



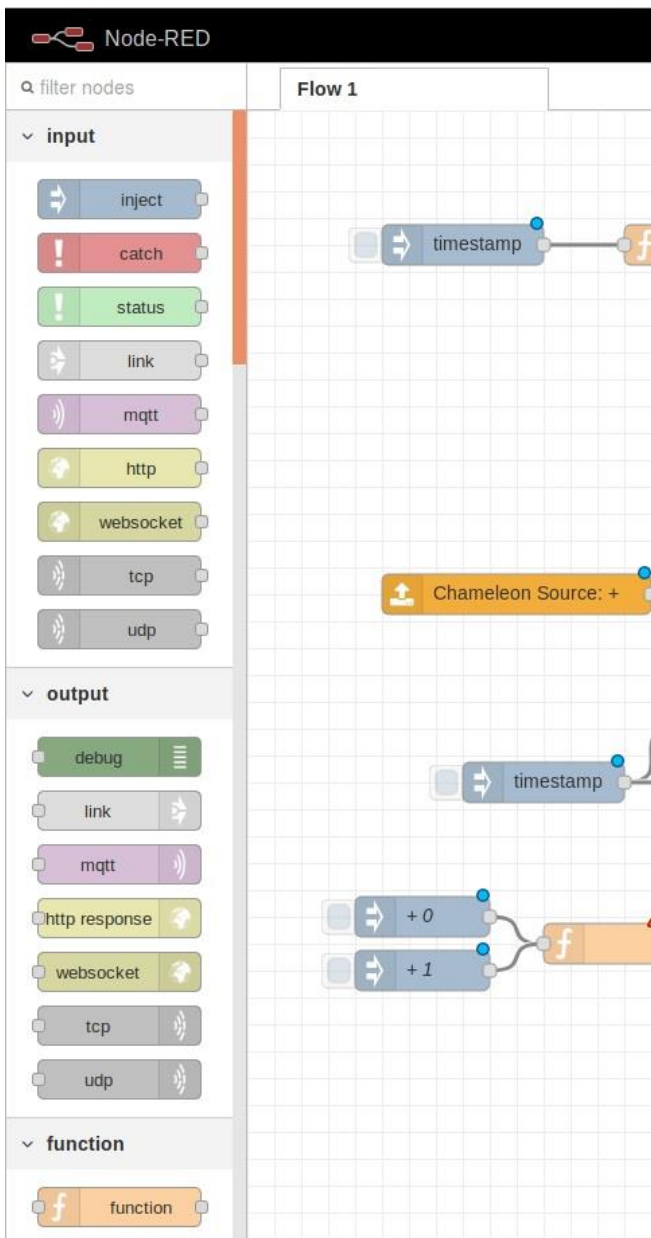
iii PRIDE
資策會 40 週年

Chameleon Node-RED 教育訓練



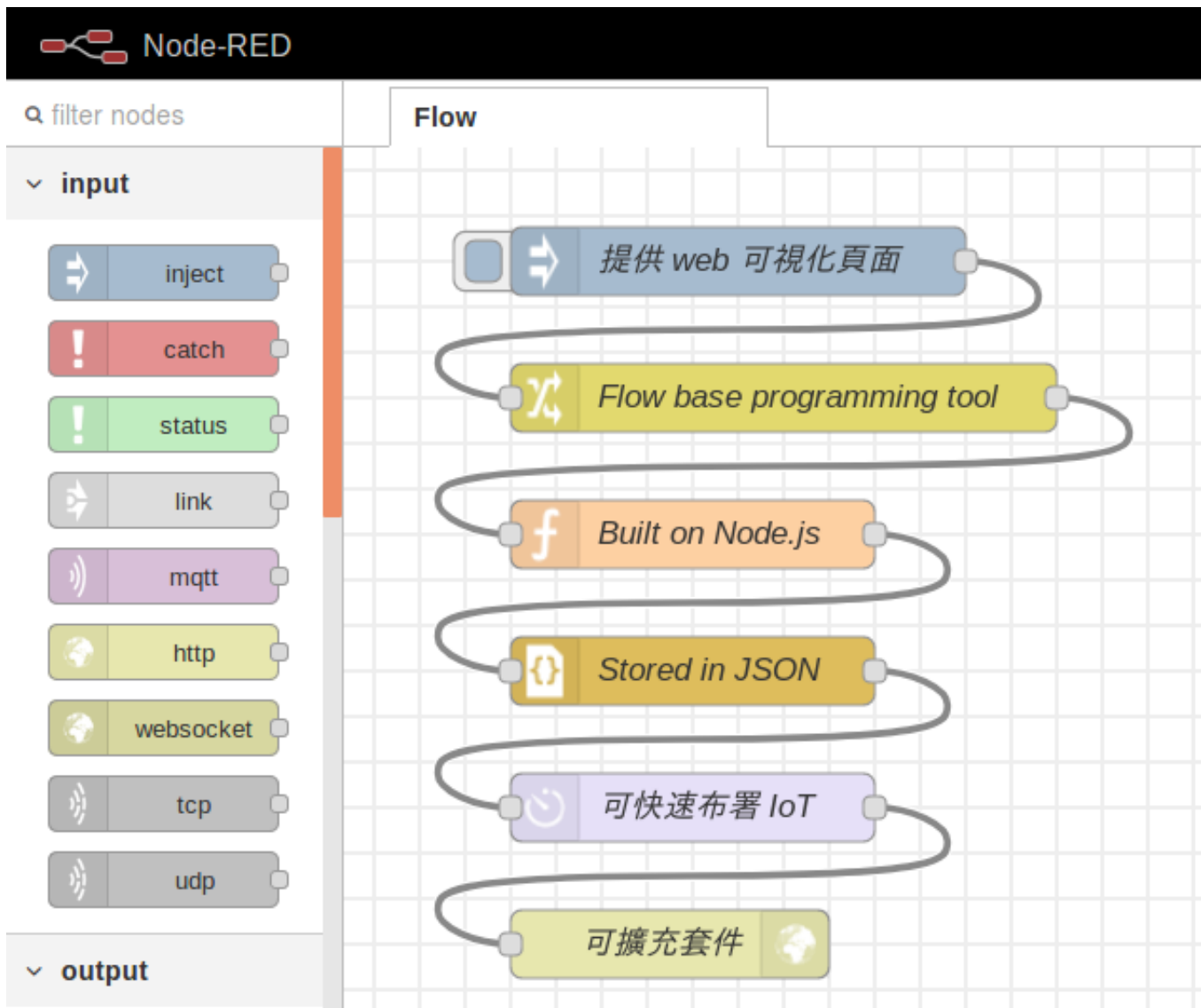
Agenda

1. Node-RED 是什麼?
2. 基本操作
3. 常用 node 介紹
4. Chameleon x Node-RED
5. 練習 & QnA





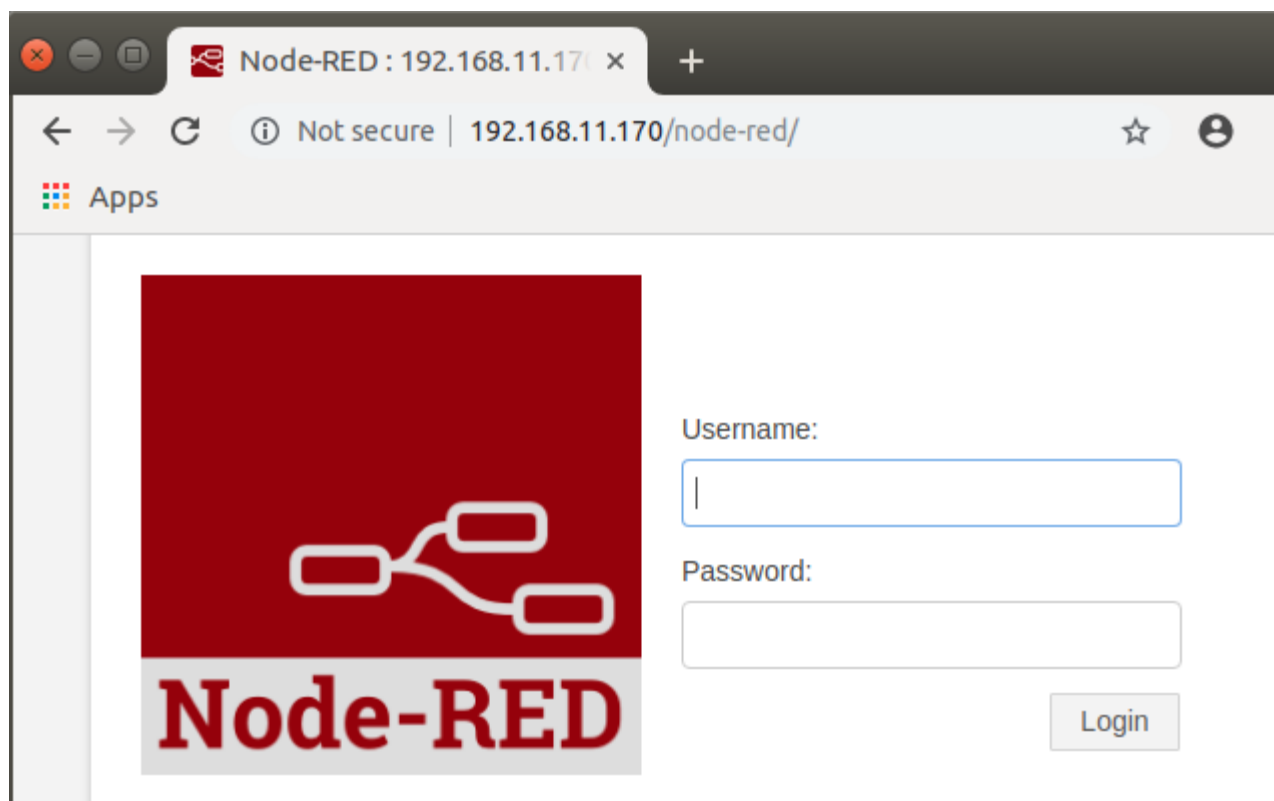
What is Node-RED





如何開啟 Node-RED

- 變色龍的 URL + /node-red
- 帳號密碼和變色龍的相同





基本介紹

The screenshot shows the Node-RED web interface. On the left, a sidebar contains a search bar and two categories of nodes: 'output' (debug, link, mqtt, http response, websocket, tcp, udp) and 'function' (function, template, delay). A red box highlights the entire sidebar area, with a red arrow pointing to it labeled 'node'. In the center workspace, a flow named 'Flow 1' is shown. It consists of an 'inject' node connected to a 'function' node, which is then connected to a 'msg.payload' output node. A red box highlights this entire flow, with a red arrow pointing to it labeled '這也是 flow'. Above the workspace, a red arrow points to the '+' button with the label '新增 flow'. On the right, a sidebar shows the 'info' tab with a table of flow information. A red arrow points to the 'info' tab label with the label 'node 訊息'. Another red arrow points to the 'debug' icon (a bug) with the label 'debug 訊息'. The table in the 'info' tab is as follows:

node 訊息	
Information	
Flow	"adb52fe4.95b53"
Name	Flow 1
Status	Enabled
Description	

At the bottom of the right sidebar, there is a text box that says: 'Dragging a node onto a wire will splice it into the link'.



編輯 flow

Node-RED

Filter nodes

output

- debug
- link
- mqtt
- http response
- websocket
- tcp
- udp

Flow 1 點2下

Edit flow: Flow 1

Delete 刪除 flow

Cancel Done

Name Flow 1 改 flow 名稱

Status ☒ Enabled enable / disable

Description

h1 h2 h3 B I </> [List Icon] [List Icon] [Quote Icon] [Minus Icon] [Link Icon]

1



編輯 node

double click

Edit trigger node

Delete Cancel Done

Properties

Send a_z 1

then wait for

250 Milliseconds

☐ extend delay if new message arrives

then send a_z 0

Reset the trigger if:

- msg.reset is set
- msg.payload equals optional

Handling all messages

Name Name



儲存

Apps

修改完記得儲存

Node-RED

filter nodes

Flow 1

output

- debug
- link
- mqtt
- http response
- websocket
- tcp
- udp

function

- function
- template
- delay

Flow 1 diagram:

```
graph LR; inject[inject] --> function[function]; function --> msg_payload[msg.payload];
```

info

Information

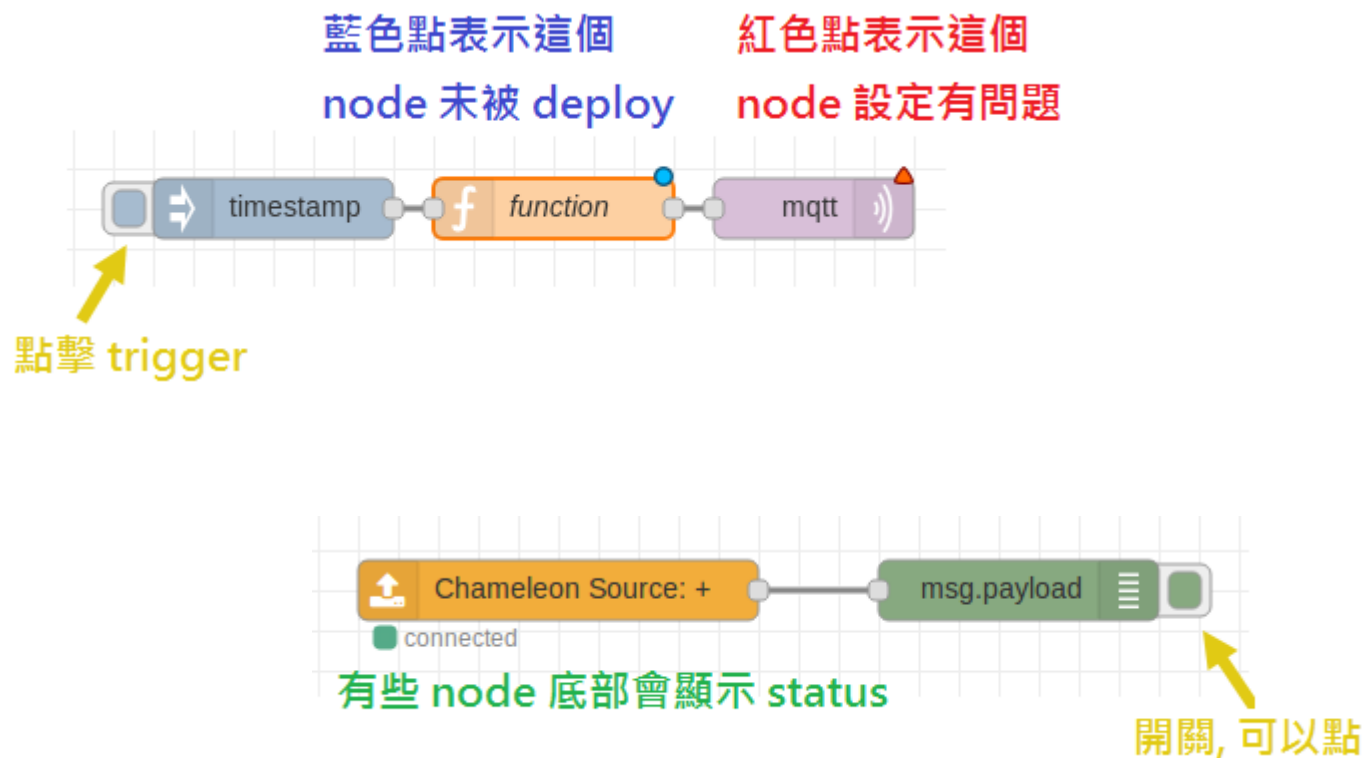
Flow	"adb52fe4.95b53"
Name	Flow 1
Status	Enabled

Description

Dragging a node onto a wire will splice it into the link



node status





匯出 flow or node

The screenshot illustrates the process of exporting a flow or node in Node-RED. On the right, the main menu is open, with the 'Export' option highlighted. A red arrow points from the 'Export' menu item to the 'Export nodes' dialog box on the left. The dialog box has three tabs: 'selected nodes', 'current flow', and 'all flows'. The 'selected nodes' tab is active, and the text '選擇的 node / 這個 flow / 所有 flow' (Selected node / This flow / All flows) is written in red above the tabs. Below the tabs, a text area displays JSON data in 'json 格式' (JSON format), with a red box highlighting the first node's coordinates: `"x": 360,`
`"y": 240,`
`"wires": []`. At the bottom of the dialog, the 'Export to clipboard' button is circled in red, with the text '複製到剪貼簿' (Copy to clipboard) written in red next to it. Other buttons visible are 'Cancel', 'Download', and 'compact formatted'.



匯入 flow or node

The screenshot shows the Node-RED web interface. On the left is the node palette with categories like 'output' and 'function'. The main workspace is titled 'Flow 1'. A menu is open, showing options: 'Clipboard', 'Library', and 'Examples'. A red arrow points from the 'Clipboard' option to the 'Import nodes' dialog box. The dialog box has a text input field with the red text '這邊貼上之前匯出的 json string' (Paste the json string exported previously here). Below the input field are 'Import to' buttons for 'current flow' and 'new flow'. At the bottom of the dialog are 'Cancel' and 'Import' buttons. The 'Import' button is circled in red, with the red text '匯入' (Import) written above it. On the right side of the interface, a hamburger menu icon is circled in red, with a red arrow pointing from it to the 'Import' option in the open menu.

Node-RED

Deploy

filter nodes

Flow 1

output

- debug
- link
- mqtt
- http response
- websocket
- tcp
- udp

function

- function
- template
- delay

Import nodes

Paste flow json or

這邊貼上之前匯出的 json string

Import to

Cancel

匯入

View

Import

Export

Search flows

Configuration nodes

Flows

Subflows

Manage palette

Settings

Keyboard shortcuts

Node-RED website

v0.20.7



node & message

- node 是最基本的元件
- node 之間使用線連接來組成 flow
- node 與 node 間傳遞使用 message, 程式中使用 **msg** variable 來取得訊息
- msg 的 type 是 JavaScript objects, 其中 **payload** property 是最常用來放要給下個 node 的資料
 - msg.payload
 - Payload 內容可以是任何的 JavaScript type



常用 node 介紹

The screenshot shows the Node-RED web interface. In the center workspace, an 'inject' node is connected to a 'msg.payload' node. A red arrow points from the 'inject' node to the 'Edit inject node' dialog box on the left. Another red arrow points from the 'msg.payload' node to the 'debug' window on the right. The 'Edit inject node' dialog shows the 'Payload' field with a dropdown menu open, displaying options like 'flow.', 'global.', 'string', 'number', 'boolean', 'JSON', 'buffer', 'timestamp', and 'env variable'. The 'debug' window shows the message content: 'this is from inject node'.

善用 debug node 來知道 msg 內容

- **inject:**
 - 產生 string, json...等資料帶給下個 node
 - 設定每隔一段時間 inject 一次
- **debug:** 查看 msg 內容
- Ref: example/exp1.json



常用 node 介紹

The screenshot shows a Node-RED workflow on a grid. It starts with a 'timestamp' node (blue square with a right arrow), followed by a 'resend every 5s' node (purple rectangle with a square-wave icon), and finally a 'msg.payload' node (green rectangle with a list icon). A red arrow points from the 'resend every 5s' node to the 'Edit trigger node' dialog box.

The 'Edit trigger node' dialog box has a title bar with 'Delete', 'Cancel', and 'Done' buttons. Below the title bar is a 'Properties' section with a gear icon. The 'Send' dropdown is set to 'the existing msg object'. The 'then' dropdown is set to 'resend it every'. Below this, there is a field for '5' and a dropdown for 'Seconds'. The 'Reset the trigger if:' section has two bullet points: 'msg.reset is set' and 'msg.payload equals' followed by an 'optional' text field. The 'Handling' dropdown is set to 'all messages'. The 'Name' field is empty.

On the right side of the interface, there is a 'debug' console window. It shows a list of messages with the following content:

```
9/26/2019, 3:01:31 PM node: 36af48b4.058dd8
msg.payload : number
2019-09-26T07:01:34.101Z

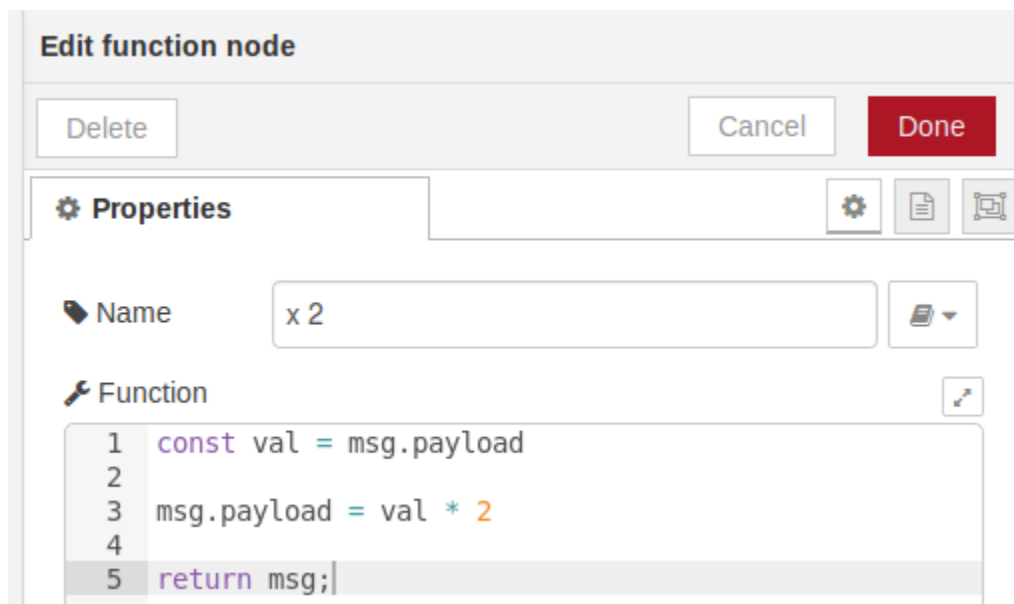
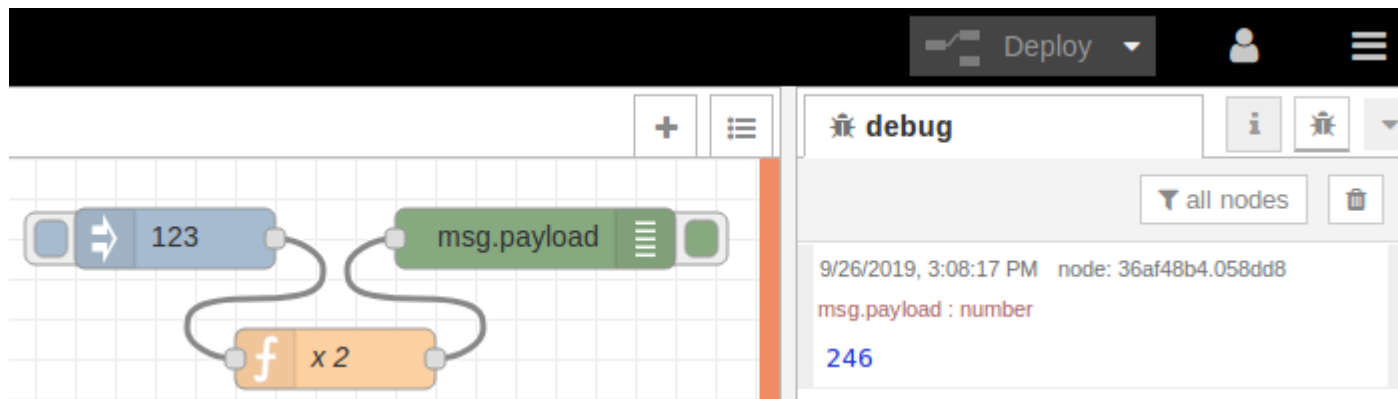
9/26/2019, 3:01:36 PM node: 36af48b4.058dd8
msg.payload : number
2019-09-26T07:01:34.101Z

9/26/2019, 3:01:41 PM node: 36af48b4.058dd8
msg.payload : number
2019-09-26T07:01:34.101Z
```

- trigger
 - 可以收到 msg 後, 等待一段時間後再送給下個 node
 - 可以將 msg 每間隔一段時間重覆送給下個 node
- Ref: [example/exp2.json](#)



常用 node 介紹



- **function**: 可寫自己的邏輯在裡面, 開發需要的功能
- Ref: [example/exp3.json](#)



context

- 用來儲存資訊的一種方法
- Context 分成
 - **Node**: 只有這個 node 可存取
 - **Flow**: 只有這個 flow (tab) 內的 node 可存取
 - **Global**: 所有 flow 內的 node 都可以存取
- (預設) context 內容會在**系統重啟**情況下被清空, 若要避免被清空需要其他設定或是 plug-in
- 若 context get 不到, 值會是 **undefined**



Node scope context

Edit function node

Delete

Cancel

Done

Properties

Settings

File

View

Name

Function

```
1 const val = msg.payload
2
3 count = context.get("count")
4 if (count === undefined)
5 {
6     count = 0
7 }
8
9 latest count = count + val
10
11 context.set("count", latest count)
12
13 msg.payload = latest count
14
15 return msg;
```

debug

Info

Logs

▼

all nodes

10/1/2019, 4:11:18 PM node: 6d3a0c5b.4ea4e4

msg.payload : number

3

10/1/2019, 4:11:19 PM node: 6d3a0c5b.4ea4e4

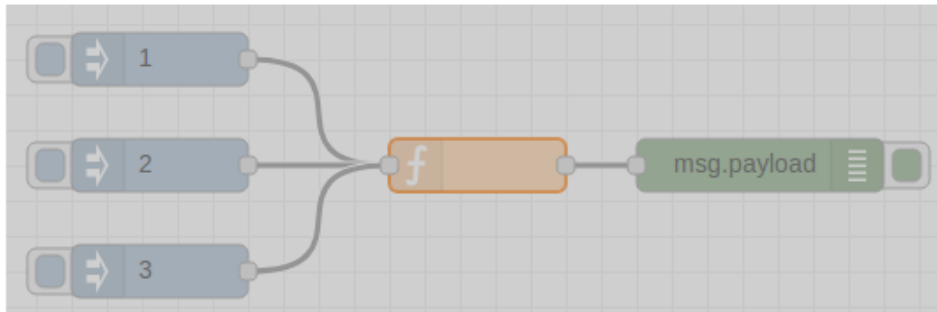
msg.payload : number

4

10/1/2019, 4:11:21 PM node: 6d3a0c5b.4ea4e4

msg.payload : number

6



- 使用 predefined variables **context**
- 重新 deploy 會清空 node context
- Ref: [example/exp4_node-context.json](#)



Flow scope context

Properties

Name: set flow context

Function

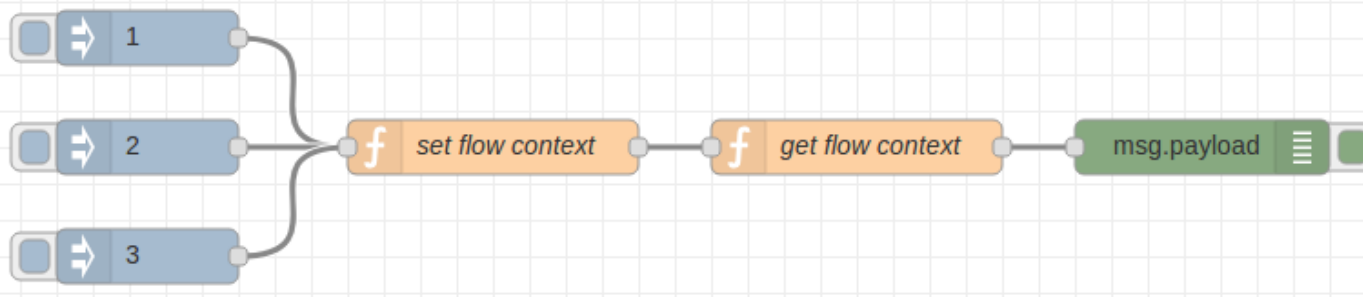
```
1 const val = msg.payload
2
3 flow.set("inject num", val)
4
5 msg.payload = "no inject num info"
6
7 return msg;
```

Properties

Name: get flow context

Function

```
1 const val = flow.get("inject num")
2
3 msg.payload = val
4
5 return msg;
```



debug

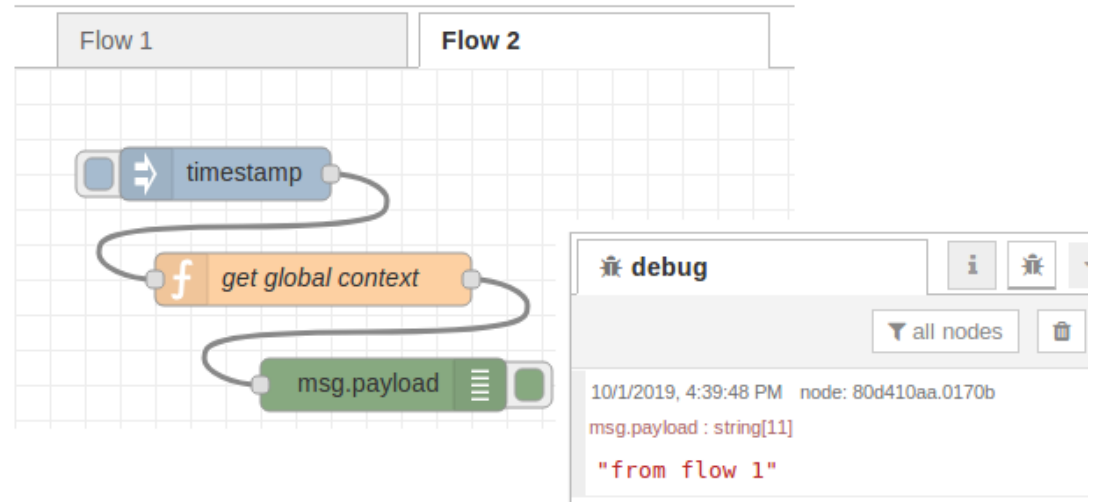
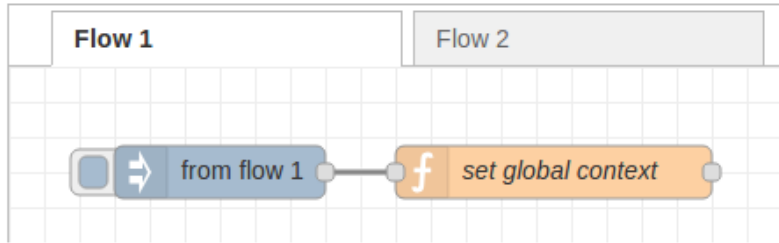
all nodes

10/1/2019, 4:25:14 PM	node: 6d3a0c5b.4ea4e4	msg.payload : number
1		
10/1/2019, 4:25:15 PM	node: 6d3a0c5b.4ea4e4	msg.payload : number
3		
10/1/2019, 4:25:16 PM	node: 6d3a0c5b.4ea4e4	msg.payload : number
2		

- 使用 predefined variables **flow**
- Ref: [example/exp5_flow-context.json](#)



Global scope context



- 使用 predefined variables **global**
- Ref: example/exp6_global-context.json

Name: set global context

Function

```
1 const message = msg.payload
2
3 global.set("flow1 info", message)
4
5 return msg;
```

Name: get global context

Function

```
1 const message = global.get("flow1 info")
2
3 msg.payload = message
4
5 return msg;
```

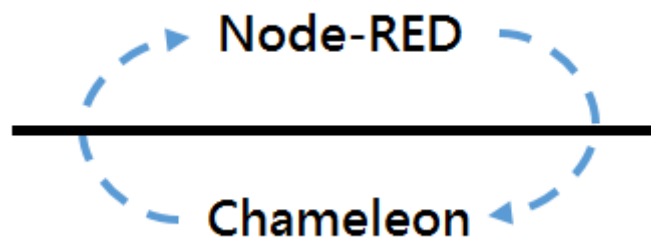


- 課堂練習
- QnA
- 中場休息



Chameleon x Node-RED

- Node-RED 上可收到變色龍所有資料
- 在 Node-RED 處理完的資料可拋回變色龍
- 常用的功能已包成 node, 減少額外開發的時間





前置作業

- Chameleon 頁面新增 Mitsubishi FX5U 機台
 - ip: 貼在 plc 上
 - port: 9600
 - 3E:ASCII
- 點位表: (fx5u.xlsx)

點位	意義	長度	備註
X0	Switch alarm	1	Alarm 點
D602	溫度	1	乘 0.1
D612	濕度	1	乘 0.1
D650	Count	1	每秒加 1
D660	Double count	1	每0.1秒加 1



新增機台-選擇PLC廠牌

☒ 開始新增設備

☒ 選擇廠牌

☐ 選擇類型

☐ 設定連線資料

☐ 測試設備連接

☐ 設定點位表

選擇設備類型

請選擇您要新增的設備類型



上一步

下一步



新增機台-設定連線資料

開始新增設備

選擇廠牌

選擇類型

設定連線資料

測試設備連接

設定點位表

設定 MELSEC 連線資料

請輸入您要連線的設備資料



FX5U

Protocol

3E:ascii

設備 ID

FX5U

設備名稱

連線

TCP

10

設定 MELSEC 連線資料

請輸入您要連線的設備資料



FX3U

連線類型

☒ TCP ☐ UDP

TCP Host

192.168.20.78

TCP Port

9600

連線間隔毫秒 (pollingPeriodMS)

1000

連線時間 (requestTimeoutMS)

500

上一步

下一步



新增機台-連線測試

✓ 開始新增設備

✓ 選擇廠牌

✓ 選擇類型

✓ 設定連線資料

✓ 測試設備連接

✓ 設定點位表

測試 MELSEC 設備連接

您可以「略過」此步驟直接設定點位表，或輸入以下資料來進行「測試」



FX5U

Command ☒ Read Word ☐ Read Bit

Register Type

D

Address

602

Quantity

1

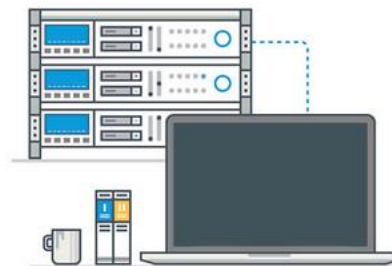
上一步

測試

略過

測試 MELSEC 設備連接

您可以「略過」此步驟直接設定點位表，或輸入以下資料來進行「測試」



測試成功，設備和主機之間的連線正常

上一步

重新測試

下一步，設定點位表



新增機台-上傳點位表

✓ 開始新增設備

✓ 選擇廠牌

✓ 選擇類型

✓ 設定連線資料

✓ 測試設備連接

✓ 設定點位表

設定點位表

請選擇您要建立點位表的方式



FX5U

上傳點位表

✓ 開始新增設備

✓ 選擇廠牌

✓ 選擇類型

✓ 設定連線資料

✓ 測試設備連接

✓ 設定點位表

設定點位表

請選擇您要建立點位表的方式



FX5U

上傳點位表

點位表 FX5U 已上傳 ✓

上一步

完成



新增機台-完成



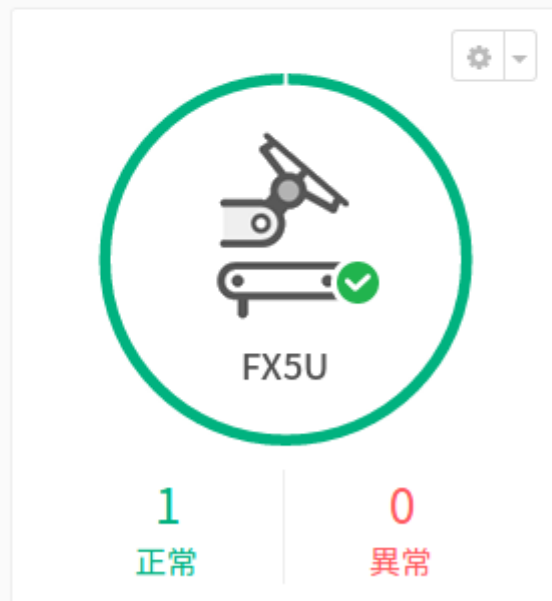
生產監控

生產日誌

檔案下載

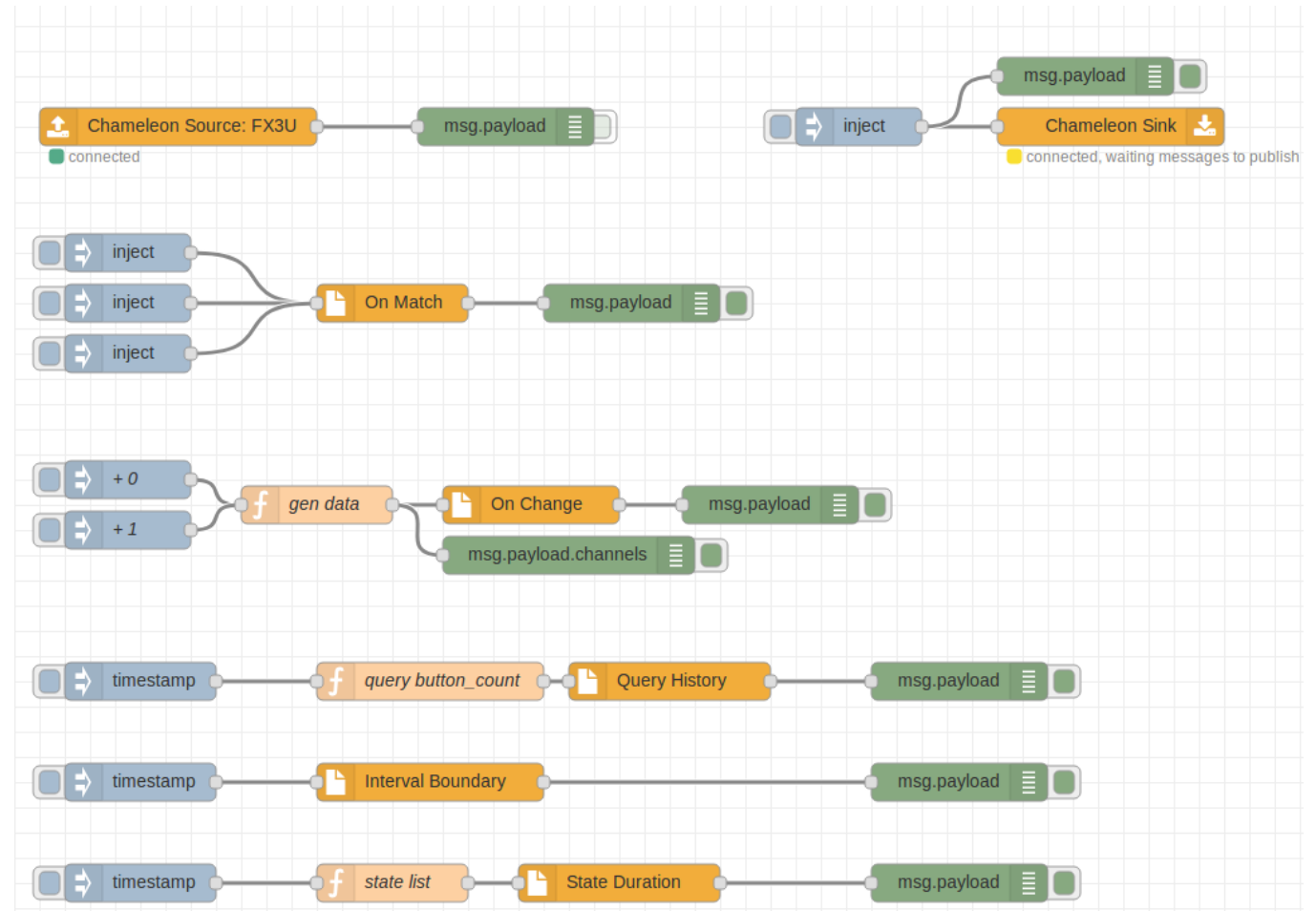
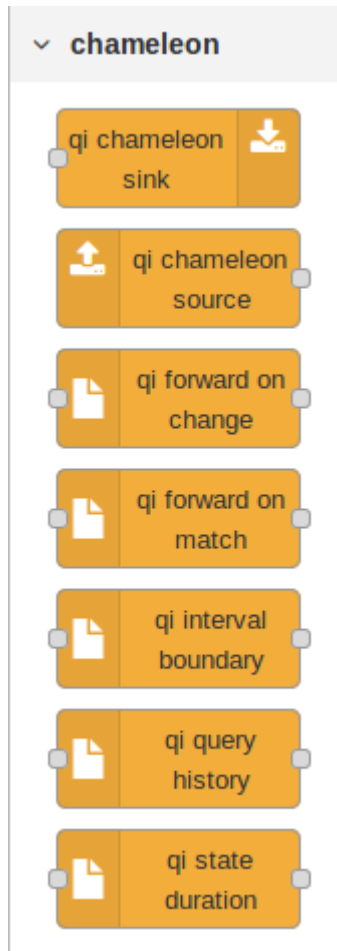
生產監控

產線 1





Node with Chameleon





qi chameleon source

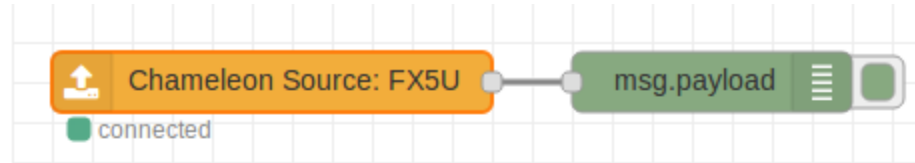
- 從 chameleon 接收資料
- 可在 node 編輯頁面設定 device id 或 “+” 接收所有 device
- 資料輸出格式

```
{  
  "deviceId": "string",  
  "at": {  
    "sec": integer,  
    "us": integer  
  },  
  "channels": {  
    "channel": value  
  }  
}
```

```
{  
  "deviceId": "FX5U",  
  "at": {  
    "sec": 1569911173,  
    "us": 0  
  },  
  "channels": {  
    "D602": 25.7,  
    "D612": 57.5,  
    "D650": 101,  
    "D660": 1006,  
    "X0": 0  
  }  
}
```



qi chameleon source



Edit qi-chameleon-source node

Delete Cancel Done

Properties

Device ID FX5U 設定 device id 來接收 FX5U 的資料

Name Name

- Ref: [example/exp7_chameleon-source.json](#)

debug

all nodes

10/2/2019, 2:02:49 PM node: c2495404.c0ba5

msg.payload : Object

- object
 - deviceId: "FX5U"
- at: object
 - sec: 1569996180
 - us: 0
- channels: object
 - D602: 26.2
 - D612: 55.3
 - D650: 20620
 - D660: 9588
 - X0: 0



qi chameleon sink

- 將資料拋回 chameleon
- **Chameleon** 需先增加虛擬設備
- 資料輸入格式

```
{  
  "deviceId": "string",  
  "at": {  
    "sec": integer,  
    "us": integer  
  },  
  "channels": {  
    "channel": value  
  }  
}
```

```
{  
  "deviceId": "test_chameleon_sink",  
  "at": {  
    "sec": 1564042800,  
    "us": 0  
  },  
  "channels": {  
    "float_channel": 1.23,  
    "integer_channel": 10,  
    "string_channel": "chameleon_no1"  
  }  
}
```



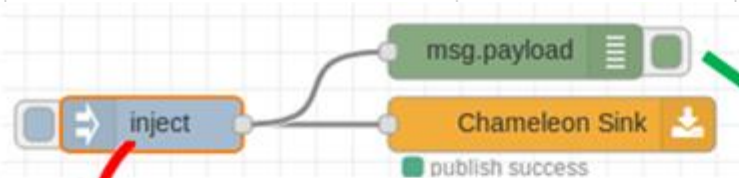
qi chameleon sink



test_chameleon_sink

告警狀態 即時狀態

#	點位 ID	點位名稱	數值類型	數值內容
👁	float_channel		number	1.59
👁	integer_channel		number	10
👁	string_channel		string	chameleon_no1



- Ref: example/exp8_chameleon-sink.json
- test_chameleon_sink_virtual.xlsx

Edit inject node > JSON editor

Cancel

Done

format JSON

```
1 {  
2   "deviceId": "test chameleon sink",  
3   "at": {  
4     "sec": 1569996660,  
5     "us": 0  
6   },  
7   "channels": {  
8     "float channel": 1.59,  
9     "integer channel": 10,  
10    "string channel": "chameleon no1"  
11  }  
12 }
```

debug

all nodes

```
10/2/2019, 3:58:08 PM node: bebf3d28.1177c  
msg.payload : Object  
  object  
    deviceId: "test_chameleon_sink"  
    at: object  
      sec: 1569996660  
      us: 0  
    channels: object  
      float_channel: 1.59  
      integer_channel: 10  
      string_channel:  
        "chameleon_no1"
```




新增虛擬設備-選擇 VIRTUAL

生產監控



選擇設備類型

請選擇您要新增的設備類型



上一步

下一步



新增虛擬設備-設定設備 ID

✓ 開始新增設備

✓ 選擇廠牌

✓ 選擇類型

✓ 設定連線資料

✓ 設定點位表

設定 VIRTUAL 連線資料

請輸入您要連線的設備資料



test-chameleon-sink

設備 ID

設備名稱

上一步

下一步



新增虛擬設備-上傳點位表

✓ 開始新增設備

✓ 選擇廠牌

✓ 選擇類型

✓ 設定連線資料

✓ 設定點位表

設定點位表

請選擇您要建立點位表的方式



test-chamel
sink

上傳點位表

✓ 開始新增設備

✓ 選擇廠牌

✓ 選擇類型

✓ 設定連線資料

✓ 設定點位表

設定點位表

請選擇您要建立點位表的方式



test_chameleon_!

上傳點位表

點位表 test_chameleon_sink 已上傳 ✓

上一步

完成



新增虛擬設備-完成



生產監控

生產日誌

檔案下載

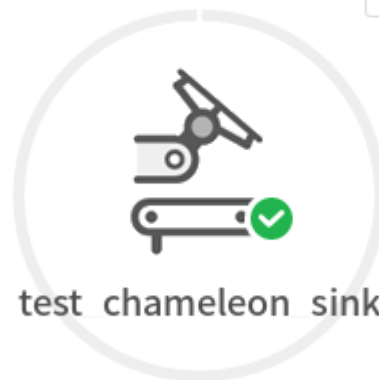
生產監控

產線 1



1
正常

0
異常



-
正常

-
異常



qi on match

- 資料 filter
- 需在 node 編輯頁面設定 Schema
- Schema 格式 (example: qi-on-match.schema)
 - \$schema
 - type
 - required
 - properties



qi on match

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": [
    "deviceId",
    "at",
    "channels"
  ],
  "properties": { ...
  }
}
```



qi on match

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": [ ],
  "properties": {
    "deviceId": {
      "type": "string"
    },
    "at": {
      "type": "object",
      "required": [
        "sec",
        "us"
      ],
      "properties": {
        "sec": {
          "type": "integer",
          "minimum": 0
        },
        "us": {
          "type": "integer",
          "minimum": 0,
          "maximum": 999999
        }
      }
    },
    "channels": [ ]
  }
}
```

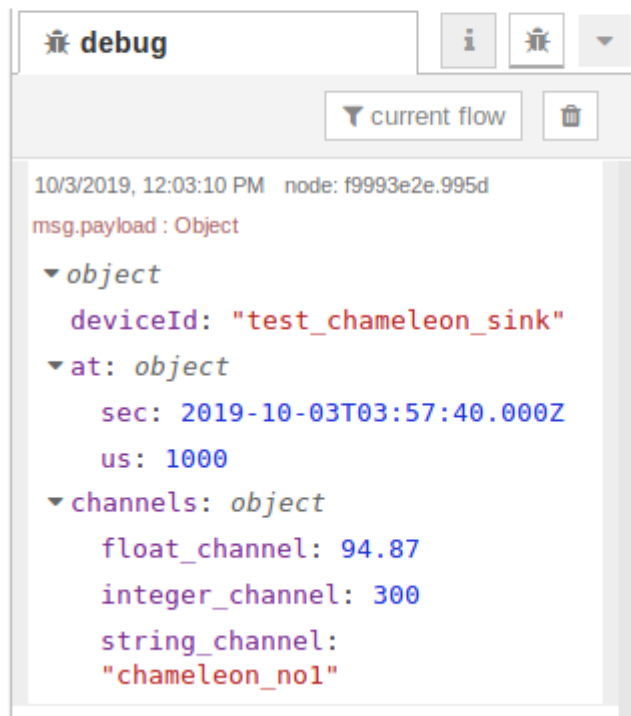
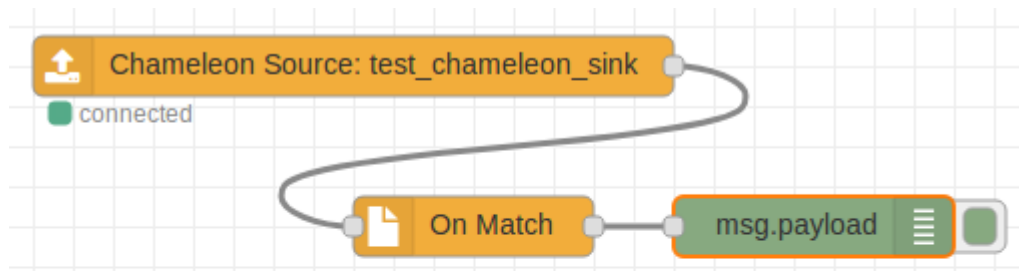


qi on match

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": [
  ],
  "properties": {
    "deviceId": {
      "type": "string"
    },
    "at": {
    },
    "channels": {
      "type": "object",
      "required": [
        "float_channel",
        "integer_channel",
        "string_channel"
      ],
      "properties": {
        "float_channel": {
          "type": "number"
        },
        "integer_channel": {
          "type": "number"
        },
        "string_channel": {
          "type": "string"
        }
      }
    }
  }
}
```




qi on match



- Ref: example/exp9_on-match.json
- qi-on-match.schema



qi on match

Edit qi-forward-on-match node

Delete Cancel Done

Properties

Schema {"\$schema": "http://json-schema.org/draft-07/schema#"} ...

☒ Remove additional

Name

經過 on-match node 後
不在 properties::channels::required 的
channel 會被過濾掉

Edit qi-forward-on-match node > JSON editor

Cancel Done

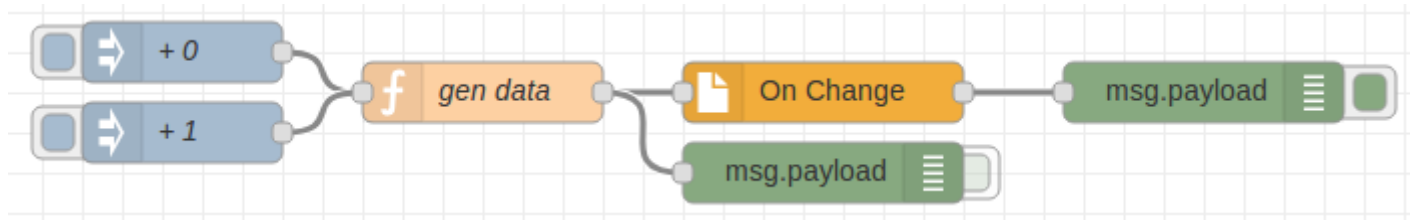
format JSON

```
1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "type": "object",
4   "required": [
5     "deviceId",
6     "at",
7     "channels"
8   ],
9   "properties": {
10    "deviceId": {
11      "type": "string"
12    },
13    "at": {
14      "type": "object",
15      "required": [
16        "sec",
17        "us"
18      ],
19      "properties": {
20        "sec": {
21          "type": "integer",
22          "minimum": 0
23        },
24        "us": {
25          "type": "integer",
26          "minimum": 0,
27          "maximum": 999999
28        }
29      }
30    },
31    "channels": {
32      "type": "object",
33      "required": [
34        "float channel",
35        "integer channel",
36        "string channel"
37      ],
38      "properties": {
39        "float channel": {
40          "type": "number"
41        },
```



qi on change

- Channel 的 value 值改變才會往下送



Ref: [example/exp10_on-change.json](#)

debug

all nodes

10/3/2019, 2:04:53 PM node: 1726e160.7a587f
msg.payload : Object

- object
 - deviceId: "dev"
 - at: object
 - channels: object
 - accumulate_value: 0

10/3/2019, 2:04:55 PM node: 1726e160.7a587f
msg.payload : Object

- object
 - deviceId: "dev"
 - at: object
 - channels: object
 - accumulate_value: 1



qi query history

- 從 database query 資料
- Query string (以 msg.query 為主)
 1. 編輯 node 內的 **Query** 欄位
 2. 或是在之前的 node 準備好 **msg.query**
- Query string 格式: **InfluxQL**
 - the InfluxDB SQL-like query language
 - https://docs.influxdata.com/influxdb/v1.7/query_language/



qi query history

The screenshot displays the qi query history interface. At the top, a workflow consists of four nodes: a blue 'timestamp' node, an orange 'build query string' node, a yellow 'Query History' node, and a green 'msg.payload' node. Below the workflow, the 'Edit function node' dialog is open for the 'build query string' node. The 'Name' field is set to 'query button_count'. The 'Function' field contains the following SQL query:

```
1 msg.query = `  
2   SELECT *  
3   FROM FX5U  
4   LIMIT 2  
5   `  
6  
7   return msg;
```

On the right side, the 'debug' console shows the execution results for the 'Query History' node. The output is an array with two elements:

```
10/3/2019, 2:18:15 PM node: 80b6e92c.449b68  
msg.payload : array[2]  
  array[2]  
    0: object  
      time: "2019-10-03T02:31:00.003Z"  
      D602: 24.4  
      D612: 59.9  
      D650: 3653  
      D660: 36526  
      X0: 0  
    1: object
```

- Ref: [example/exp11_query-history.json](#)



qi interval boundary

- 取得某段的開始及結束時間
- node 編輯頁面設定
 - Timezone: UTC/Local Time
 - Interval Duration: default is 86400 (1 天)
 - Shift interval: 輸入整數, 以當下時間為基礎做 interval 的 shift, 若 interval duration 是 86400, 輸入 “-2” 會取得前天的開始/結束時間



qi interval boundary

timestamp Interval Boundary msg.payload

Edit qi-interval-boundary node

Delete Cancel Done

Properties

Timezone Local Time

Interval Duration 86400

Shift Interval 0

Name Name

debug

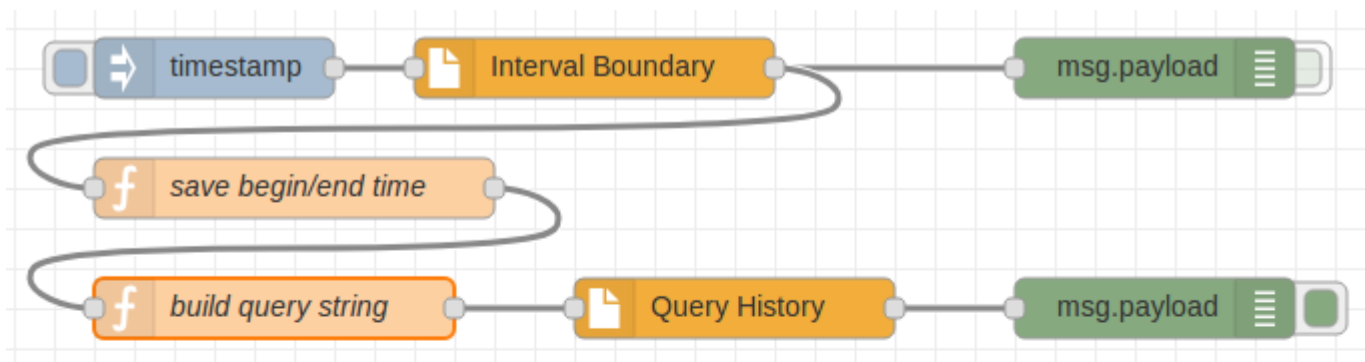
10/2/2019, 4:30:12 PM node: 6c84efba.60f3c
msg.payload : Object

```
{ begin: "2019-10-01T16:00:00.000Z", end: "2019-10-02T16:00:00.000Z" }
```

- Ref: [example/exp12_interval-boundary.json](#)



interval boundary + query history example



result of Interval Boundary

10/4/2019, 4:29:07 PM node: 1798a542.6dab9b

msg.payload : Object

```
▶ { begin:
  "2019-10-03T16:00:00.000Z", end:
  "2019-10-04T16:00:00.000Z" }
```

debug

all nodes

10/4/2019, 4:26:35 PM node: 9fc5215a.da1d6

msg.payload : array[5]

▼ array[5]

▶ 0: object

▶ 1: object

▼ 2: object

time:

"2019-10-04T07:45:00.000Z"

D612: 62

▶ 3: object

▶ 4: object

- Ref: [example/exp13_query-history-example.json](#)



interval boundary + query history example

Edit function node

Delete

Cancel

Done

Properties

Name

save begin/end time

Function

```
1 const begin = msg.payload.begin
2 const end = msg.payload.end
3
4 msg.begin time = begin
5 msg.end time = end
6
7 return msg;
```

Edit function node

Delete

Cancel

Done

Properties

Name

build query string

Function

```
1 const begin time = msg.begin time
2 const end time = msg.end time
3
4 msg.query = `
5   SELECT D612
6   FROM FX5U
7   WHERE time >= '${begin time.toISOString()}'
8   AND time < '${end time.toISOString()}'
9   LIMIT 5
10 `
11
12 return msg;
```



qi state duration

- 取得每個 state 的累積時間(單位: 毫秒 **millisecond**)
 - 設定要算的 state name
 - 資料輸入格式:
 - msg.payload: influxDB query result
 - msg.since: (optional) Date
 - msg.until: (optional) Date
- ```
[
 {
 "time": <date>,
 "<channel>": <value>
 },
]
```

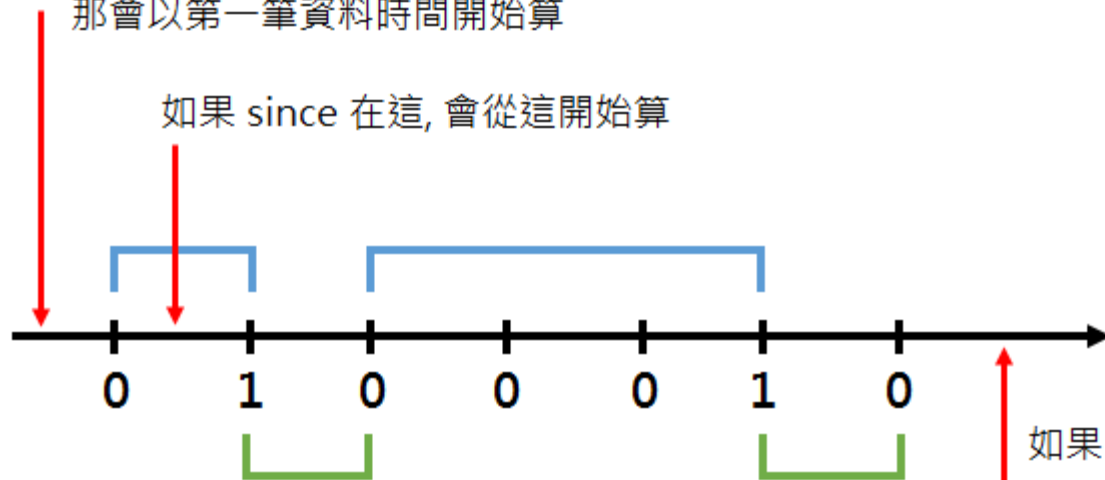


# qi state duration

- msg.payload: 允許重覆連續的 state
- msg.since: 若沒給, 會以 payload 第一筆資料的時間開始算
- msg.until: 若沒給, 會以 payload 最後一筆資料的時間為止

如果 since 小於第一筆資料時間, 或是沒給 since  
那會以第一筆資料時間開始算

如果 since 在這, 會從這開始算



until 時間不能小於最後一筆資料的時間



# qi state duration

Diagram showing a sequence of nodes: timestamp, state list, State Duration, and msg.payload.

**Edit qi-state-duration node**

Delete Cancel Done

**Properties**

State Name: operation

Name: Name

**debug**

10/3/2019, 3:34:35 PM node: 39834e25.7bfdba  
msg.payload : array[2]

```
▼ array[2]
 ▼ 0: object
 state: 0
 duration: 255000
 ▼ 1: object
 state: 1
 duration: 44000
```

**Function**

```
1 msg.payload = [
2 {
3 "time":new Date("2019-10-02T02:50:35.000Z"),
4 "operation":0
5 }, // 0:46 -> since: 2019-10-02T02:50:54.000Z -> 0:27
6 {
7 "time":new Date("2019-10-02T02:51:21.000Z"),
8 "operation":1
9 }, // 0:26
10 {
11 "time":new Date("2019-10-02T02:51:47.000Z"),
12 "operation":0
13 }, // 0:42
14 {
15 "time":new Date("2019-10-02T02:52:29.000Z"),
16 "operation":1
17 }, // 0:9
18 {
19 "time":new Date("2019-10-02T02:52:38.000Z"),
20 "operation":0
21 }, // 0:6
22 {
23 "time":new Date("2019-10-02T02:52:44.000Z"),
24 "operation":1
25 }, // 0:9
26 {
27 "time":new Date("2019-10-02T02:52:53.000Z"),
28 "operation":0
29 } // 3:0
30]
31
32 msg.since = new Date("2019-10-02T02:50:54.000Z")
33 msg.until = new Date("2019-10-02T02:55:53.000Z")
34
35 return msg;
```

- Ref: example/exp14\_state-duration.json

