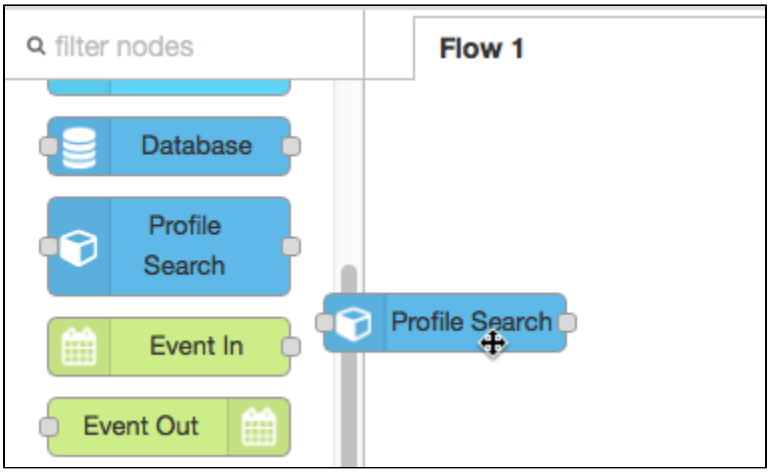
Working with Nodes

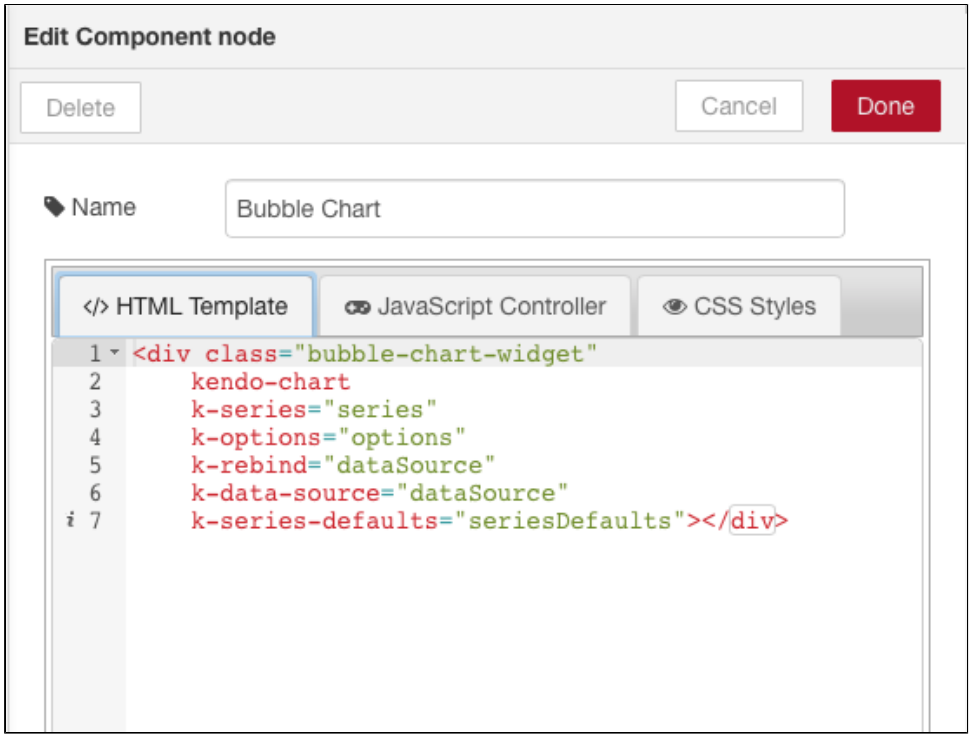
Nodes are selected and chained together to design the flows that define a service that implements a certain policy. To use a node in your flow, drag it into a sheet from the parasoft palette.



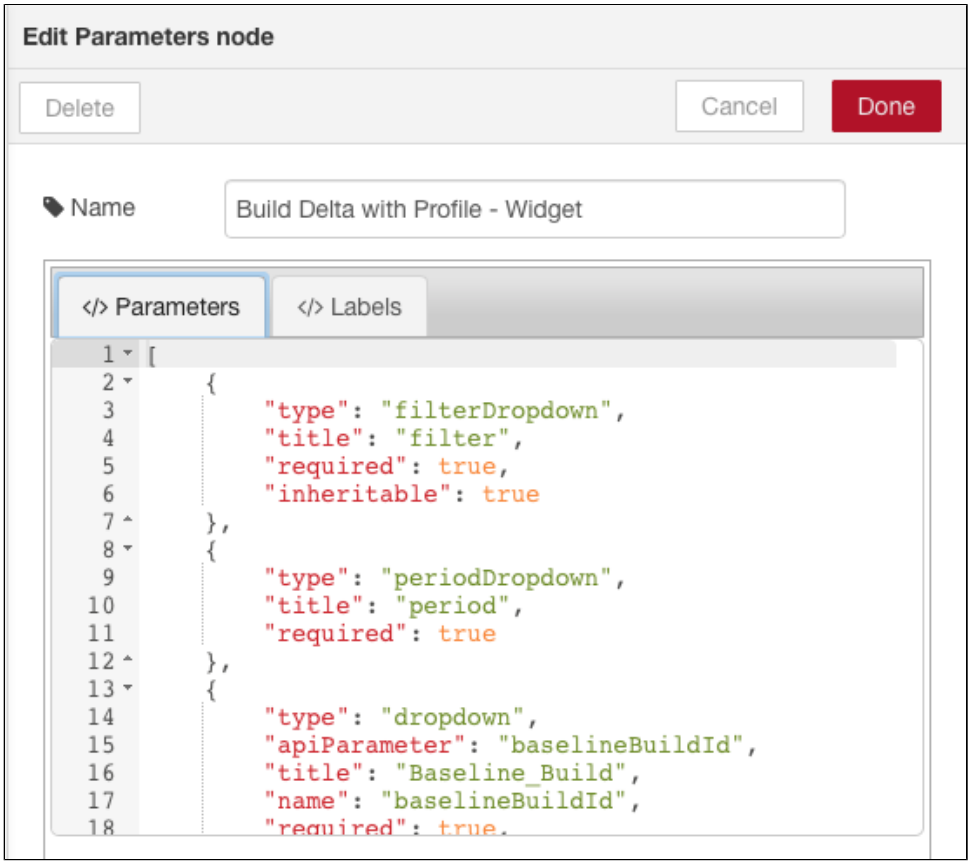
In this section:

Parasoft Nodes

Extension Designer ships with several nodes specific to Parasoft functionality.

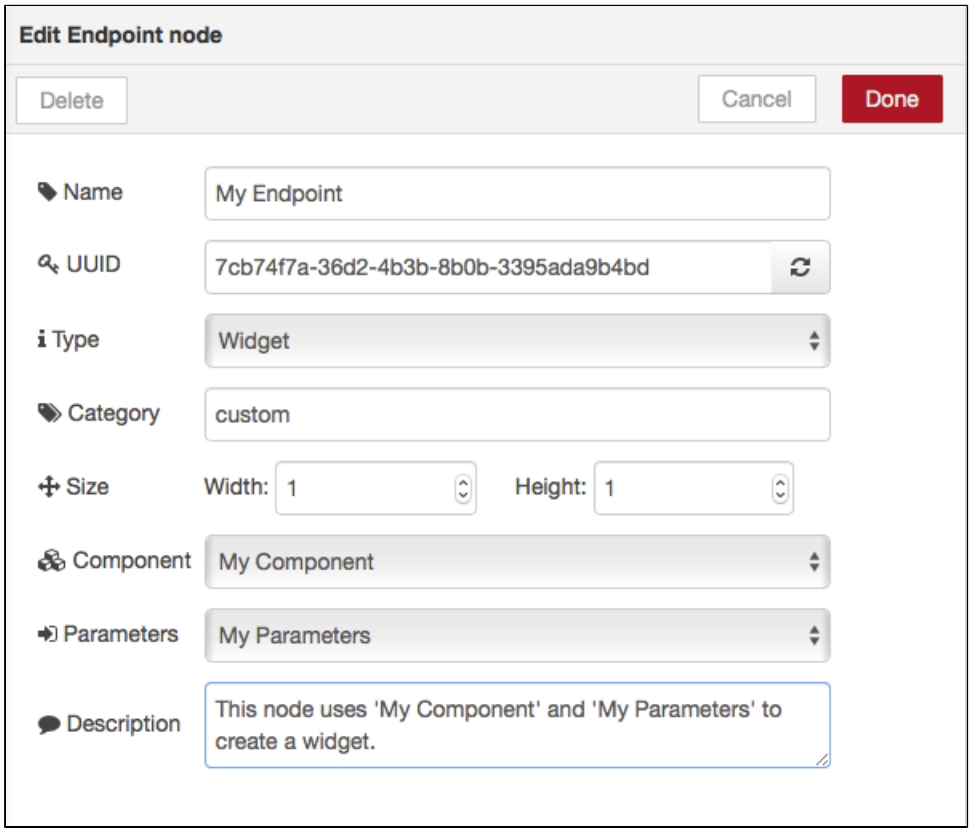
Component	The Component node is used to create reusable front end web components. This node, as well as the Parameters node, is called by the Endpoint node and must be configured to create front end components. You can import pre-programmed sample Component nodes from the library. See Creating Custom DTP Widgets Using Extension Designer for details. 

Parameters	The Parameters node is used to create reusable front end web components. This node, as well as the Component node, is called by the Endpoint node and must be configured to create front end components. You can import pre-programmed sample Parameters nodes from the library. See Creating Custom DTP Widgets Using Extension Designer for details.



Endpoint

The Endpoint node exposes a service endpoint that encapsulates all the functionality necessary to create a full-stack web component. The Endpoint node references the Component and Parameters nodes, which must be configured for this node to function.



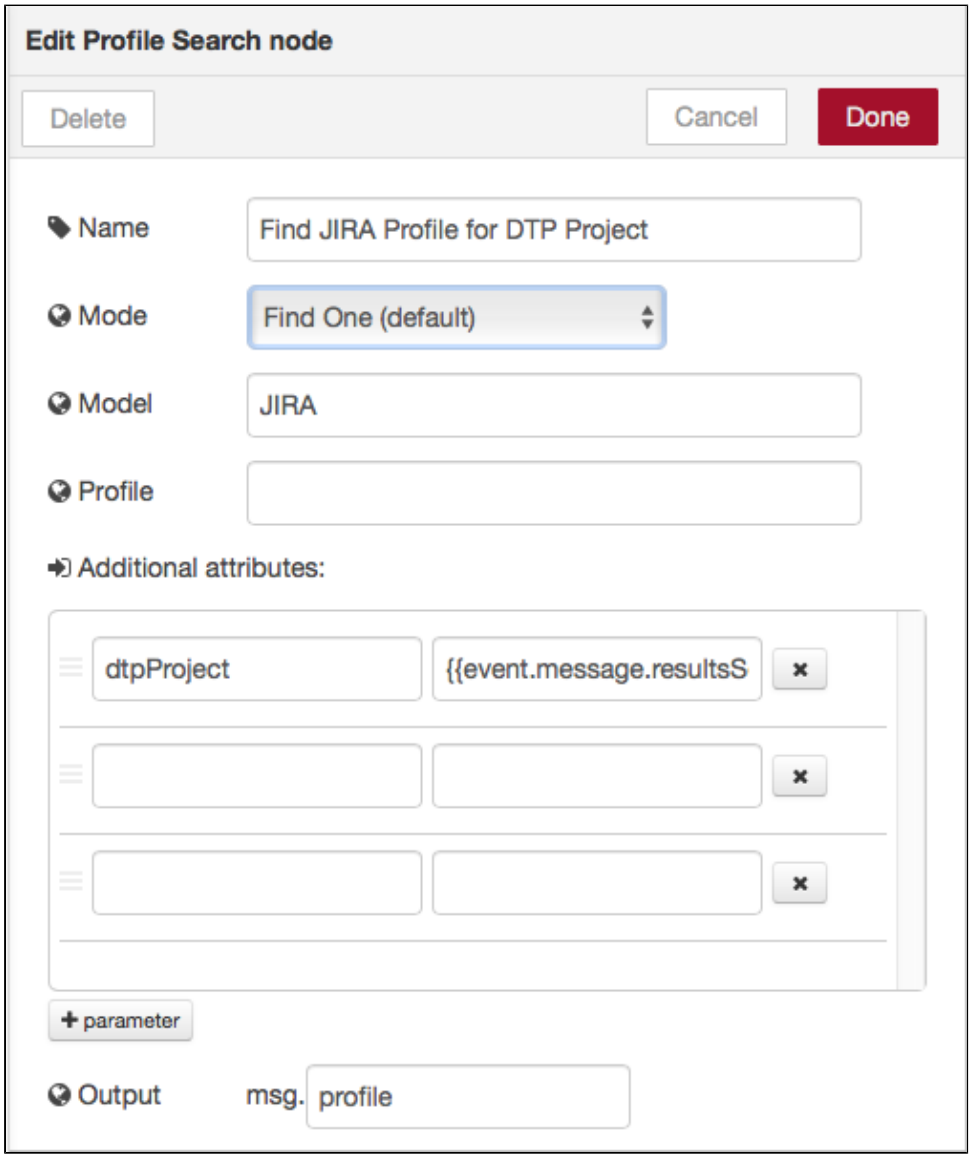
You can implement multiple instances of the same flow when in your services, but if your flows contain Endpoint nodes, then the UUID for each node must be different. Otherwise, only the first instance of the flow will be read. You can manually change the UUID or delete it and re-save the node with an empty UUID field. Saving the empty field will generate a new UUID unique to that instance.

You can configure the Endpoint node to expose the following types:

- **General**: The General endpoint does not have a UI component. It can be used for general request/response scenarios (e.g., triggering a flow via a cURL command).
- **Parameter Values**: The Parameter endpoint type is used to control the contents of drop-down menus in the widgets you are creating. See Configuring Parameter Nodes.
- **Practice**: The Practice endpoint type points to a Component node configured to display a Policy Center Practice widget.
- **Report**: The Report endpoint type points to a Component node configured to display a drill-down report and its associated Parameters node.
- **Widget**: The Widget endpoint type points to a Component node configured to display a widget and its associated Parameters node.

Profile Search

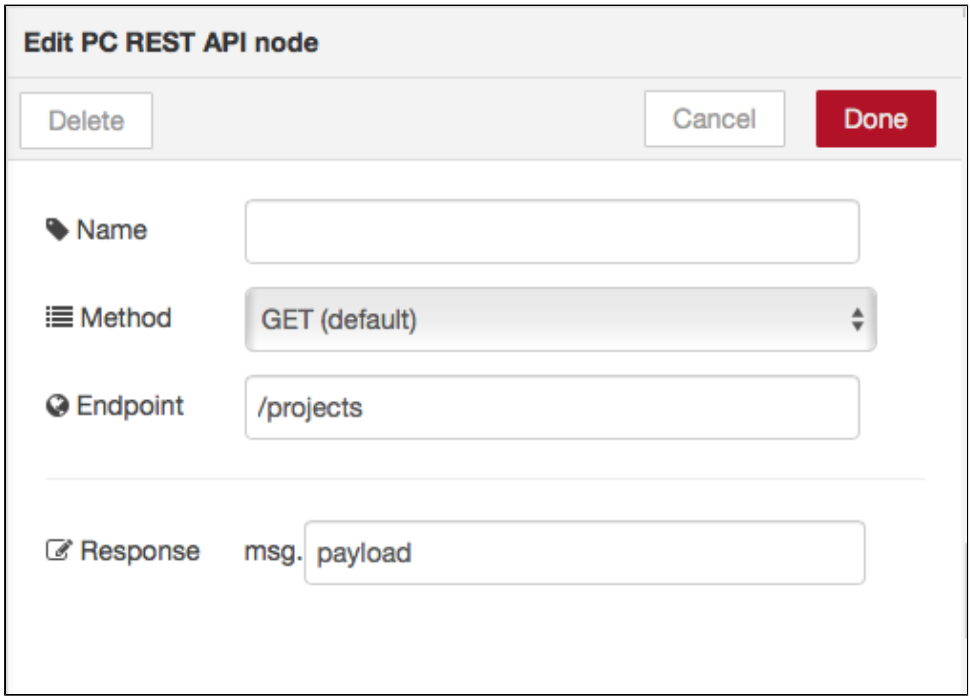
The Profile Search node gathers model profiles to be used in the flow.

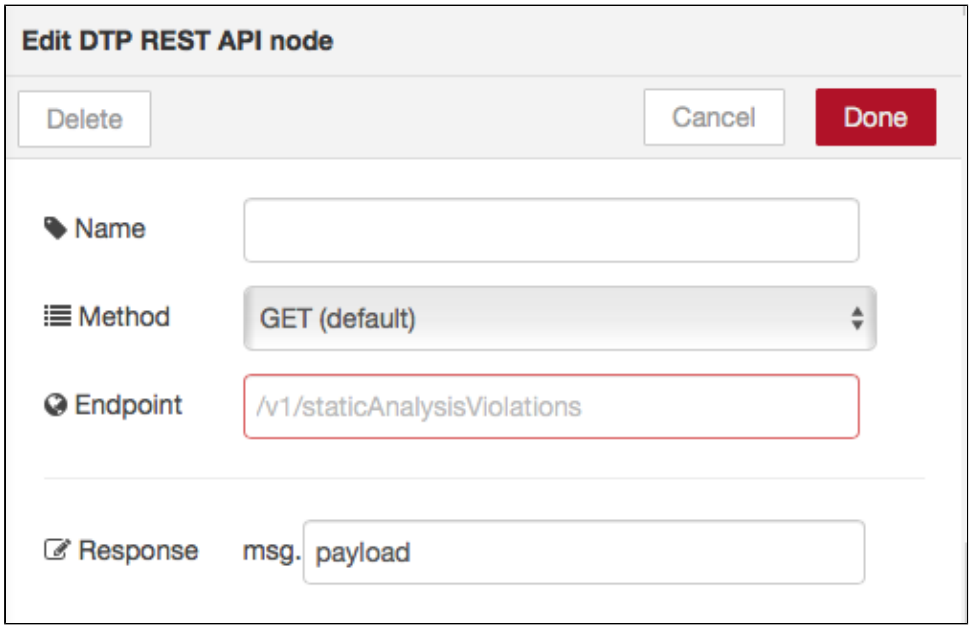
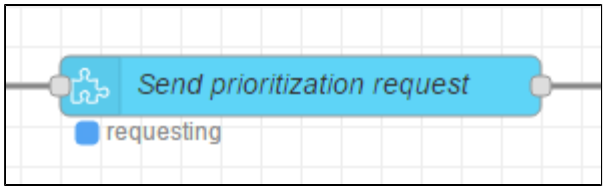
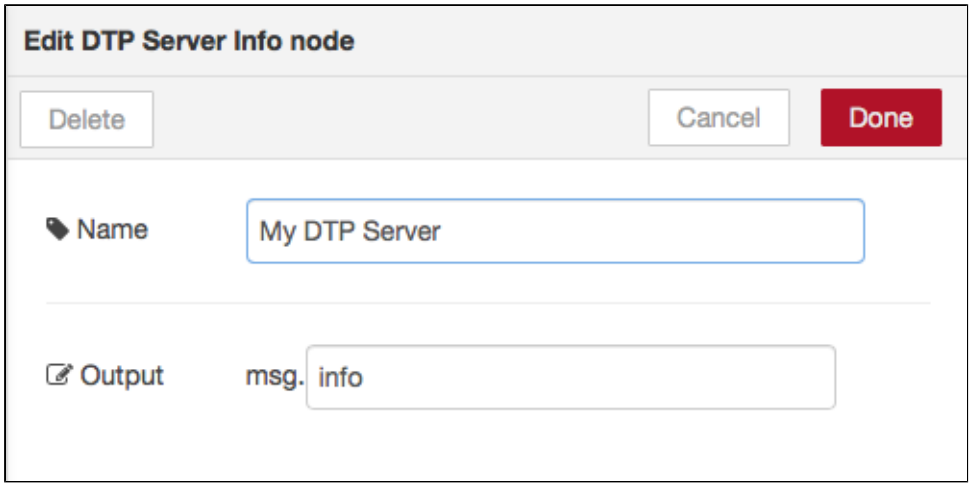


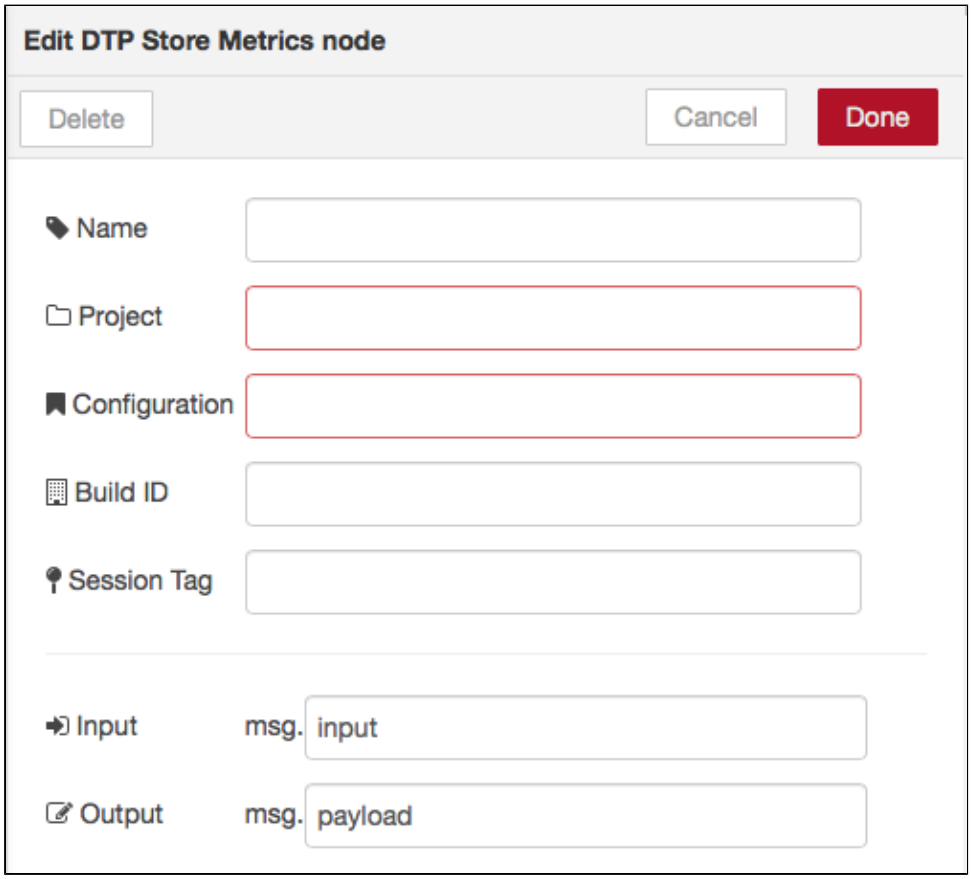
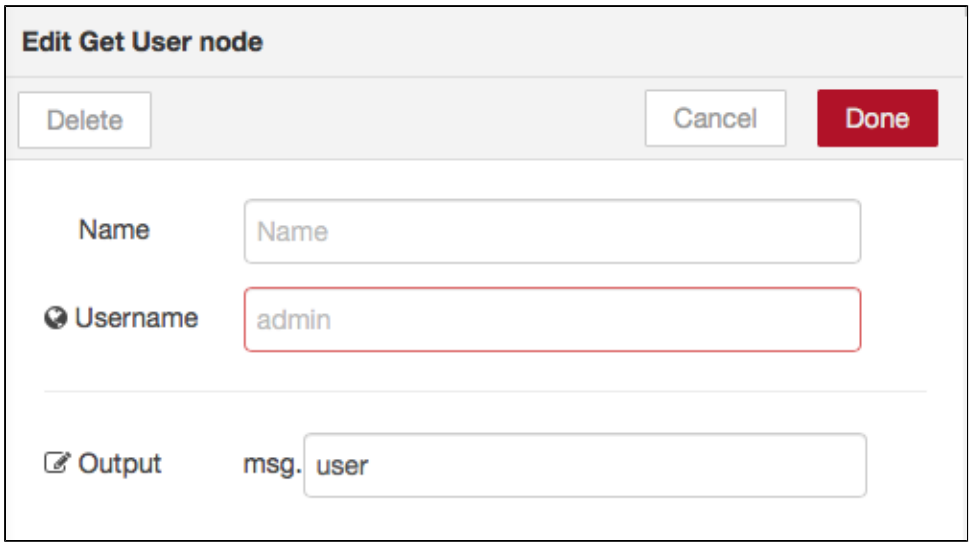
You can perform finer searches by specifying additional search parameters in the Additional attributes section.

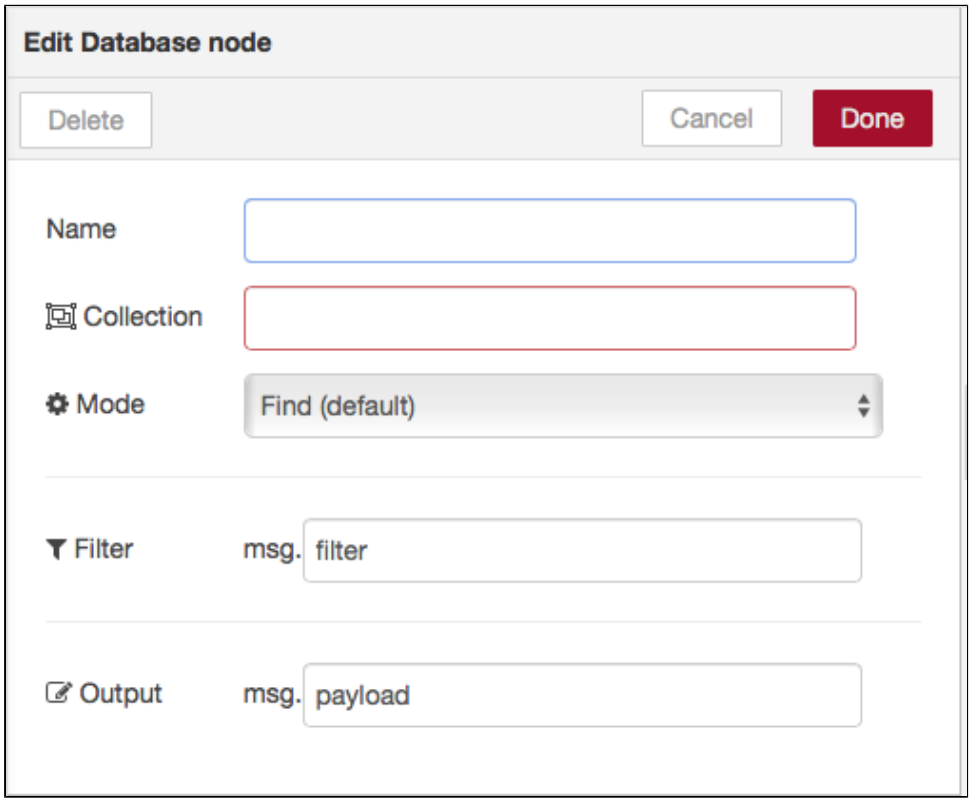
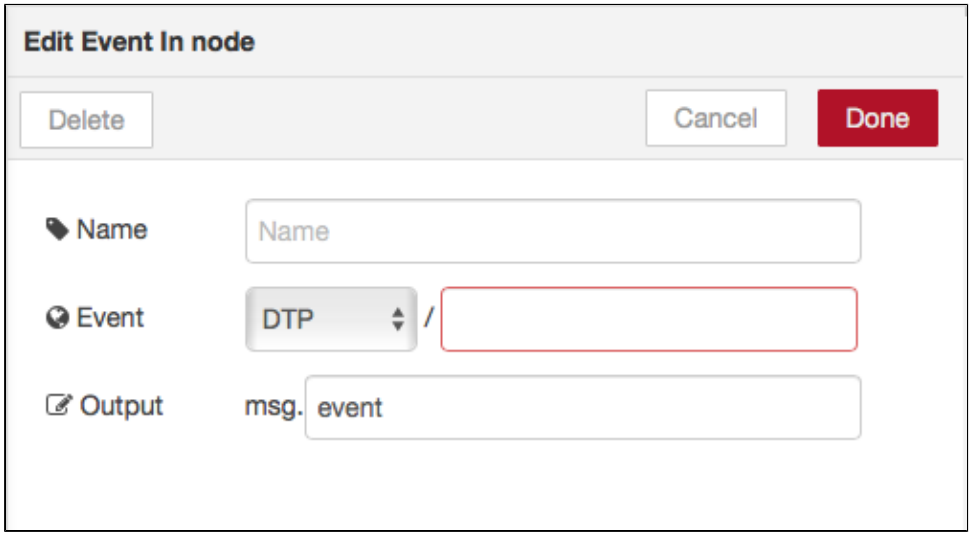
If a report schema has additional attributes assigned to it, then individual report profiles can be found by adding key/value pairs to the additional search parameters which match those additional attributes.

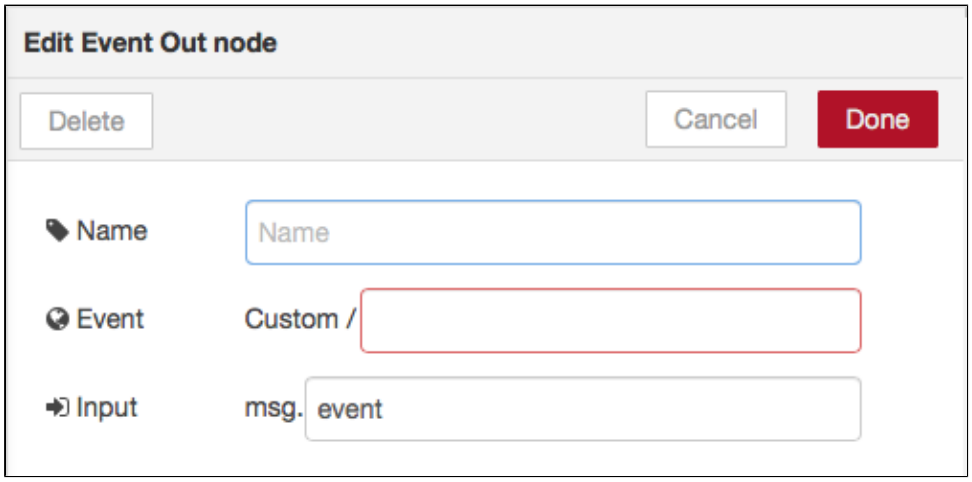
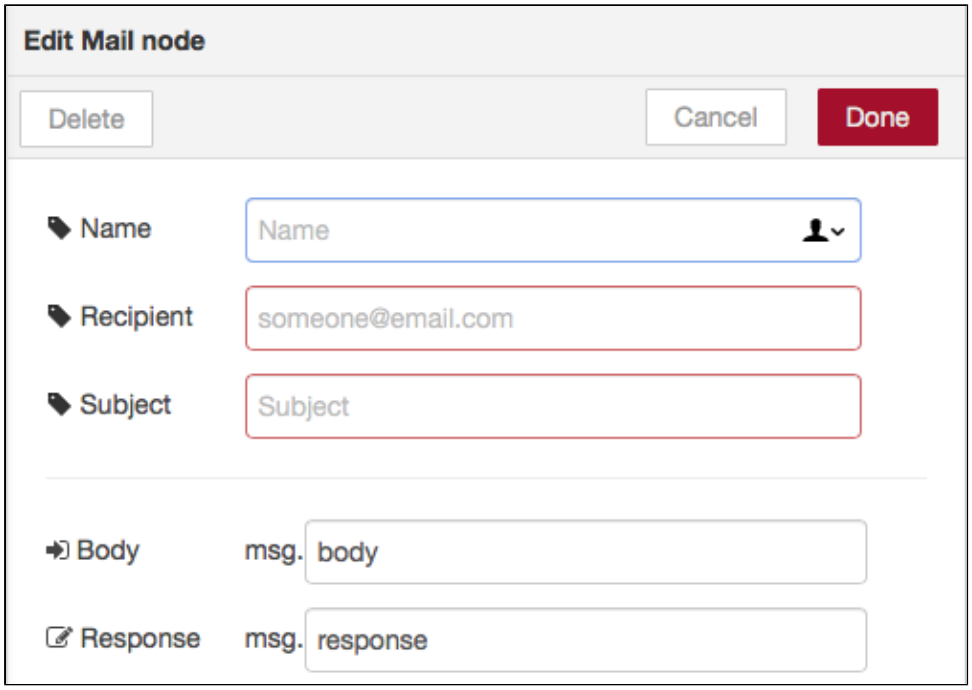
The key/value pairs created in this area can also mustache values ({{{property}}}) from the msg object that is passed to the node.

PC REST API	This general purpose node makes requests the Policy Center REST API.  Open the following URL in a browser to view the Policy Center REST API documentation and learn about the available endpoints: http://<HOST>:<PORT>/apidoc

DTP REST API	This general purpose node makes request to the DTP REST API. See REST API for information about endpoints in DTP.  There may be instances in which a DTP REST API node shows a persistent "requesting" message.  Check the $DTP_SERVICES_HOME/logs/<service UUID>.log file or the $DTP_SERVICES_HOME/logs/main.log file for error messages. In most instances, this issue is related to the service attempting to send a very large payload to the DTP REST API. Reduce or split the payload and make multiple connections to post the data.
DTP Server Info	This node gathers the DTP server information set in the configuration page (see DTP Enterprise Pack Configuration). 

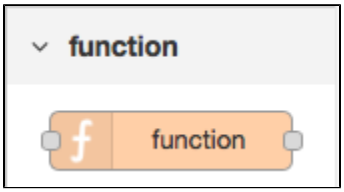
DTP Store Metrics	This node allows you to define a new metric type and send the new metric results to DTP in order to store custom metrics. 
Get User	This node gets DTP user information. 

Database	This node allows you to interact with the Enterprise Pack database. You can use the node to persist data across flow invocations and service restarts. 
Event In	This node is used to subscribe to MQTT events from the event broker. 

Event Out	This node publishes a event. 
Mail	This node can sent emails using the SMTP mail server defined in the Configuration tab. See DTP Enterprise Pack Configuration. 
i18n	This node stores localization information.

The Function Node

The function node is not a Parasoft-specific node, but it is an important part of nearly every flow in . You can write your own logic for your flow by adding JavaScript to the node. The function node is in the function palette.

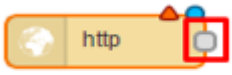
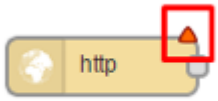


The node is pre-loaded with useful libraries:

lodash	Add the following line to your function node to initialize lodash and make its functionality is available. `var _ = context.global.lodash;` See https://lodash.com/docs to learn more about this utility library.
moment	This utility library is used to manipulate date and time in your flows. Add the following line to your function node to initialize moment and make its functionality is available. `var moment = context.glboal.moment;` See https://momentjs.com/docs/moment to learn more about this utility library.
os	This module from node.js detects operating system-specific information. Add the following line to your function node to initialize os and make its functionality is available. `var os = context.global.os;` See https://nodejs.org/api/os.html to learn more about this utility library.

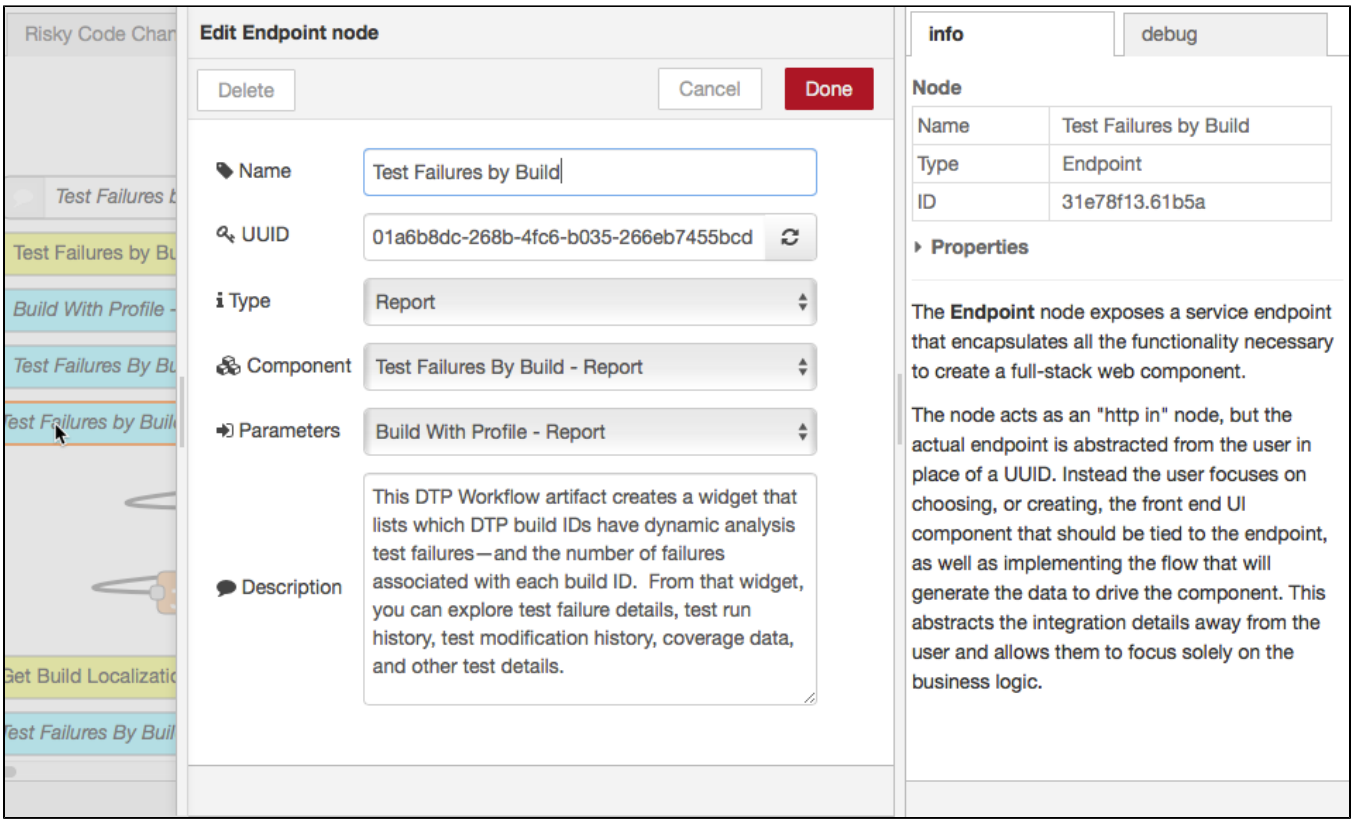
Click on the following link to learn more about the function node: https://nodered.org/docs/writing-functions.

Node Markings

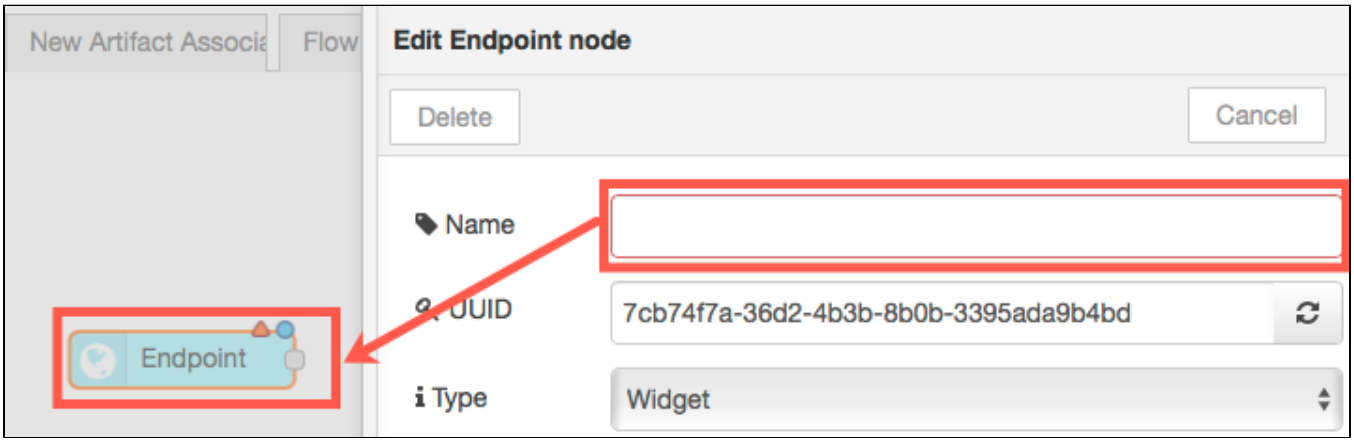
Input node		Nodes process data from left to right. A white square on the left hand side indicates that the node can receive inputs from a message object. The node's documentation will describe the expected input “msg” format.
Output node		A white square on the right hand side indicates that the node can output payload data. If there is more than one icon on the right hand side, it means that the node can output multiple sets of data.
Nodes with inputs and outputs		If a node has squares on both the left and right, it means it accepts inputs from a message object and can output payload data. For details on the expected inputs and outputs, see each node's on-line documentation (select the node, then review the details in the Info tab).
Node with missing configuration		A red triangle indicates that all required fields are not properly configured. Double-click the node to find out what field(s) require additional configuration (they will be marked with a red box, like the URL field below), then update the node accordingly.
Unsaved node		This blue circle indicates that the node is not yet saved in the server. If there is any node with a blue circle icon, the **Deploy** button will become active.

Editing Nodes

Double-click a node to open the node settings dialog. Different nodes have different editing controls depending on their function.



The Info tab shows documentation for the node. Many nodes have required fields. If a required field is not configured, it will be outlined in red. If you attempt to save a node that does not have a value in all of the required fields, a red triangle icon will be displayed above the node.



Mustache Format

You can use double curly brackets ({{ }}), referred to as “mustaches,” in function nodes, REST API nodes, and other nodes to mark message variables.

Example

The following snippet is added to your node:

```
Hello {{name}}. Today is {{date}}
```

The following message is received:

```
{
        name: "Fred",
        date: "Monday"
        payload: ...
}
```

The payload will result in the following:

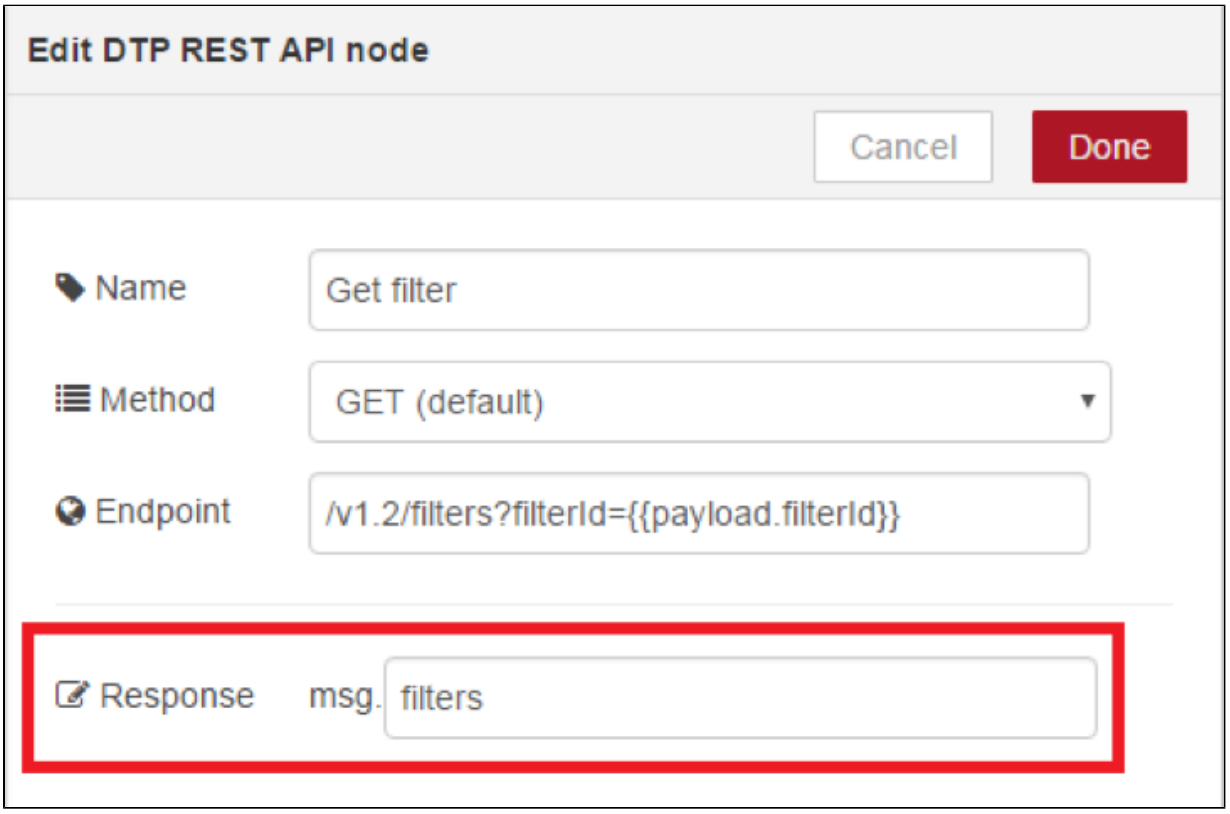
`Hello Fred. Today is Monday`

Additionally, you can render unescaped HTML in some nodes with triple curly bracket notation. For example, `{{{something}}}`.

Input Output Fields

Many nodes contain input and output fields, which are prefixed with "msg." The prefix indicates that the field is used to configure the property on the “msg” object that should be used as an input or an output (depending on the field's purpose).

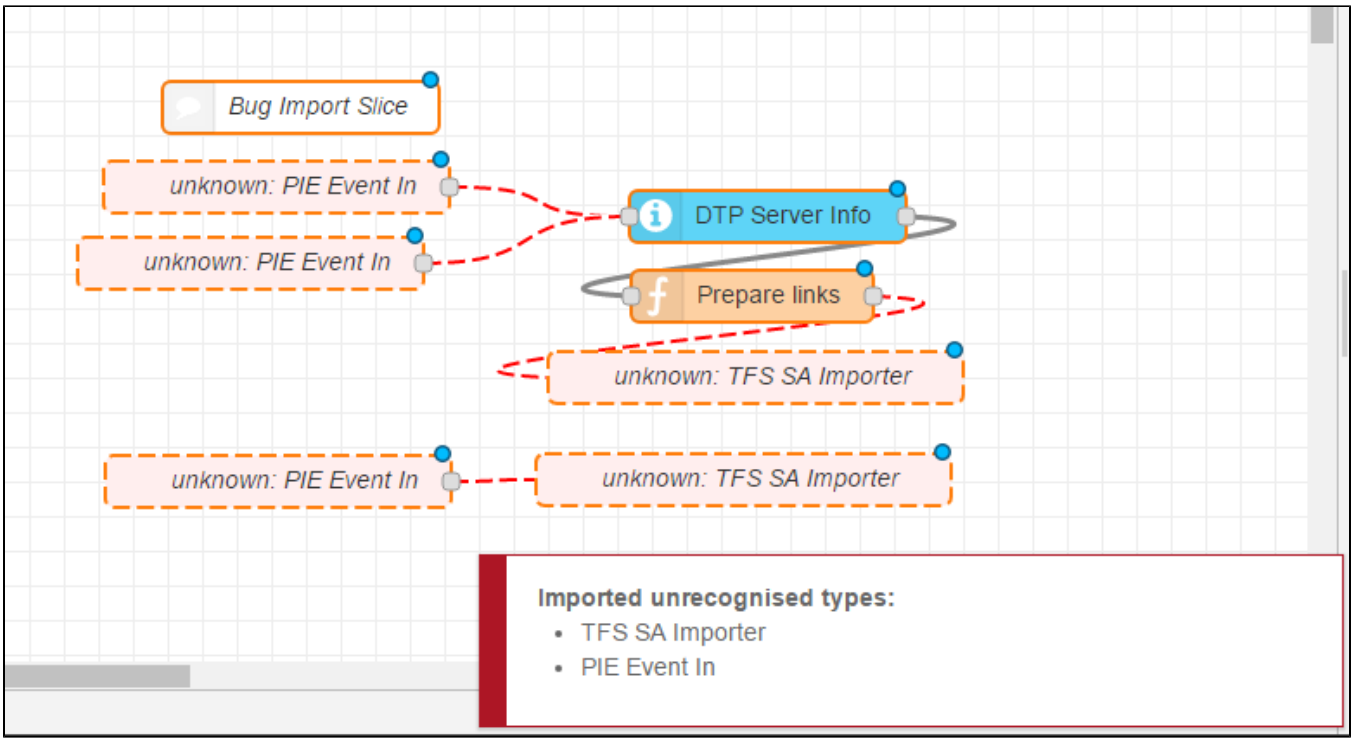
For example, one of the most common nodes that contains input and output properties is the DTP REST API node:



When the “GET” method is selected, the DTP REST API node provides only a “Response” field. This response field is used to define the property of the “msg” object where the response from the API should be written. Providing a method for defining the output property simplifies managing data from various resources, because you would otherwise need to add additional function nodes to move or restructure the data before additional data can be retrieved.

Addressing Unknown Nodes in a Deployed Slice

Missing or unrecognized nodes prevent all of the deployed flows from functioning properly. This might occur if, for example, you install an artifact that depends on an additional artifact that has not been installed.



If a deployed slice contains an unknown/unrecognized node, then either:

- Remove it from , or
- Install/reinstall the artifact that contains the missing node

Restart the service to take effect. New nodes are only registered when the service starts up.

As a best practice, be sure that when you uninstall an artifact from the Artifact Manager, you can also remove any flows that are broken by the removal of the artifact.