HITACHI
Inspire the Next

# Debugging and Verifying Flows

2 July 2019

**Kunihiko Toumura**

Research & Development Group
Hitachi, Ltd.

# 1. Introduction

HITACHI
Inspire the Next

Ideas and status of debugging and verification functions.

- Extension ideas
  - Stepwise execution, Breakpoint

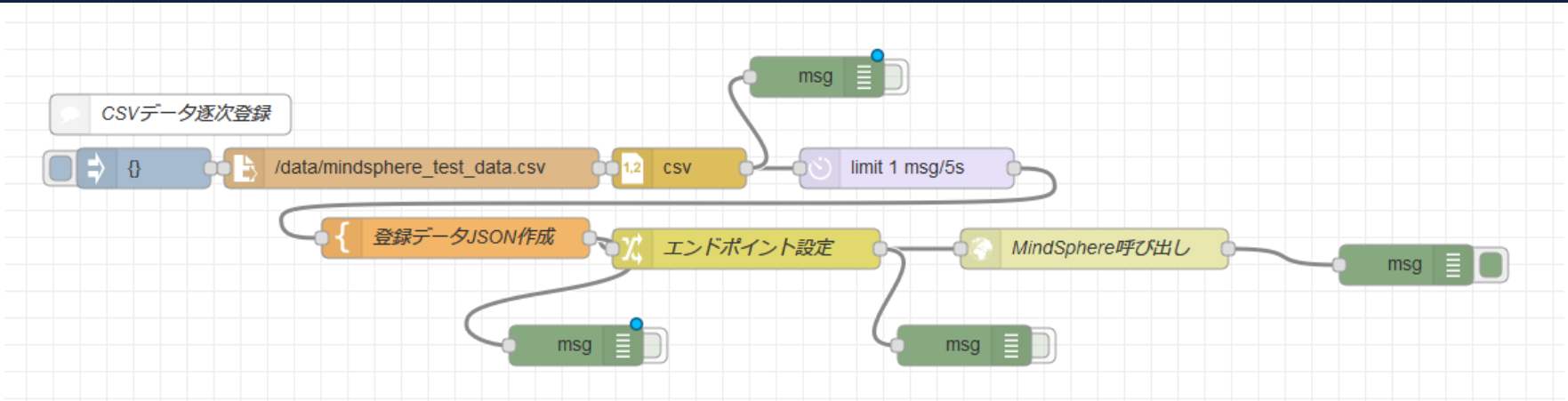
- Current status of our implementation
  - Flow testing
  - Flow linter

# 2-1. Debugging function: Breakpoint, inspection and stepwise execution

HITACHI
Inspire the Next

Current issues:

- Difficult to know which node causes error in a flow, especially in a long flow. To put debug nodes on every connections is current workaround for this.



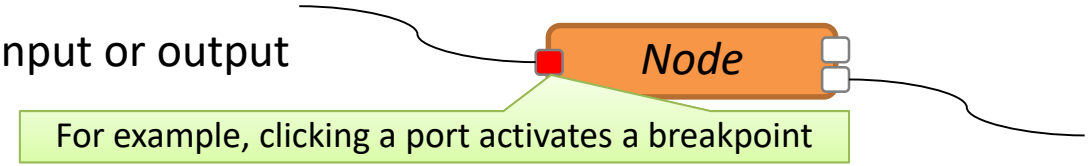

Following function may improve development productivity:

- Breakpoint
- Inspection of variables
- Resuming execution

# 2-2. Function Details

HITACHI
Inspire the Next

- Breakpoint
  - setting a breakpoint on node input or output




  - when a message reaches any breakpoints, the runtime suspends execution.
    - suspending not only receiving message handler, but also all other processing.

- Inspection of variables.
    - view (and update) receiving message.
    - view (and update) contexts, other global variables of the flow.
    - (view (and update) other messages,  history of messages, etc.)

- Resuming execution
  - normal resuming
  - initialize the flow and re-execute
  - etc.

# 2-3. Discussion

HITACHI
Inspire the Next

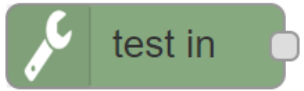
- UI implementation
  - In Node-RED flow editor:
    - too complicated for novice user?
  - Combine with general JavaScript source code debugger (e.g. Visual Studio Code)
    - how to handle a flow information?

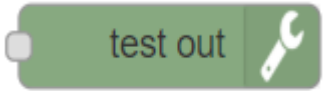
# 3-1. Debugging function 2: Flow testing

HITACHI
Inspire the Next

- Flow testing is a function to test original flow operation.
  - Node-RED user can test flows without programming.
  - Node-RED developer can test flows with CLI.

- There are two new nodes called Test-in node and Test-out node.



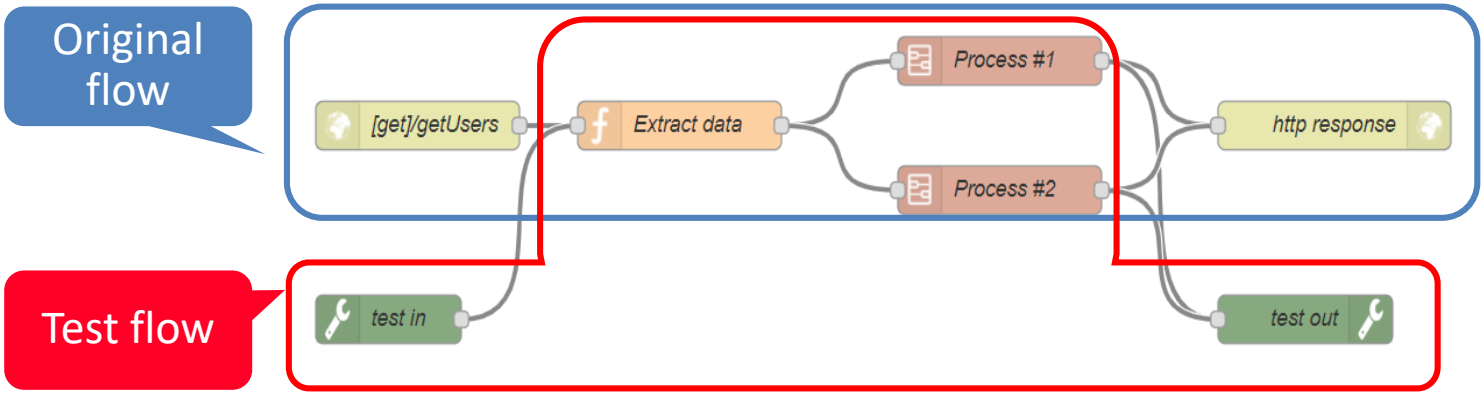

  - Test-in node sends a mock message on behalf of the actual input node such as http-in node.

  - Test-out node receives a message on behalf of the actual output node such as http-response node.

# 3-1. Debugging function 2: Flow testing

HITACHI
Inspire the Next

- Flow example:




- Test-in node has multiple test cases and sends mock messages for cases.
- Test-out node verifies whether the received message is exactly the expected value or not.

# 3-1. Debugging function 2: Flow testing

HITACHI
Inspire the Next

- Run flow testing on the CLI.

```
grunt flow-test --testItem=“./test.json”
```

➢ input (--test-set)
Specify the json file for which the test target "test-in node ID" and "Label" are set.

➢ output

```
    Flow test:77a02011.510cc
      √ Label:Test Case 01 (1006ms)
      √ Label:Test Case 02 (1002ms)


  2 passing (22s)


Done.
```

# 3-2. Current Status

HITACHI
Inspire the Next

- Design: https://github.com/node-red/designs/pull/8/files
- Draft PR: https://github.com/node-red/node-red/pull/2118

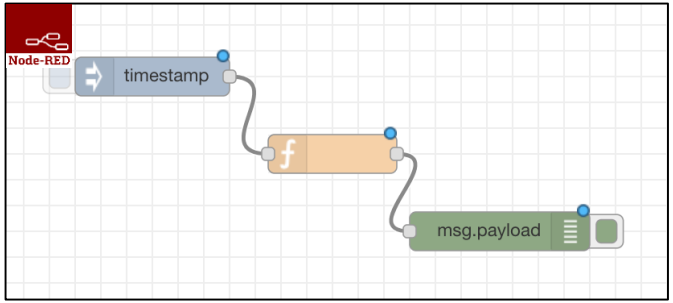
Please comment and review this design.

# 4. Verification Function: Flow Linter

HITACHI
Inspire the Next

Check a flow whether it adhere to certain style guidelines.

## Editor




Flow.json

## Flow lint tool

```
% nrlint flow.json
```

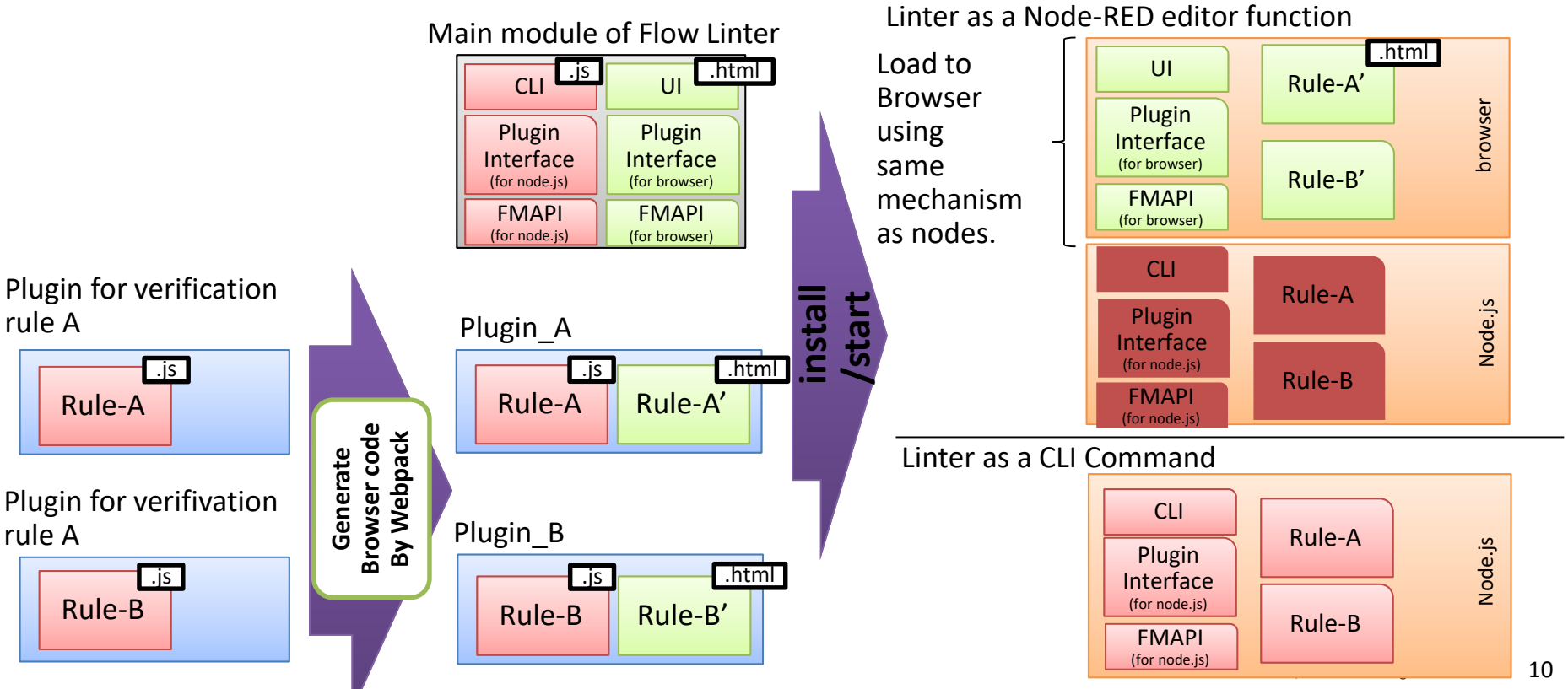
## Report

```
Flow “Flow 1”:
  Warning:
    id "7a40a7b9.91c5b8“:
       Function node has no name.
       (no-empty-function-name)
    id "7a40a7b9.91c5b8“:
       Unused variable: tmp.
       (no-unused-variable)
…
```

# 4-2. Structure of Flow Linter

HITACHI
Inspire the Next

- Use ‘Webpack’ to bundle and generate rule code for browsers.
- Each module are loaded to browser using node-loading mechanisms.



# 4-3. Current Status

HITACHI
Inspire the Next

- Use same validation plug-in in editor and runtime, rewrote Flow Manipulation API so that plug-in code doesn’t use advanced (ES5 or later) JavaScript language functions.

- Creating sample plug-in for Flow Linter and check feasibility of plug-in function and Flow Manipulation API

HITACHI
Inspire the Next

# END

## Debugging and Verifying Flows

2 July 2019

**Kunihiko Toumura**

Research & Development Group
Hitachi, Ltd.

HITACHI

Inspire the Next