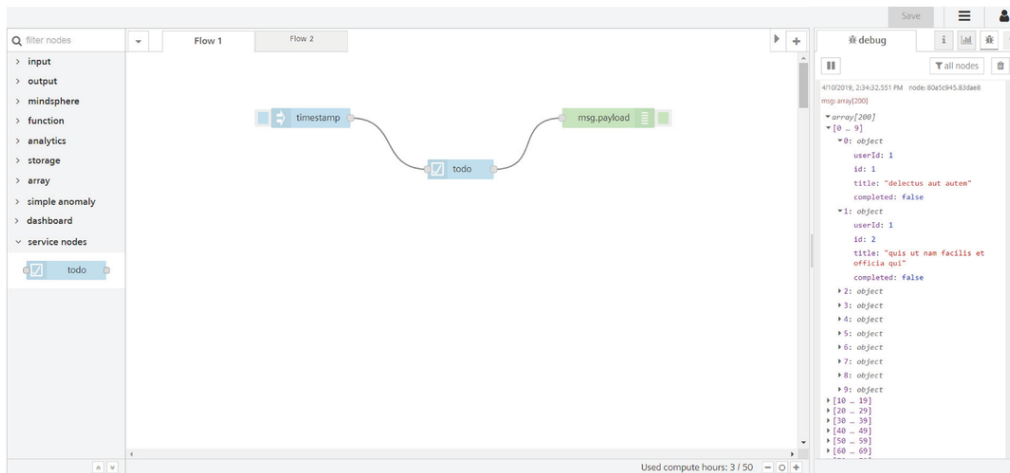
Connect the "todo" node and attach the timestamp input along with a message payload output. Edit the "todo" node in the properties if required. Refer to "Create custom nodes" 2. section to view the HTML code.



3. Inject the timestamp.

Result

The output will be visible in the message payload:



Example for execution time calculation in VFC 11

11.1 Example for execution time calculation in VFC

Visual Flow Creator calculates the execution time of your flows. The calculation starts when a flow is started and ends if all the involved nodes stop their execution. The time will be rounded up to the next 100ms.

For example, if you execute a flow which takes 240ms to complete, 300ms would be charged from your available compute hours.

Example

The following examples show:

- This flow sends the email notification in **200ms**.

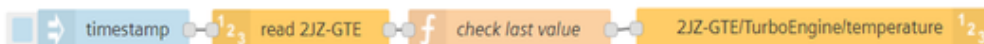


This flow reads the timeseries data to check the last value and writes the new value in

- **400ms**.

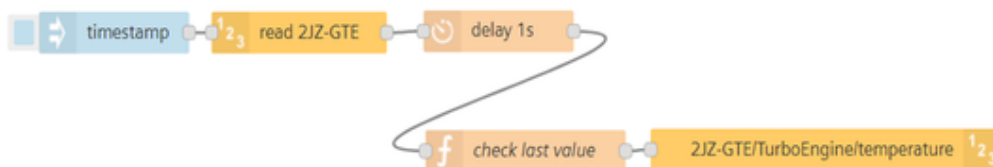


This example could be modified in order to calculate KPI's. If the flow is runs every hour, the monthly consumption would be " $0.4 * 24 * 30 = 288$ seconds".



This flow shows the "wait time" will not be charged. **200ms + 300ms = 500ms**. In this flow insert delay nodes will split the flow execution into two parts. Each execution will be

- charged.

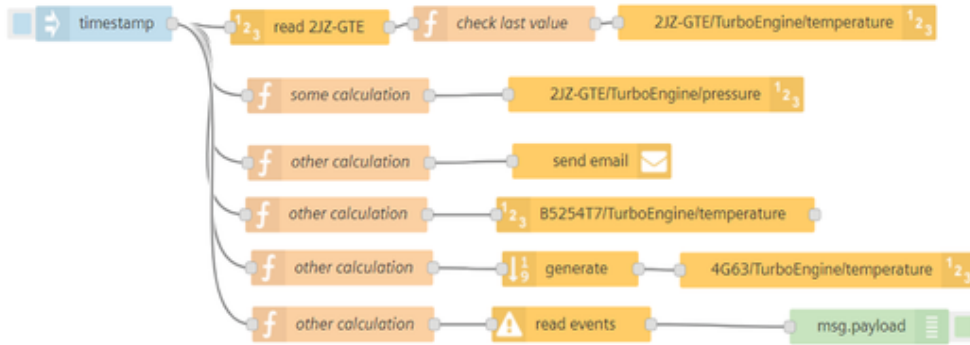


- This flow shows that the timed out flows will be charged for **30 seconds**.



This flow can optimize the execution time of "multiple" flows using only one inject node in

- **800ms will be charged for the execution of the displayed flow.**



12.1 Appendix

Technical specifications

The following table contains the technical specifications of Visual Flow Creator:

| Description | Limitations |
|---------------------------------------------------------------------------------------------------------------------|---------------|
| Maximum number of flow tabs per user in the working area | 30 |
| Maximum number of nodes within a flow tab | 200 |
| Maximum project size limit | 3 MB |
| Maximum number of projects | 10 |
| Maximum calculation duration of a flow | 30 seconds |
| Maximum number of the node context variables in function node | 5 per node |
| Maximum number of the flow context variables in function node | 20 per tab |
| Maximum number of the global context variables in function node | 100 per user |
| Time period for saving data in the context of the "function" node after the last read/write access or system reboot | 6 months |
| Maximum number of context tags within the "function" node | 1 |
| Maximum file size for data transmission within the "read file" and "write file" nodes | 1 MB |
| Maximum number of time series values per read operation | 2000 |
| Maximum limit of global context variables | 100 |
| Maximum size of a global context for the Production / Start for Free environments | 5 MB / 128 KB |
| Maximum flow context variables | 20 |

| Description | Limitations |
|----------------------------------------------------------------------------------------|---------------|
| Maximum size of a flow context for the Production / Start for Free environments | 5 MB / 128 KB |
| Maximum node context variables | 5 |
| Maximum size of a each node context for the Production / Start for Free environments | 2 MB / 128 KB |
| Maximum size of a environment context for the Production / Start for Free environments | 5 MB / 128 KB |
| Maximum size for an email notification | 250 KB |
| Maximum size of a function node for all environments | 1 MB |
| Maximum limit of emails per day per environment | 1000 |
| Maximum limit of MQTT messages per environment in 1 minute | 10 |
| Maximum limit of MQTT messages per environment in 1 hour | 500 |
| Maximum number of HTTP-In requests per second per user | 2 |
| Maximum size of data that combine node can cache | 12KB |

Visibility of flows/ execution time of nodes

- All created flows are visible for all users of a environment. You cannot modify the flows of another user.
- The execution time of all nodes for a environment can be limited depended on the application and platform load.

Differences between Visual Flow Creator and Node-RED

The following table shows the difference between Visual Flow Creator and Node-RED:

| Visual Flow Creator | Node-RED |
|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| Maximum running duration of one flow is 30 seconds (or 2 minutes with the power mode option). | Maximum running duration of one flow is not limited. |
| Application runs on AWS and Azure. | Application runs on local system. |
| Maximum 5 custom modules can be uploaded for the custom nodes. | No limit to create custom modules using Node-RED library. |

| Visual Flow Creator | Node-RED |
|-------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| No palette manager for the additional nodes. | All nodes can be managed from the palette manager. |
| Industrial IoT authentication or private key is required to access the http endpoint using "Http-in" node. | No authentication is required to access the http endpoint with "Http-in" node. |
| No access to <code>msg.req</code> and <code>msg.res</code> objects in function nodes. | Access to <code>msg.req</code> and <code>msg.res</code> objects in function nodes is available. |
| No authentication is required to access APIs. | Authentication is required to access APIs. |
| Name of the global context variable is "glob". | Name of the global context variable is "global". |
| "Tenant" context variable is available, in addition to "glob", "flow" and "context". | No "environment" context variable. |
| Many Industrial IoT specific nodes are available | No specific Industrial IoT nodes. |
| Dashboard nodes are extended to support multi-user functionality using "Only one browser" option in edit dashboard node properties. | No dashboard nodes are extended to support multi-user functionality |
| Dashboard Viewer is available. | No separate Dashboard Viewer. |
| Insights Hub Monitor plugin is available. | No Insights Hub Monitor plugin. |
| No keyboard settings. | Keyboard settings is available. |

13.1 Glossary

IoT

IoT is the abbreviation for the Internet of Things. The IoT networks objects of every type with the Internet. The objects can communicate with one another over the Internet and perform various tasks.

Node

Nodes are configurable pieces of functionality that process input data and produce output data. Examples range from data processing to complex functions and online services.

Payload

Payload is the container for the user data (data packet). User data can be simple strings or arrays. An array consists of mapping JavaScript objects. For processing of time series data, arrays of objects are used that consist of a time stamp and a measured value, e.g.: { "_time": "2017-12-27T07:35:45", "pos_X": 112.2 }.