HITACHI
Inspire the Next

Graceful Shutdown / Flow Linter

12/19/2019

Kunihiko Toumura

Research and Development Group
Hitachi, Ltd.

1. Graceful Shutdown: Current Status and Next Steps

HITACHI
Inspire the Next

- Current status:
  - Finishing initial implementation of shutdown logic (#2296)
    - Assuming all nodes uses Messaging API, Node-RED runtime can detect whether in-process nodes are existing.
      - If there are nodes which don’t use Node Messaging API, the graceful shutdown may fail (premature shutdown, or longer wait for shutdown)
- Next steps:
  - Test of shutdown logic using (Messaging API-supported) core nodes.
  - Design of setting UI (next page)
- Pull Requests:
  - Messaging API support: https://github.com/node-red/node-red/pull/2402
  - [Draft] Graceful shutdown: https://github.com/node-red/node-red/pull/2296
  - initial proposal of graceful shutdown: https://github.com/node-red/designs/pull/16

2. Graceful Shutdown: Settings UI

To use graceful shutdown function:

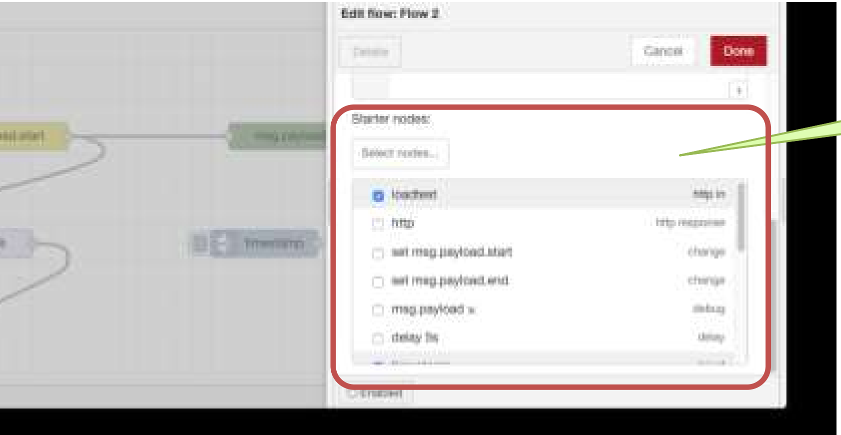
1. Enable the function in ‘settings.js’

```
{…
   gracefulShutdown: true,
   gracefulPeriod: 10000,
…}
```

Default value is false. If true, graceful shutdown process is enabled

Timeout of graceful period (in millisecond)

2. Designate the nodes which initiate flows, such as Inject, HTTP-in, etc. (“starter” nodes). To this, we currently use a configuration panel of the flow.



Starter nodes:

Select nodes...

☑ loadtest http in
□ http http response
□ set msg.payload.start change
□ set msg.payload.end change
□ msg.payload » debug
□ delay 5s delay

UI is similar to Catch/Complete node.

3. Flow Linter: Current Status and Next Steps

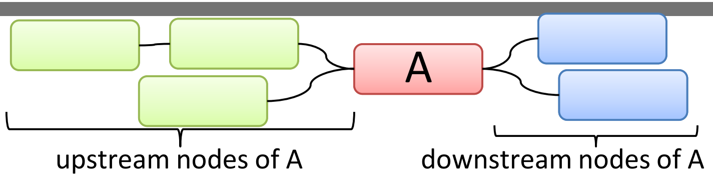
- Current status:
  - Implementing CLI version of flow linter
  - Improving Flow Manipulating API (search, loop detection, etc.)

- Next steps:
  - Design and implementation of Editor-integrated flow linter (- ’20 March)

- Pull Request
  - Add flow linter proposal: https://github.com/node-red/designs/pull/1
  - No PR for codes. Currently using our private repository:
    - https://github.com/node-red-hitachi/node-red-flow-linter
    - When the design of flow linter is approved, I’d like to move this repository to Node-RED repository.
      - https://github.com/node-red/node-red-flow-linter etc.

4. Flow Linter: updating design of Flow Manip. API and rule plug-in

- Added rule-plugin for check matching of HTTP-in and HTTP-response nodes
  - Add search functions to Flow Manip. API
- Plug-in code generation mechanism for browser will be considered after specification of command-line interface version is stabilized.



Current list of Flow Manipulation API (class FlowSet):

Category	function	description
create	FlowSet.**parseFlow**(*parsed flow.json*) -> *FlowSet*	create FlowSet object from flow.json file
read	FlowSet.prototype.**getAllNodesArray**() -> [FMNode]	dump all node as array.
	FlowSet.prototype.**get{Node/Flow/Config/Subflow}**(*node-id*) -> *{FMNode/FMFlow/FMConfig/FMSubflow}*	get {node/flow/config/subflow} by ID
search	FlowSet.prototype.**{next/prev}**(*node-id*) -> *[node-id]*	get nodes which are directly connected on {output/input} ports of the node.
	FlowSet.prototype.**{downstream/upstream}**(*node-id*) -> *[node-id]*	get all nodes which can be followed from the {output/input} port of the node.
	FlowSet.prototype.**connected**(*node-id*) -> *[node-id]*	get all nodes which can be followed from the output or input port of the node. (i.e. fs.downstream(n) + fs.upstream(n))