

Apache NiFi 3

Building a Data Flow

Date of Publish: 2020-12-15



<https://docs.cloudera.com/>

Contents

| | |
|--|----------|
| Building a DataFlow..... | 3 |
| Adding Components to the Canvas..... | 3 |
| Component Versions..... | 11 |
| Sorting and Filtering Components..... | 12 |
| Changing Component Versions..... | 13 |
| Understanding Version Dependencies..... | 15 |
| Configuring a Processor..... | 18 |
| Settings Tab..... | 18 |
| Scheduling Tab..... | 20 |
| Properties Tab..... | 23 |
| Comments Tab..... | 27 |
| Additional Help..... | 28 |
| Parameters..... | 28 |
| Parameter Contexts..... | 29 |
| Adding a Parameter to a Parameter Context..... | 30 |
| Assigning a Parameter Context to a Process Group..... | 33 |
| Referencing Parameters..... | 36 |
| Accessing Parameters..... | 43 |
| Using Custom Properties with Expression Language..... | 45 |
| Variables..... | 46 |
| Referencing Custom Properties via nifi.properties..... | 57 |
| Controller Services..... | 57 |
| Adding Controller Services for Reporting Tasks..... | 57 |
| Adding Controller Services for Dataflows..... | 61 |
| Enabling/Disabling Controller Services..... | 63 |
| Reporting Tasks..... | 64 |
| Connecting Components..... | 68 |
| Details Tab..... | 69 |
| Settings..... | 70 |
| Changing Configuration and Context Menu Options..... | 76 |
| Bending Connections..... | 77 |
| Processor Validation..... | 78 |
| Site-to-Site..... | 78 |
| Configure Site-to-Site client NiFi instance..... | 80 |
| Configure Site-to-Site Server NiFi Instance..... | 81 |
| Example Dataflow..... | 82 |

Building a DataFlow

A DFM is able to build an automated dataflow using the NiFi UI. Simply drag components from the toolbar to the canvas, configure the components to meet specific needs, and connect the components together.

Adding Components to the Canvas

The User Interface section above outlined the different segments of the UI and pointed out a Components Toolbar. This section looks at each of the Components in that toolbar:



Processor: The Processor is the most commonly used component, as it is responsible for data ingress, egress, routing, and manipulating. There are many different types of Processors. In fact, this is a very common Extension Point in NiFi, meaning that many vendors may implement their own Processors to perform whatever functions are necessary for their use case. When a Processor is dragged onto the canvas, the user is presented with a dialog to choose which type of Processor to use:

Add Processor

Source
Displaying 219 of 219

all groups ▾

amazon attributes
 avro aws consume
 csv database fetch
 files get hadoop
 ingest input insert
 json listen logs
 message put
 remote restricted
 source sql text
 update

| Type ▾ | Version | Tags |
|------------------------|---------|---|
| AttributeRollingWindow | 1.2.0 | rolling, data science, Attribute Expression Language, st... |
| AttributesToJSON | 1.2.0 | flowfile, json, attributes |
| Base64EncodeContent | 1.2.0 | encode, base64 |
| CaptureChangeMySQL | 1.2.0 | cdc, jdbc, mysql, sql |
| CompareFuzzyHash | 1.2.0 | fuzzy-hashing, hashing, cyber-security |
| CompressContent | 1.2.0 | lzma, decompress, compress, snappy framed, gzip, sna... |
| ConnectWebSocket | 1.2.0 | subscribe, consume, listen, WebSocket |
| ConsumeAMQP | 1.2.0 | receive, amqp, rabbit, get, consume, message |
| ConsumeEWS | 1.2.0 | EWS, Exchange, Email, Consume, Ingest, Message, Get,... |
| ConsumeIMAP | 1.2.0 | Imap, Email, Consume, Ingest, Message, Get, Ingress |
| ConsumeJMS | 1.2.0 | jms, receive, get, consume, message |
| ConsumeKafka | 1.2.0 | PubSub, Consume, Inqest, Get, Kafka, Ingress, Topic, 0... |

AttributeRollingWindow 1.2.0

org.apache.nifi - nifi-stateful-analysis-nar

Track a Rolling Window based on evaluating an Expression Language expression on each FlowFile and add that value to the processor's state. Each FlowFile will be emitted with the count of FlowFiles and total aggregate value of values processed in the current time window.

CANCEL
ADD

In the top-right corner, the user is able to filter the list based on the Processor Type or the Tags associated with a Processor. Processor developers have the ability to add Tags to their Processors. These tags are used in this dialog for filtering and are displayed on the left-hand side in a Tag Cloud. The more Processors that exist with a particular Tag, the larger the Tag appears in the Tag Cloud. Clicking a Tag in the Cloud will filter the available Processors to only those that contain that Tag. If multiple Tags are selected, only those Processors that contain all of those Tags are shown. For example, if we want to show only those Processors that allow us to ingest files, we can select both the files Tag and the ingest Tag:

Add Processor

Source Filter

all groups Displaying 11 of 219

| Type ^ | Version | Tags |
|-----------|---------|---|
| FetchFTP | 1.2.0 | input, ftp, get, fetch, retrieve, files, source, remote, ingest |
| FetchFile | 1.2.0 | ingress, input, restricted, get, files, source, local, filesystem, ingest |
| FetchSFTP | 1.2.0 | input, get, fetch, retrieve, files, sftp, source, remote, ingest |
| GetFTP | 1.2.0 | input, FTP, get, fetch, retrieve, files, source, remote, ingest |
| GetFile | 1.2.0 | ingress, input, restricted, get, files, source, local, filesystem, ingest |
| GetHDFS | 1.2.0 | restricted, get, fetch, HDFS, hadoop, source, filesystem, ingest |
| GetSFTP | 1.2.0 | input, get, fetch, retrieve, files, sftp, source, remote, ingest |
| ListFTP | 1.2.0 | input, ftp, files, source, list, remote, ingest |
| ListFile | 1.2.0 | file, get, source, list, filesystem, ingest |
| ListHDFS | 1.2.0 | get, HDFS, hadoop, source, list, filesystem, ingest |
| ListSFTP | 1.2.0 | input, files, sftp, source, list, remote, ingest |

FetchFTP 1.2.0 org.apache.nifi - nifi-standard-nar

Fetches the content of a file from a remote SFTP server and overwrites the contents of an incoming FlowFile with the content of the remote file.

CANCEL
ADD

Restricted components will be marked with a



icon next to their name. These are components that can be used to execute arbitrary unsanitized code provided by the operator through the NiFi REST API/UI or can be used to obtain or alter data on the NiFi host system using the NiFi OS credentials. These components could be used by an otherwise authorized NiFi user to go beyond the intended use of the application, escalate privilege, or could expose data about the internals of the NiFi process or the host system. All of these capabilities should be considered privileged, and admins should be aware of these capabilities and explicitly enable them for a subset of trusted users. Before a user is allowed to create and modify restricted components they must be granted access. Hovering over the



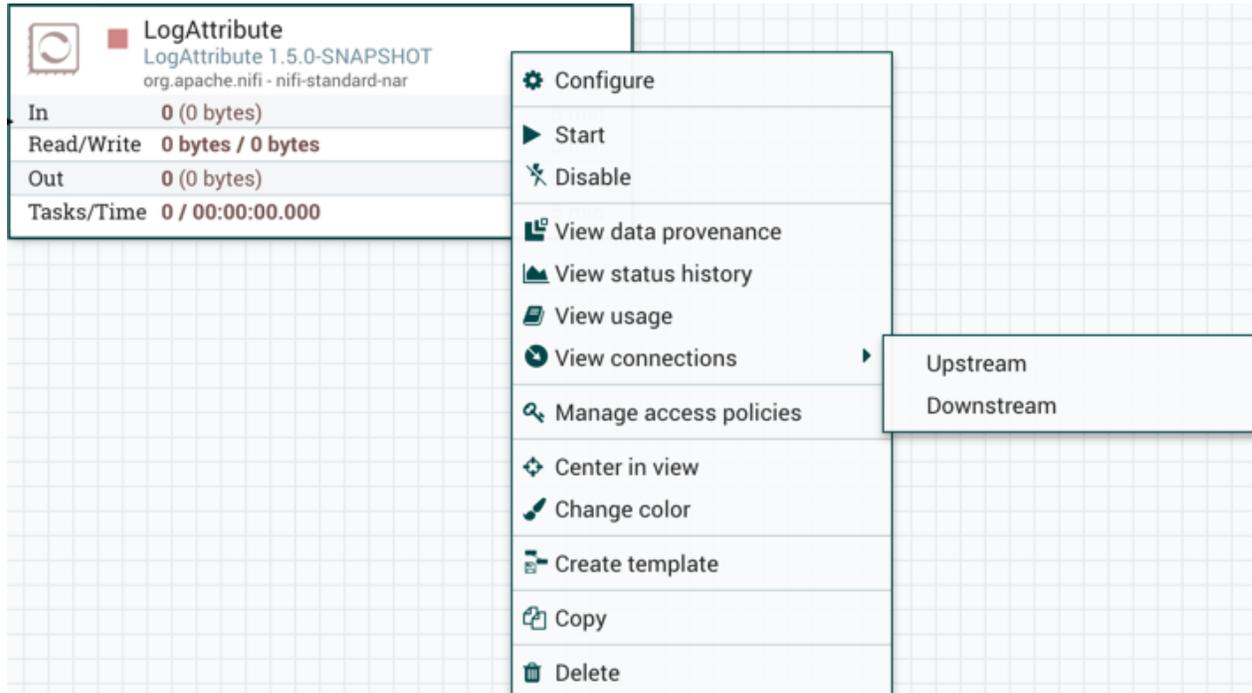
icon will display the specific permissions a restricted component requires. Permissions can be assigned regardless of restrictions. In this case, the user will have access to all restricted components. Alternatively, users can be assigned access to specific restrictions. If the user has been granted access to all restrictions a component requires, they will have access to that component assuming otherwise sufficient permissions. For more information refer to *Accessing the UI with Multi-Tenant Authorization and Restricted Components in Versioned Flows*.

Clicking the "Add" button or double-clicking on a Processor Type will add the selected Processor to the canvas at the location that it was dropped.



Note: For any component added to the canvas, it is possible to select it with the mouse and move it anywhere on the canvas. Also, it is possible to select multiple items at once by either holding down the Shift key and selecting each item or by holding down the Shift key and dragging a selection box around the desired components.

Once you have dragged a Processor onto the canvas, you can interact with it by right-clicking on the Processor and selecting an option from the context menu. The options available to you from the context menu vary, depending on the privileges assigned to you.



While the options available from the context menu vary, the following options are typically available when you have full privileges to work with a Processor:

- **Configure:** This option allows the user to establish or change the configuration of the Processor (see *Configuring a Processor*).



Note: For Processors, Ports, Remote Process Groups, Connections and Labels, it is possible to open the configuration dialog by double-clicking on the desired component.

- **Start or Stop:** This option allows the user to start or stop a Processor; the option will be either Start or Stop, depending on the current state of the Processor.
- **Enable or Disable:** This option allows the user to enable or disable a Processor; the option will be either Enable or Disable, depending on the current state of the Processor.
- **View data provenance:** This option displays the NiFi Data Provenance table, with information about data provenance events for the FlowFiles routed through that Processor (see *Data Provenance*).
- **View status history:** This option opens a graphical representation of the Processor's statistical information over time.
- **View usage:** This option takes the user to the Processor's usage documentation.
- **View connections#Upstream:** This option allows the user to see and "jump to" upstream connections that are coming into the Processor. This is particularly useful when processors connect into and out of other Process Groups.
- **View connections#Downstream:** This option allows the user to see and "jump to" downstream connections that are going out of the Processor. This is particularly useful when processors connect into and out of other Process Groups.
- **Center in view:** This option centers the view of the canvas on the given Processor.
- **Change color:** This option allows the user to change the color of the Processor, which can make the visual management of large flows easier.
- **Create template:** This option allows the user to create a template from the selected Processor.

- Copy: This option places a copy of the selected Processor on the clipboard, so that it may be pasted elsewhere on the canvas by right-clicking on the canvas and selecting Paste. The Copy/Paste actions also may be done using the keystrokes Ctrl-C (Command-C) and Ctrl-V (Command-V).
- Delete: This option allows the DFM to delete a Processor from the canvas.



Input Port: Input Ports provide a mechanism for transferring data into a Process Group. When an Input Port is dragged onto the canvas, the DFM is prompted to name the Port. All Ports within a Process Group must have unique names.

All components exist only within a Process Group. When a user initially navigates to the NiFi page, the user is placed in the Root Process Group. If the Input Port is dragged onto the Root Process Group, the Input Port provides a mechanism to receive data from remote instances of NiFi via *Site to Site*. In this case, the Input Port can be configured to restrict access to appropriate users, if NiFi is configured to run securely. For information on configuring NiFi to run securely, see the *System Administration documentation* in our *Reference* section.



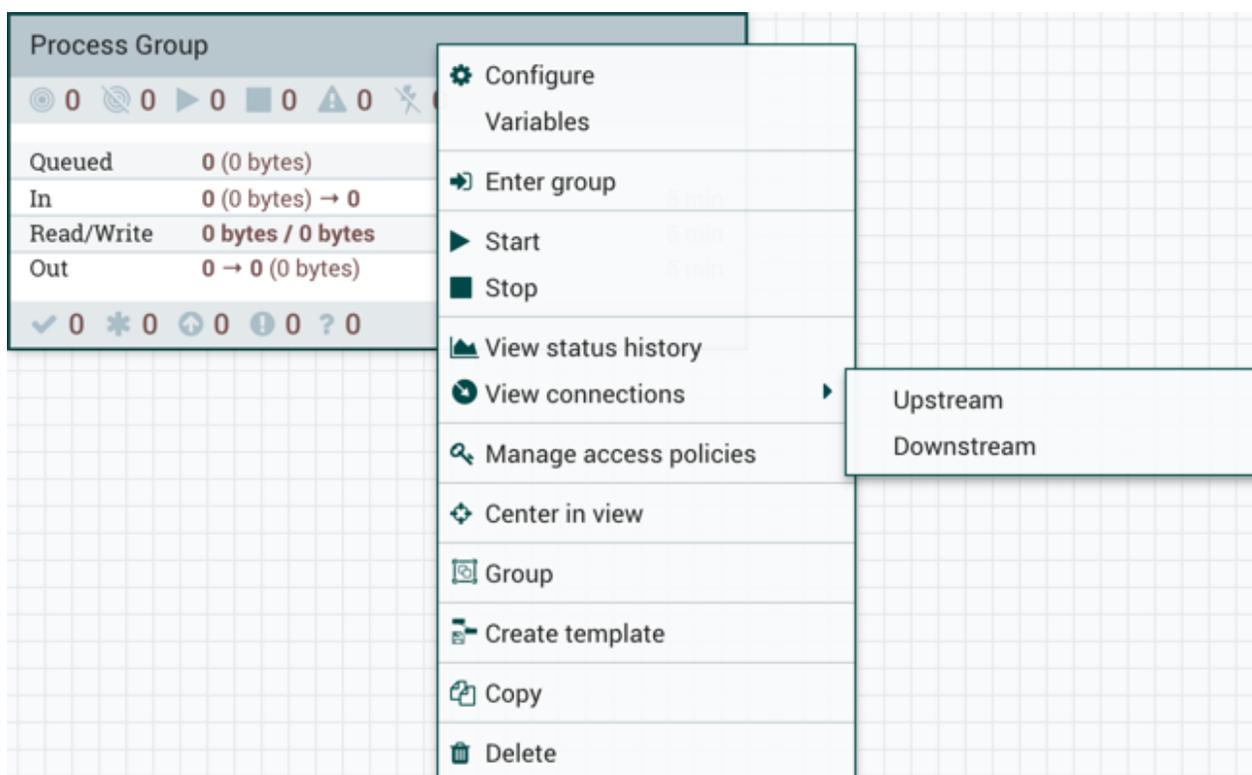
Output Port: Output Ports provide a mechanism for transferring data from a Process Group to destinations outside of the Process Group. When an Output Port is dragged onto the canvas, the DFM is prompted to name the Port. All Ports within a Process Group must have unique names.

If the Output Port is dragged onto the Root Process Group, the Output Port provides a mechanism for sending data to remote instances of NiFi via *Site to Site*. In this case, the Port acts as a queue. As remote instances of NiFi pull data from the port, that data is removed from the queues of the incoming Connections. If NiFi is configured to run securely, the Output Port can be configured to restrict access to appropriate users. For information on configuring NiFi to run securely, see the *System Administration documentation* in our *Reference* section.



Process Group: Process Groups can be used to logically group a set of components so that the dataflow is easier to understand and maintain. When a Process Group is dragged onto the canvas, the DFM is prompted to name the Process Group. All Process Groups within the same parent group must have unique names. The Process Group will then be nested within that parent group.

Once you have dragged a Process Group onto the canvas, you can interact with it by right-clicking on the Process Group and selecting an option from the context menu. The options available to you from the context menu vary, depending on the privileges assigned to you.



While the options available from the context menu vary, the following options are typically available when you have full privileges to work with the Process Group:

- **Configure:** This option allows the user to establish or change the configuration of the Process Group.
- **Variables:** This option allows the user to create or configure variables within the NiFi UI.
- **Enter group:** This option allows the user to enter the Process Group.



Note: It is also possible to double-click on the Process Group to enter it.

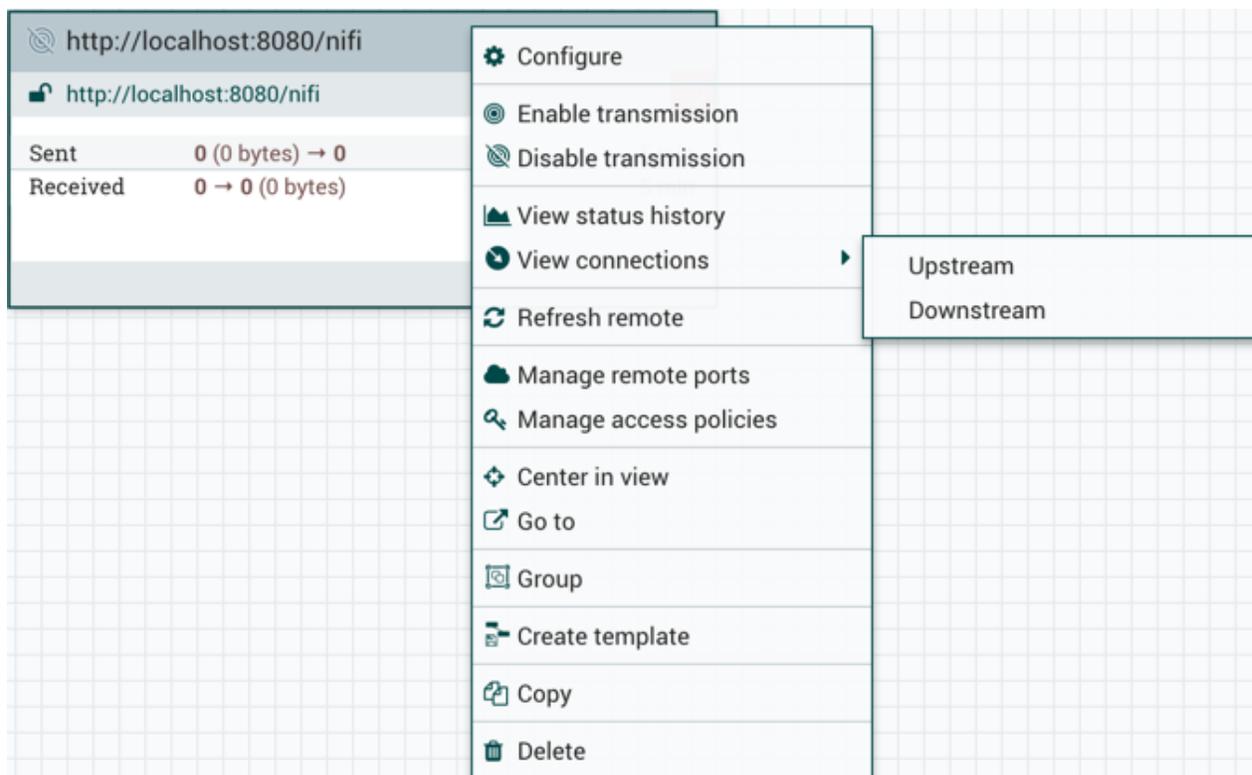
- **Start:** This option allows the user to start a Process Group.
- **Stop:** This option allows the user to stop a Process Group.
- **View status history:** This option opens a graphical representation of the Process Group's statistical information over time.
- **View connections#Upstream:** This option allows the user to see and "jump to" upstream connections that are coming into the Process Group.
- **View connections#Downstream:** This option allows the user to see and "jump to" downstream connections that are going out of the Process Group.
- **Center in view:** This option centers the view of the canvas on the given Process Group.
- **Group:** This option allows the user to create a new Process Group that contains the selected Process Group and any other components selected on the canvas.
- **Create template:** This option allows the user to create a template from the selected Process Group.
- **Copy:** This option places a copy of the selected Process Group on the clipboard, so that it may be pasted elsewhere on the canvas by right-clicking on the canvas and selecting Paste. The Copy/Paste actions also may be done using the keystrokes Ctrl-C (Command-C) and Ctrl-V (Command-V).
- **Delete:** This option allows the DFM to delete a Process Group.



Remote Process Group: Remote Process Groups appear and behave similar to Process Groups. However, the Remote Process Group (RPG) references a remote instance of NiFi. When an RPG is dragged onto the canvas, rather than being prompted for a name, the DFM is prompted for the URL of the remote NiFi instance. If the remote NiFi is a clustered instance, adding two or more cluster node URLs is recommended so that an initial connection can be made even if one of the nodes is unavailable. Multiple URLs can be specified in a comma-separated format.

When data is transferred to a clustered instance of NiFi via an RPG, the RPG will first connect to the remote instance whose URL is configured to determine which nodes are in the cluster and how busy each node is. This information is then used to load balance the data that is pushed to each node. The remote instances are then interrogated periodically to determine information about any nodes that are dropped from or added to the cluster and to recalculate the load balancing based on each node's load. For more information, see the section on *Site to Site*.

Once you have dragged a Remote Process Group onto the canvas, you can interact with it by right-clicking on the Remote Process Group and selecting an option from the context menu. The options available to you from the menu vary, depending on the privileges assigned to you.



The following options are typically available when you have full privileges to work with the Remote Process Group:

- **Configure:** This option allows the user to establish or change the configuration of the Remote Process Group.
- **Enable transmission:** Makes the transmission of data between NiFi instances active (see *Remote Process Group Transmission*).
- **Disable transmission:** Disables the transmission of data between NiFi instances.
- **View status history:** This option opens a graphical representation of the Remote Process Group's statistical information over time.
- **View connections#Upstream:** This option allows the user to see and "jump to" upstream connections that are coming into the Remote Process Group.
- **View connections#Downstream:** This option allows the user to see and "jump to" downstream connections that are going out of the Remote Process Group.

- Refresh remote: This option refreshes the view of the status of the remote NiFi instance.
- Manage remote ports: This option allows the user to see input ports and/or output ports that exist on the remote instance of NiFi that the Remote Process Group is connected to. Note that if the Site-to-Site configuration is secure, only the ports that the connecting NiFi has been given access to will be visible.
- Center in view: This option centers the view of the canvas on the given Remote Process Group.
- Go to: This option opens a view of the remote NiFi instance in a new tab of the browser. Note that if the Site-to-Site configuration is secure, the user must have access to the remote NiFi instance in order to view it.
- Group: This option allows the user to create a new Process Group that contains the selected Remote Process Group and any other components selected on the canvas.
- Create template: This option allows the user to create a template from the selected Remote Process Group.
- Copy: This option places a copy of the selected Process Group on the clipboard, so that it may be pasted elsewhere on the canvas by right-clicking on the canvas and selecting Paste. The Copy/Paste actions also may be done using the keystrokes Ctrl-C (Command-C) and Ctrl-V (Command-V).
- Delete: This option allows the DFM to delete a Remote Process Group from the canvas.



Funnel: Funnels are used to combine the data from many Connections into a single Connection. This has two advantages. First, if many Connections are created with the same destination, the canvas can become cluttered if those Connections have to span a large space. By funneling these Connections into a single Connection, that single Connection can then be drawn to span that large space instead. Secondly, Connections can be configured with FlowFile Prioritizers. Data from several Connections can be funneled into a single Connection, providing the ability to Prioritize all of the data on that one Connection, rather than prioritizing the data on each Connection independently.



Template: Templates can be created by DFMs from sections of the flow, or they can be imported from other dataflows. These Templates provide larger building blocks for creating a complex flow quickly. When the Template is dragged onto the canvas, the DFM is provided a dialog to choose which Template to add to the canvas:

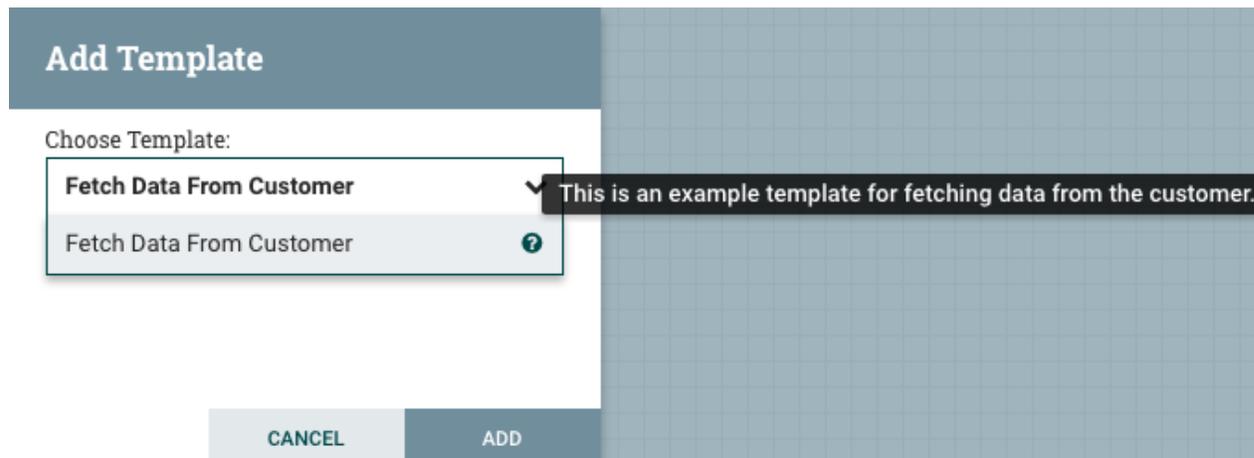
Add Template

Choose Template:

Fetch Data From Customer ▼

CANCEL **ADD**

Clicking the drop-down box shows all available Templates. Any Template that was created with a description will show a question mark icon, indicating that there is more information. Hovering over the icon with the mouse will show this description:



Label: Labels are used to provide documentation to parts of a dataflow. When a Label is dropped onto the canvas, it is created with a default size. The Label can then be resized by dragging the handle in the bottom-right corner. The Label has no text when initially created. The text of the Label can be added by right-clicking on the Label and choosing Configure.

Component Versions

You have access to information about the version of your Processors, Controller Services, and Reporting Tasks. This is especially useful when you are working within a clustered environment with multiple NiFi instances running different versions of a component or if you have upgraded to a newer version of a processor. The Add Processor, Add Controller Service, and Add Reporting Task dialogs include a column identifying the component version, as well as the name of the component, the organization or group that created the component, and the NAR bundle that contains the component.

Add Processor

Source example

all groups ▼ Displaying 2 of 221

| Type | Version ▲ | Tags |
|-------------|-----------|---------|
| MyProcessor | 1.0 | example |
| MyProcessor | 2.0 | example |

amazon attributes

avro aws

consume csv

database fetch

files get hadoop

ingest input

insert json listen

logs message

put remote

restricted source

sql text update

Name **Group** **Bundle**

↓ ↓ ↓

MyProcessor 1.0 org.apache.nifi - nifi-example-processors-nar

Provide a description

CANCEL
ADD

Each component displayed on the canvas also contains this information.

MyProcessor

MyProcessor 1.0

org.apache.nifi - nifi-example-processors-nar

| | | |
|-------------------|--------------------------|--------------|
| In | 0 (0 bytes) | 5 min |
| Read/Write | 0 bytes / 0 bytes | 5 min |
| Out | 0 (0 bytes) | 5 min |
| Tasks/Time | 0 / 00:00:00.000 | 5 min |

Sorting and Filtering Components

When you are adding a component, you can sort on version number or filter based on originating source.

To sort based on version, click the version column to display in ascending or descending version order.

To filter based on source group, click the source drop-down in the upper left of your Add Component dialog, and select the group you want to view.

Add Processor

Source

Displaying 2 of 221

example

all groups ▼

| Type | Version ▲ | Tags |
|-------------|-----------|---------|
| MyProcessor | 1.0 | example |
| MyProcessor | 2.0 | example |

amazon

attributes

avro

aws

consume

csv

database

fetch

files

get

hadoop

ingest

input

insert

json

listen

logs

message

put

remote

restricted

source

sql

text

update

MyProcessor 1.0

org.apache.nifi - nifi-example-processors-nar

Provide a description

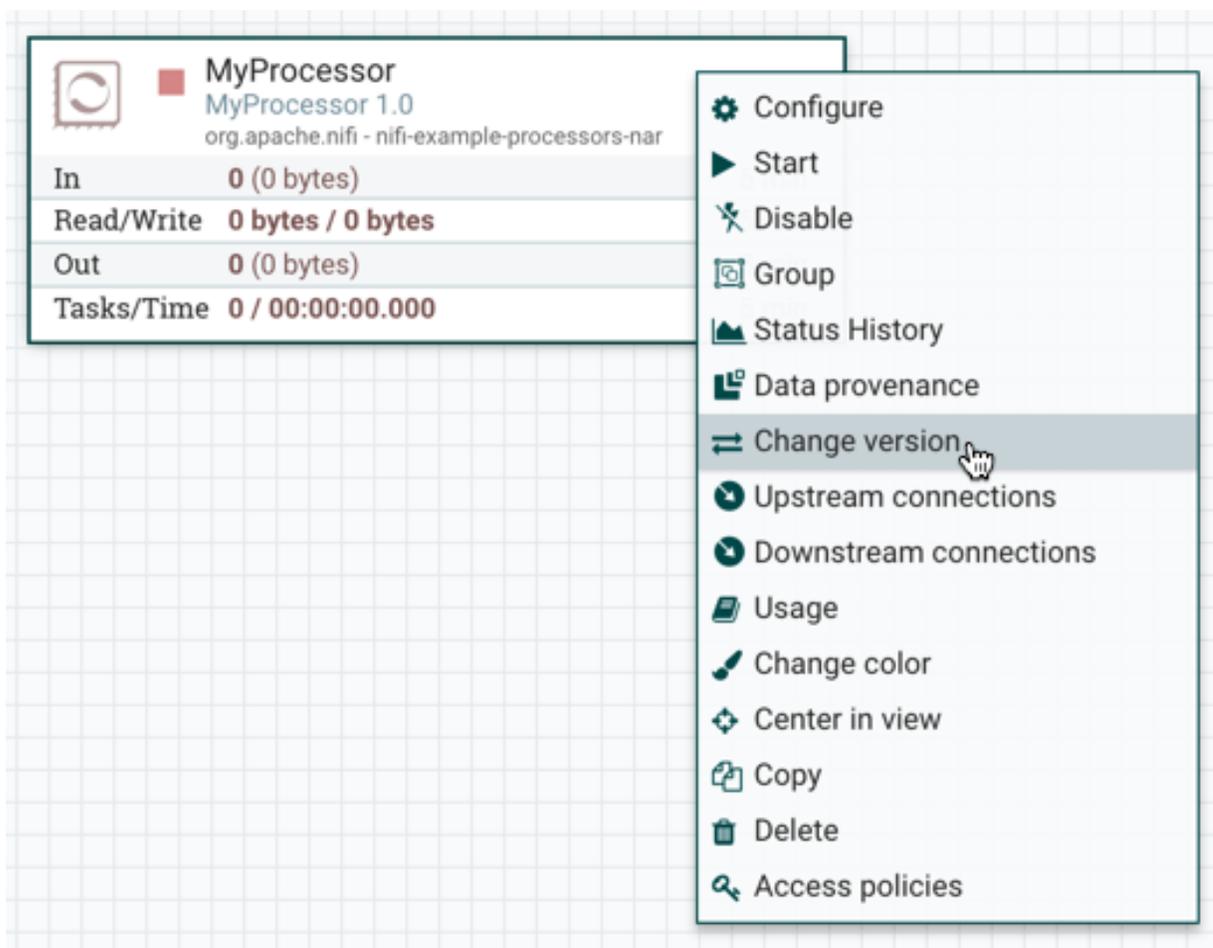
CANCEL

ADD

Changing Component Versions

To change a component version, perform the following steps.

1. Right-click the component on the canvas to display configuration options.
2. Select Change version.



3. In the Component Version dialog, select the version you want to run from the Version drop-down menu.

Component Version

Name
MyProcessor

Bundle
org.apache.nifi - nifi-example-processors-nar

Version

| | |
|-----|---|
| 1.0 | ▼ |
| 2.0 | ? |
| 1.0 | ? |

Restriction
No restriction

Description
Provide a description

CANCEL **APPLY**

Understanding Version Dependencies

When you are configuring a component, you can also view information about version dependencies.

1. Right-click your component and select Configure to display the Configure dialog for your component.
2. Click the Properties tab.
3. Click the information icon to view any version dependency information.

Configure Processor

SETTINGS SCHEDULING **PROPERTIES** COMMENTS

Required field +

| Property | Value |
|-------------|--------------|
| My Property | No value set |
| My Service | No value set |

Example Controller Service Property
Supports expression language: false
Requires Controller Service: MyService 1.0 from org.apache.nifi - nifi-example-service-api-nar

CANCEL APPLY

In the following example, MyProcessor version 1.0 is configured properly with the controller service StandardMyService version 1.0:

Configure Processor

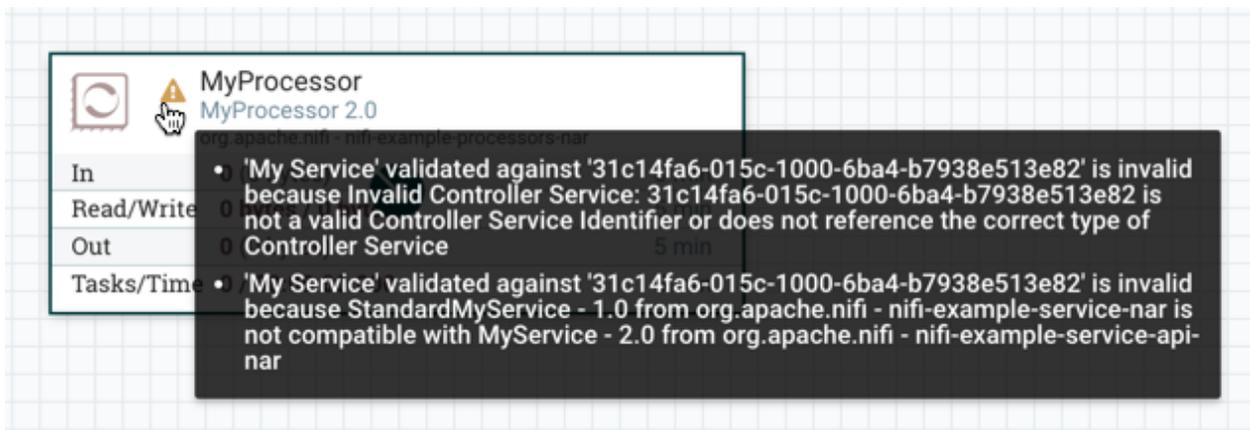
SETTINGS
SCHEDULING
PROPERTIES
COMMENTS

Required field +

| Property | Value |
|--|--|
| My Property ? | test |
| My Service ? | StandardMyService 1.0 → |

CANCEL
APPLY

If the version of MyProcessor is changed to an incompatible version (MyProcessor 2.0), validation errors will be displayed on the processor:



and an error message will be displayed in the processor's controller service configuration since the service is no longer valid:

Configure Processor

SETTINGS
SCHEDULING
PROPERTIES
COMMENTS

Required field +

| Property | Value | |
|-------------|-------|--|
| My Property | ? | Property1 |
| My Service | ? | Incompatible Controller Service Configured |

CANCEL
APPLY

Configuring a Processor

To configure a processor, right-click on the Processor and select the Configure option from the context menu. Alternatively, just double-click on the Processor. The configuration dialog is opened with four different tabs, each of which is discussed below. Once you have finished configuring the Processor, you can apply the changes by clicking "Apply" or cancel all changes by clicking "Cancel".

Note that after a Processor has been started, the context menu shown for the Processor no longer has a Configure option but rather has a View Configuration option. Processor configuration cannot be changed while the Processor is running. You must first stop the Processor and wait for all of its active tasks to complete before configuring the Processor again.

Note that entering certain control characters are not supported and will be automatically filtered out when entered. The following characters and any unpaired Unicode surrogate codepoints will not be retained in any configuration:

```
[#x0], [#x1], [#x2], [#x3], [#x4], [#x5], [#x6], [#x7], [#x8], [#xB], [#xC],
[#xE], [#xF], [#x10], [#x11], [#x12], [#x13], [#x14], [#x15], [#x16],
[#x17], [#x18], [#x19], [#x1A], [#x1B], [#x1C], [#x1D], [#x1E], [#x1F],
[#xFFFE], [#xFFFF]
```

Settings Tab

The first tab in the Processor Configuration dialog is the Settings tab:

Configure Processor

SETTINGS
SCHEDULING
PROPERTIES
COMMENTS

Name
RouteOnAttribute Enabled

Id
313ab384-015c-1000-46ec-0ead80a5eb1e

Type
RouteOnAttribute 1.2.0

Bundle
org.apache.nifi - nifi-standard-nar

Penalty Duration ?
30 sec

Yield Duration ?
1 sec

Bulletin Level ?
WARN ▼

Automatically Terminate Relationships ?
 unmatched
FlowFiles that do not match any user-define expression will be routed here

CANCEL
APPLY

This tab contains several different configuration items. First, it allows the DFM to change the name of the Processor. The name of a Processor by default is the same as the Processor type. Next to the Processor Name is a checkbox, indicating whether the Processor is Enabled. When a Processor is added to the canvas, it is enabled. If the Processor is disabled, it cannot be started. The disabled state is used to indicate that when a group of Processors is started, such as when a DFM starts an entire Process Group, this (disabled) Processor should be excluded.

Below the Name configuration, the Processor's unique identifier is displayed along with the Processor's type and NAR bundle. These values cannot be modified.

Next are two dialogues for configuring 'Penalty Duration' and 'Yield Duration'. During the normal course of processing a piece of data (a FlowFile), an event may occur that indicates that the data cannot be processed at this time but the data may be processable at a later time. When this occurs, the Processor may choose to Penalize the FlowFile. This will prevent the FlowFile from being Processed for some period of time. For example, if the Processor is to push the data to a remote service, but the remote service already has a file with the same name as the filename that the Processor is specifying, the Processor may penalize the FlowFile. The 'Penalty Duration' allows the DFM to specify how long the FlowFile should be penalized. The default value is 30 seconds.

Similarly, the Processor may determine that some situation exists such that the Processor can no longer make any progress, regardless of the data that it is processing. For example, if a Processor is to push data to a remote service and that service is not responding, the Processor cannot make any progress. As a result, the Processor should 'yield', which will prevent the Processor from being scheduled to run for some period of time. That period of time is specified by setting the 'Yield Duration'. The default value is 1 second.

The last configurable option on the left-hand side of the Settings tab is the Bulletin level. Whenever the Processor writes to its log, the Processor also will generate a Bulletin. This setting indicates the lowest level of Bulletin that should be shown in the User Interface. By default, the Bulletin level is set to WARN, which means it will display all warning and error-level bulletins.

The right-hand side of the Settings tab contains an 'Automatically Terminate Relationships' section. Each of the Relationships that is defined by the Processor is listed here, along with its description. In order for a Processor to be considered valid and able to run, each Relationship defined by the Processor must be either connected to a downstream component or auto-terminated. If a Relationship is auto-terminated, any FlowFile that is routed to that Relationship will be removed from the flow and its processing considered complete. Any Relationship that is already connected to a downstream component cannot be auto-terminated. The Relationship must first be removed from any Connection that uses it. Additionally, for any Relationship that is selected to be auto-terminated, the auto-termination status will be cleared (turned off) if the Relationship is added to a Connection.

Scheduling Tab

The second tab in the Processor Configuration dialog is the Scheduling Tab:

Scheduling Strategy

The first configuration option is the Scheduling Strategy. There are three possible options for scheduling components:

Timer driven: This is the default mode. The Processor will be scheduled to run on a regular interval. The interval at which the Processor is run is defined by the 'Run Schedule' option (see below).

Event driven: When this mode is selected, the Processor will be triggered to run by an event, and that event occurs when FlowFiles enter Connections feeding this Processor. This mode is currently considered experimental and is not supported by all Processors. When this mode is selected, the 'Run Schedule' option is not configurable, as the Processor is not triggered to run periodically but as the result of an event. Additionally, this is the only mode for which the 'Concurrent Tasks' option can be set to 0. In this case, the number of threads is limited only by the size of the Event-Driven Thread Pool that the administrator has configured.

CRON driven: When using the CRON driven scheduling mode, the Processor is scheduled to run periodically, similar to the Timer driven scheduling mode. However, the CRON driven mode provides significantly more flexibility at the expense of increasing the complexity of the configuration. The CRON driven scheduling value is a string of six required fields and one optional field, each separated by a space. These fields are:

| Field | Valid values |
|-----------------|------------------|
| Seconds | 0-59 |
| Minutes | 0-59 |
| Hours | 0-23 |
| Day of Month | 1-31 |
| Month | 1-12 or JAN-DEC |
| Day of Week | 1-7 or SUN-SAT |
| Year (optional) | empty, 1970-2099 |

You typically specify values one of the following ways:

- Number: Specify one or more valid value. You can enter more than one value using a comma-separated list.
- Range: Specify a range using the <number>-<number> syntax.
- Increment: Specify an increment using <start value>/<increment> syntax. For example, in the Minutes field, 0/15 indicates the minutes 0, 15, 30, and 45.

You should also be aware of several valid special characters:

- * - Indicates that all values are valid for that field.
- ? - Indicates that no specific value is specified. This special character is valid in the Days of Month and Days of Week field.
- L - You can append L to one of the Day of Week values, to specify the last occurrence of this day in the month. For example, 1L indicates the last Sunday of the month.

For example:

- The string 0 0 13 * * ? indicates that you want to schedule the processor to run at 1:00 PM every day.
- The string 0 20 14 ? * MON-FRI indicates that you want to schedule the processor to run at 2:20 PM every Monday through Friday.
- The string 0 15 10 ? * 6L 2011-2017 indicates that you want to schedule the processor to run at 10:15 AM, on the last Friday of every month, between 2011 and 2017.

For additional information and examples, see the <https://www.quartz-scheduler.org/documentation/quartz-2.2.2/tutorials/tutorial-lesson-06.html> in the <https://www.quartz-scheduler.org/documentation/>.

Concurrent Tasks

Next, the Scheduling tab provides a configuration option named 'Concurrent Tasks'. This controls how many threads the Processor will use. Said a different way, this controls how many FlowFiles should be processed by this Processor at the same time. Increasing this value will typically allow the Processor to handle more data in the same amount of time. However, it does this by using system resources that then are not usable by other Processors. This essentially provides a relative weighting of Processors - it controls how much of the system's resources should be allocated to this Processor instead of other Processors. This field is available for most Processors. There are, however, some types of Processors that can only be scheduled with a single Concurrent task.

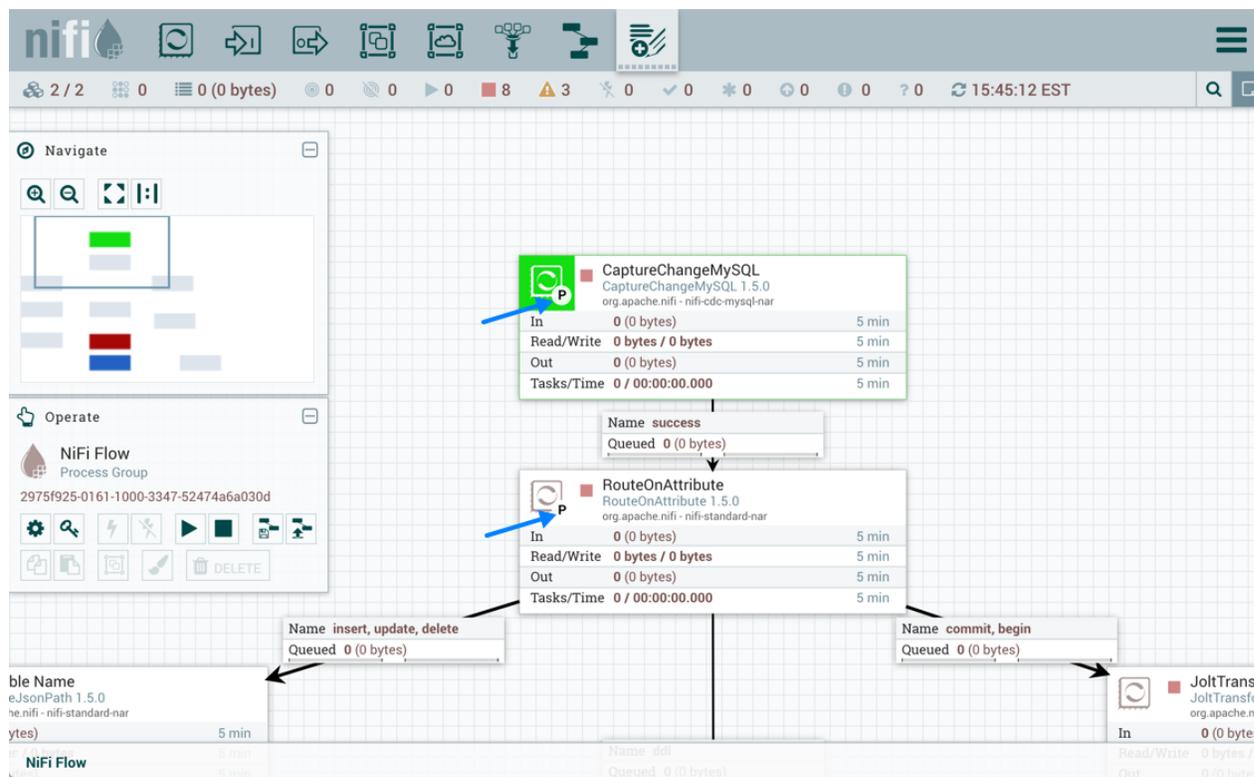
Run Schedule

The 'Run Schedule' dictates how often the Processor should be scheduled to run. The valid values for this field depend on the selected Scheduling Strategy (see above). If using the Event driven Scheduling Strategy, this field is not available. When using the Timer driven Scheduling Strategy, this value is a time duration specified by a number

followed by a time unit. For example, 1 second or 5 mins. The default value of 0 sec means that the Processor should run as often as possible as long as it has data to process. This is true for any time duration of 0, regardless of the time unit (i.e., 0 sec, 0 mins, 0 days). For an explanation of values that are applicable for the CRON driven Scheduling Strategy, see the description of the CRON driven Scheduling Strategy itself.

Execution

The Execution setting is used to determine on which node(s) the Processor will be scheduled to execute. Selecting 'All Nodes' will result in this Processor being scheduled on every node in the cluster. Selecting 'Primary Node' will result in this Processor being scheduled on the Primary Node only. Processors that have been configured for 'Primary Node' execution are identified by a "P" next to the processor icon:



To quickly identify 'Primary Node' processors, the "P" icon is also shown in the Processors tab on the Summary page:

NiFi Summary

PROCESSORS INPUT PORTS OUTPUT PORTS REMOTE PROCESS GROUPS CONNECTIONS PROCESS GROUPS

Displaying 22 of 22

Filter by name View: Single node Cluster

| Name | Type | Process Group | Run Status | In / Size 5 min | Read / Write 5 min | Out / Size 5 min | Tasks / Time 5 min |
|--------------------|--------------------|---------------|------------|-----------------|--------------------|------------------|--------------------|
| CaptureChangeMySQL | CaptureChangeMy... | NiFi Flow | Stopped | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| CaptureChangeMySQL | CaptureChangeMy... | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| EnforceOrder | EnforceOrder | NiFi Flow | Stopped | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| EnforceOrder | EnforceOrder | PG1 | Stopped | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| Get Table Name | Evaluate.JsonPath | NiFi Flow | Stopped | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| Get Table Name | Evaluate.JsonPath | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| JoltTransformJSON | JoltTransform.JSON | NiFi Flow | Stopped | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| JoltTransformJSON | JoltTransform.JSON | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| LogAttribute | LogAttribute | NiFi Flow | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| LogAttribute | LogAttribute | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| LogAttribute | LogAttribute | NiFi Flow | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| LogAttribute | LogAttribute | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| PutDatabaseRecord | PutDatabaseRecord | NiFi Flow | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| PutDatabaseRecord | PutDatabaseRecord | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| PutDatabaseRecord | PutDatabaseRecord | NiFi Flow | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| PutDatabaseRecord | PutDatabaseRecord | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| RouteOnAttribute | RouteOnAttribute | NiFi Flow | Stopped | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |
| RouteOnAttribute | RouteOnAttribute | PG1 | Invalid | 0 (0 bytes) | 0 bytes / 0 bytes | 0 (0 bytes) | 0 / 00:00:00.000 |

Last updated: 16:16:06 EST system diagnostics

Run Duration

The right-hand side of the Scheduling tab contains a slider for choosing the 'Run Duration'. This controls how long the Processor should be scheduled to run each time that it is triggered. On the left-hand side of the slider, it is marked 'Lower latency' while the right-hand side is marked 'Higher throughput'. When a Processor finishes running, it must update the repository in order to transfer the FlowFiles to the next Connection. Updating the repository is expensive, so the more work that can be done at once before updating the repository, the more work the Processor can handle (Higher throughput). However, this means that the next Processor cannot start processing those FlowFiles until the previous Process updates this repository. As a result, the latency will be longer (the time required to process the FlowFile from beginning to end will be longer). As a result, the slider provides a spectrum from which the DFM can choose to favor Lower Latency or Higher Throughput.

Properties Tab

The Properties tab provides a mechanism to configure Processor-specific behavior. There are no default properties. Each type of Processor must define which Properties make sense for its use case. Below, we see the Properties tab for a RouteOnAttribute Processor:

Configure Processor

SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field +

| Property | Value |
|---------------------------------|------------------------|
| Routing Strategy ? | Route to Property name |

CANCEL APPLY

This Processor, by default, has only a single property: 'Routing Strategy'. The default value is 'Route to Property name'. Next to the name of this property is a small question mark symbol (

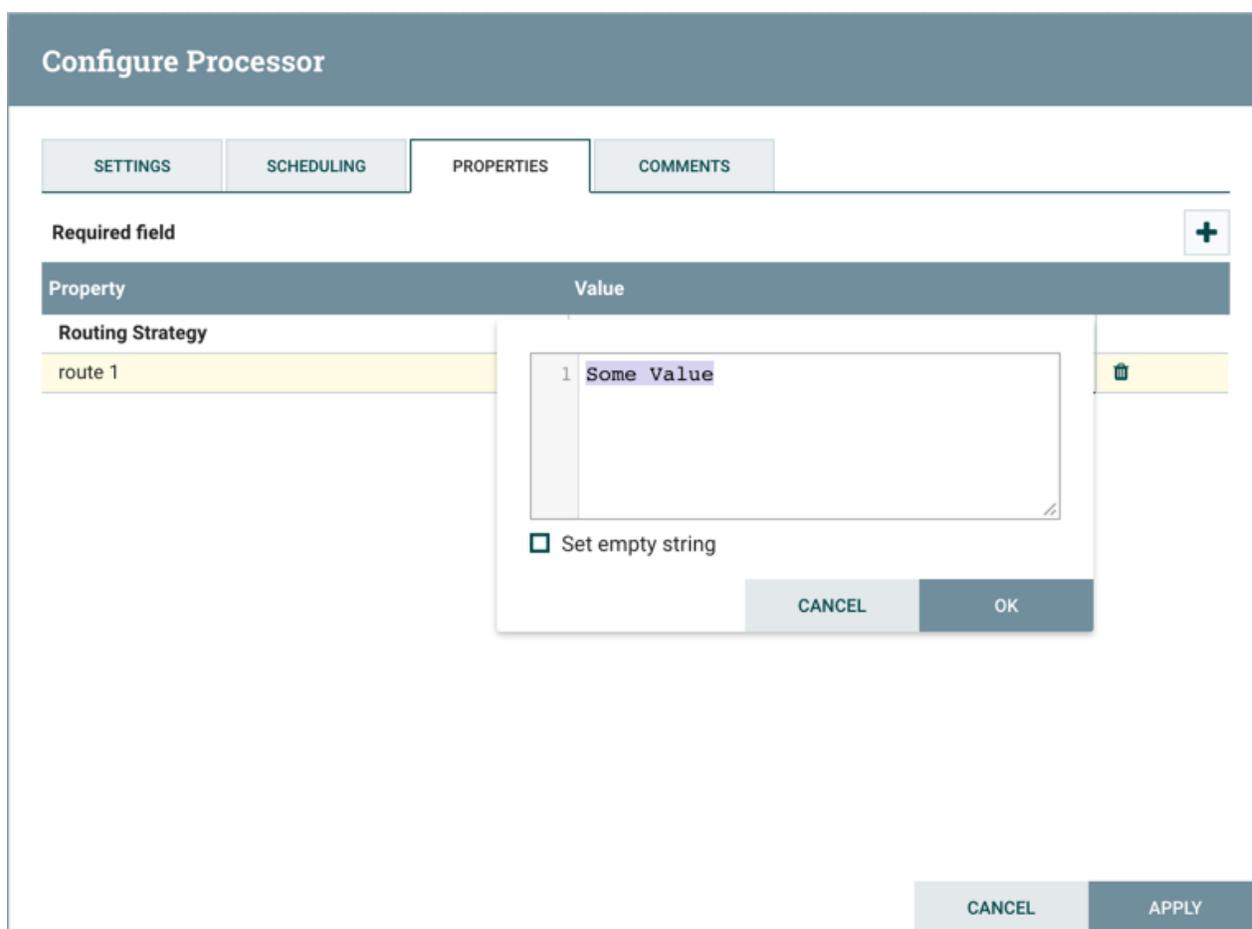


). This help symbol is seen in other places throughout the User Interface, and it indicates that more information is available. Hovering over this symbol with the mouse will provide additional details about the property and the default value, as well as historical values that have been set for the Property.

Clicking on the value for the property will allow a DFM to change the value. Depending on the values that are allowed for the property, the user is either provided a drop-down from which to choose a value or is given a text area to type a value:

The screenshot shows the 'Configure Processor' dialog with the 'PROPERTIES' tab selected. A dropdown menu is open for the 'Routing Strategy' property, showing four options: 'Route to Property name', 'Route to Property name', 'Route to 'matched' if all match', and 'Route to 'matched' if any matches'. The first option is highlighted. The dialog also features a '+', 'CANCEL', and 'APPLY' buttons.

In the top-right corner of the tab is a button for adding a New Property. Clicking this button will provide the DFM with a dialog to enter the name and value of a new property. Not all Processors allow User-Defined properties. In processors that do not allow them, the Processor becomes invalid when User-Defined properties are applied. RouteOnAttribute, however, does allow User-Defined properties. In fact, this Processor will not be valid until the user has added a property.



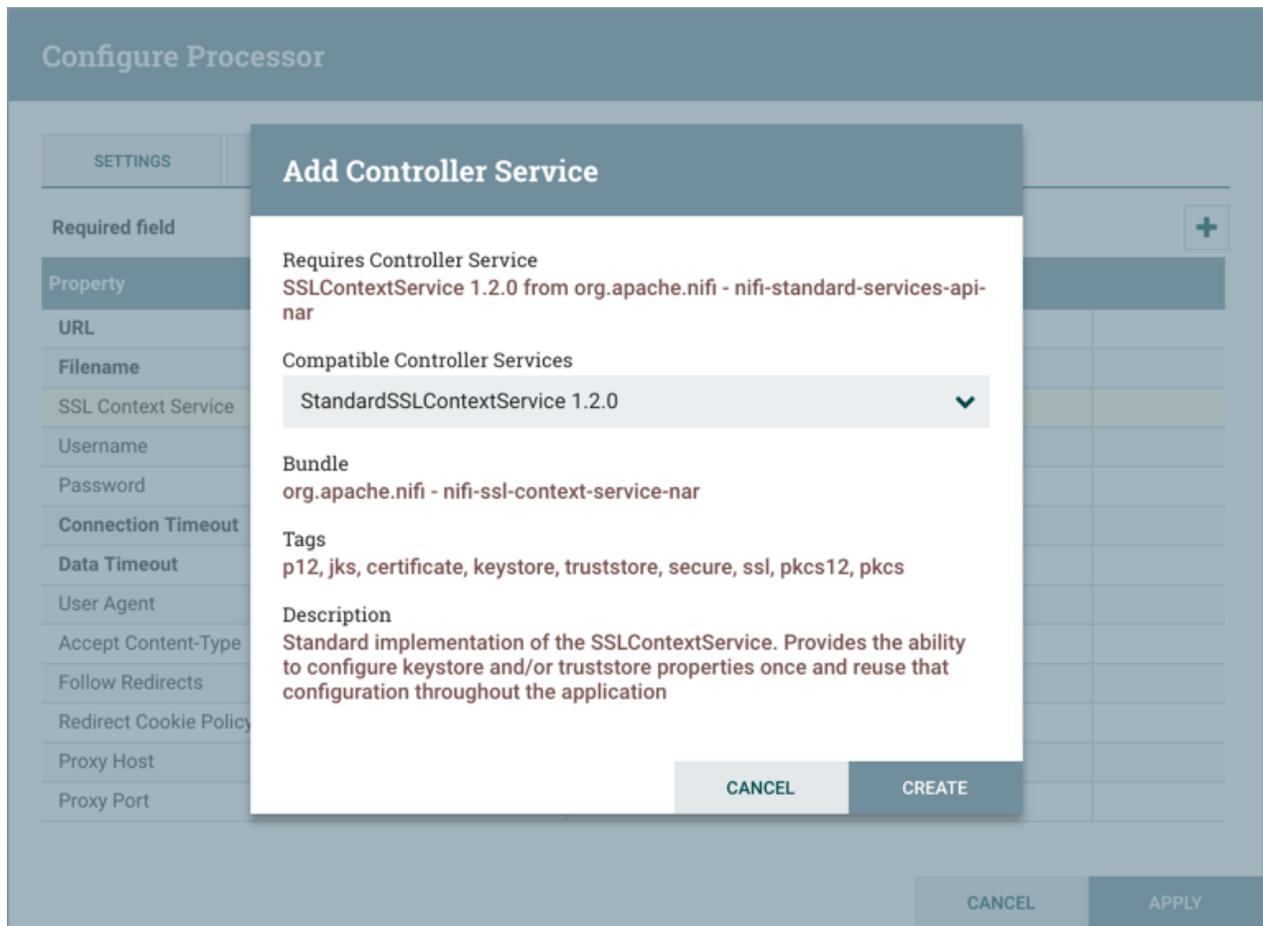
Note that after a User-Defined property has been added, an icon will appear on the right-hand side of that row (



). Clicking it will remove the User-Defined property from the Processor.

Some processors also have an Advanced User Interface (UI) built into them. For example, the UpdateAttribute processor has an Advanced UI. To access the Advanced UI, click the "Advanced" button that appears at the bottom of the Configure Processor window. Only processors that have an Advanced UI will have this button.

Some processors have properties that refer to other components, such as Controller Services, which also need to be configured. For example, the GetHTTP processor has an SSLContextService property, which refers to the StandardSSLContextService controller service. When DFMs want to configure this property but have not yet created and configured the controller service, they have the option to create the service on the spot, as depicted in the image below. For more information about configuring Controller Services, see the *Controller Services* section.



Comments Tab

The last tab in the Processor configuration dialog is the Comments tab. This tab simply provides an area for users to include whatever comments are appropriate for this component. Use of the Comments tab is optional:



The image shows a screenshot of the 'Configure Processor' dialog box in Apache NiFi. The dialog has a title bar 'Configure Processor' and four tabs: 'SETTINGS', 'SCHEDULING', 'PROPERTIES', and 'COMMENTS'. The 'COMMENTS' tab is selected, showing a large empty text area. At the bottom right, there are 'CANCEL' and 'APPLY' buttons.

Additional Help

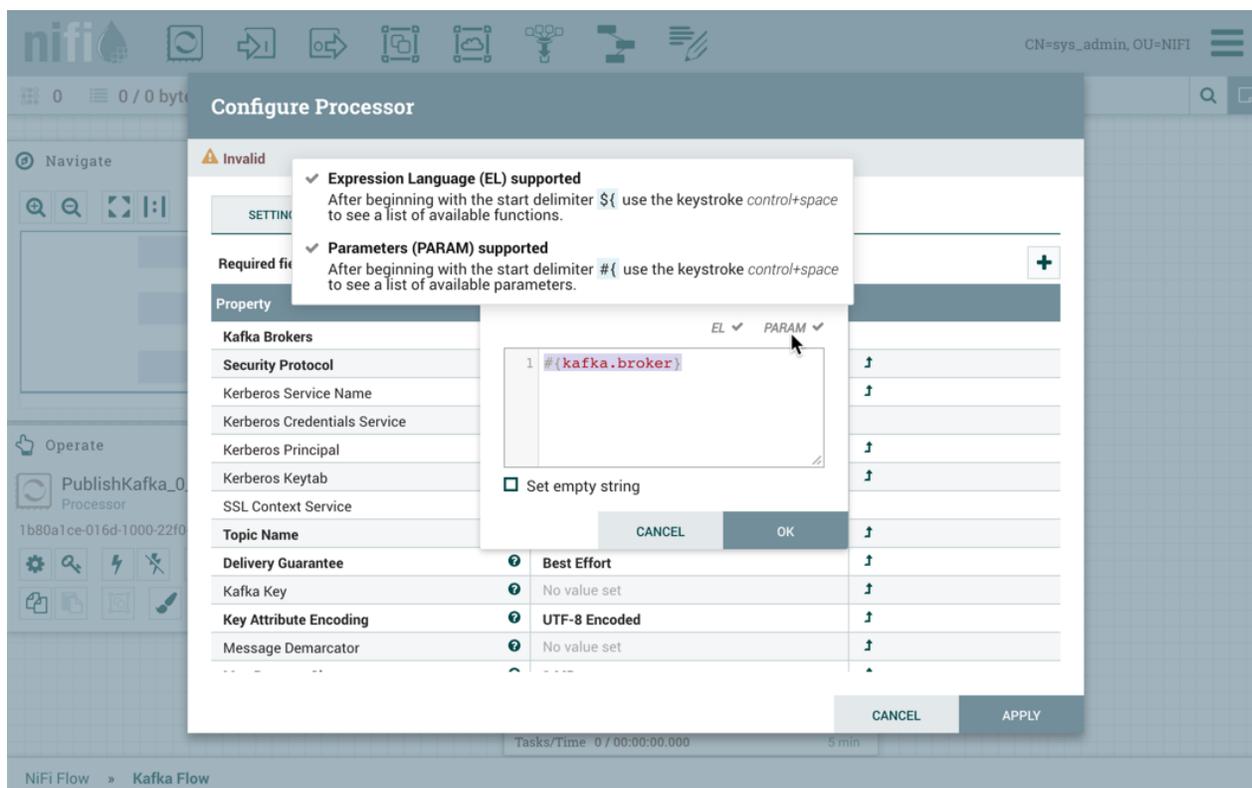
You can access additional documentation about each Processor's usage by right-clicking on the Processor and selecting 'Usage' from the context menu. Alternatively, select Help from the Global Menu in the top-right corner of the UI to display a Help page with all of the documentation, including usage documentation for all the Processors that are available. Click on the desired Processor to view usage documentation.

Parameters

The values of properties in the flow, including sensitive properties, can be parameterized using Parameters. Parameters are created and configured within the NiFi UI. Any property can be configured to reference a Parameter with the following conditions:

- A sensitive property can only reference a Sensitive Parameter
- A non-sensitive property can only reference a Non-Sensitive Parameter
- Properties that reference Controller Services can not use Parameters
- Parameters cannot be referenced in Reporting Tasks or in controller-level Controller Services

The UI indicates whether a Parameter can be used for a property value.



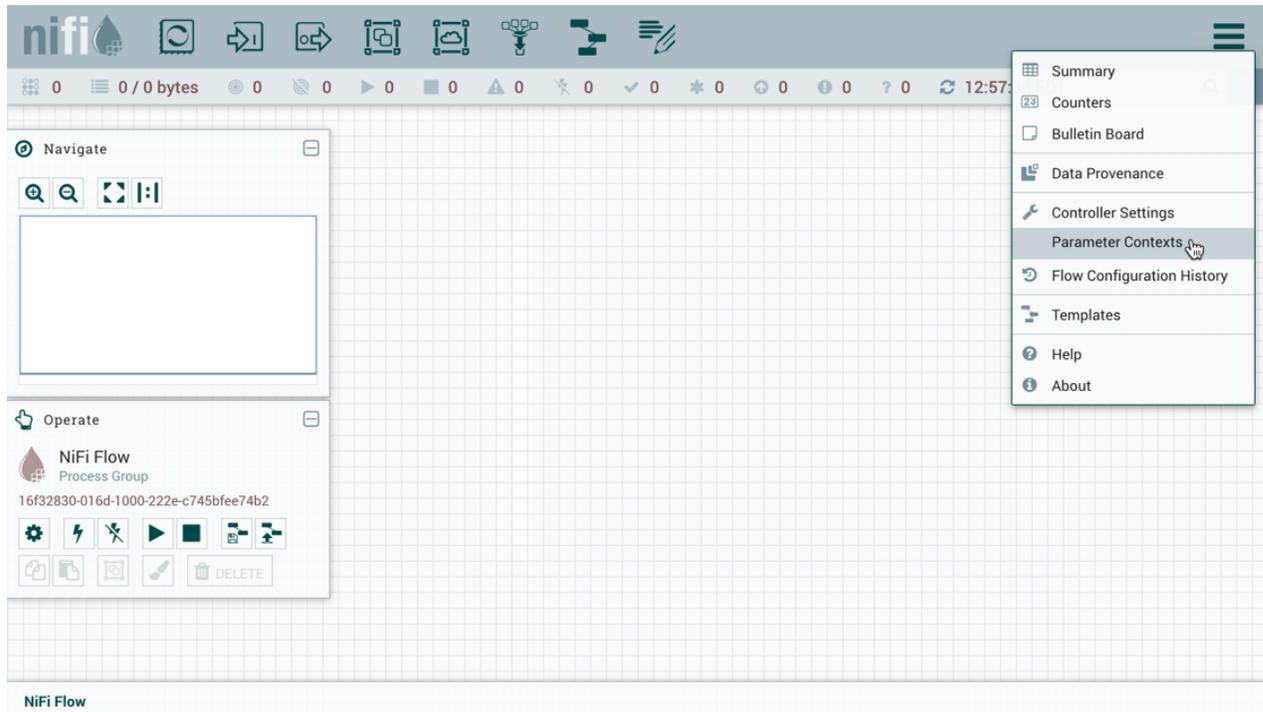
Note: Parameters have numerous advantages over Variables. In addition to sensitive value support, Parameters offer more granular control over access policies. Additionally, properties that reference Parameters are validated against the substituted value, unlike most properties that reference Variables using Expression Language.

Parameter Contexts

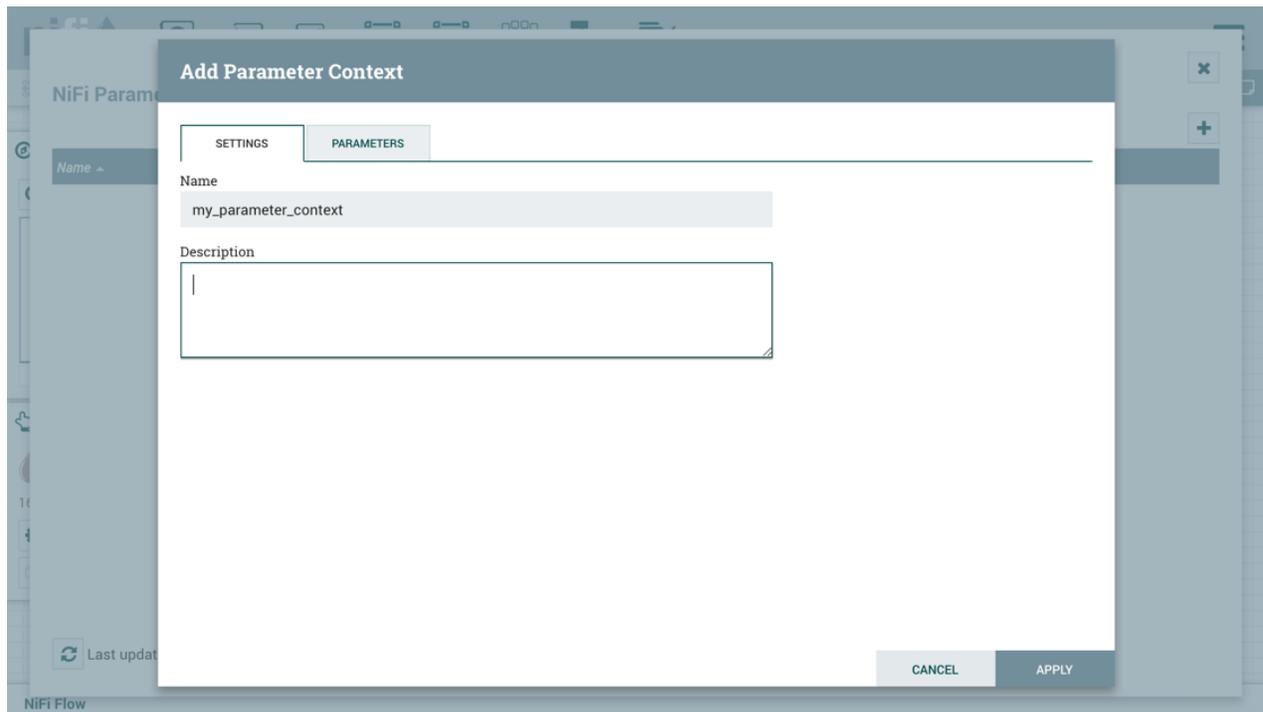
Parameters are created within Parameter Contexts. Parameter Contexts are globally defined/accessible to the NiFi instance. Access policies can be applied to Parameter Contexts to determine which users can create them. Once created, policies to read and write to a specific Parameter Context can also be applied (see *Accessing Parameters* for more information).

Creating a Parameter Context

To create a Parameter Context, select Parameter Contexts from the Global Menu:



In the Parameter Contexts window, click the + button in the upper-right corner and the Add Parameter Context window opens. The window has two tabs: Settings and Parameters.

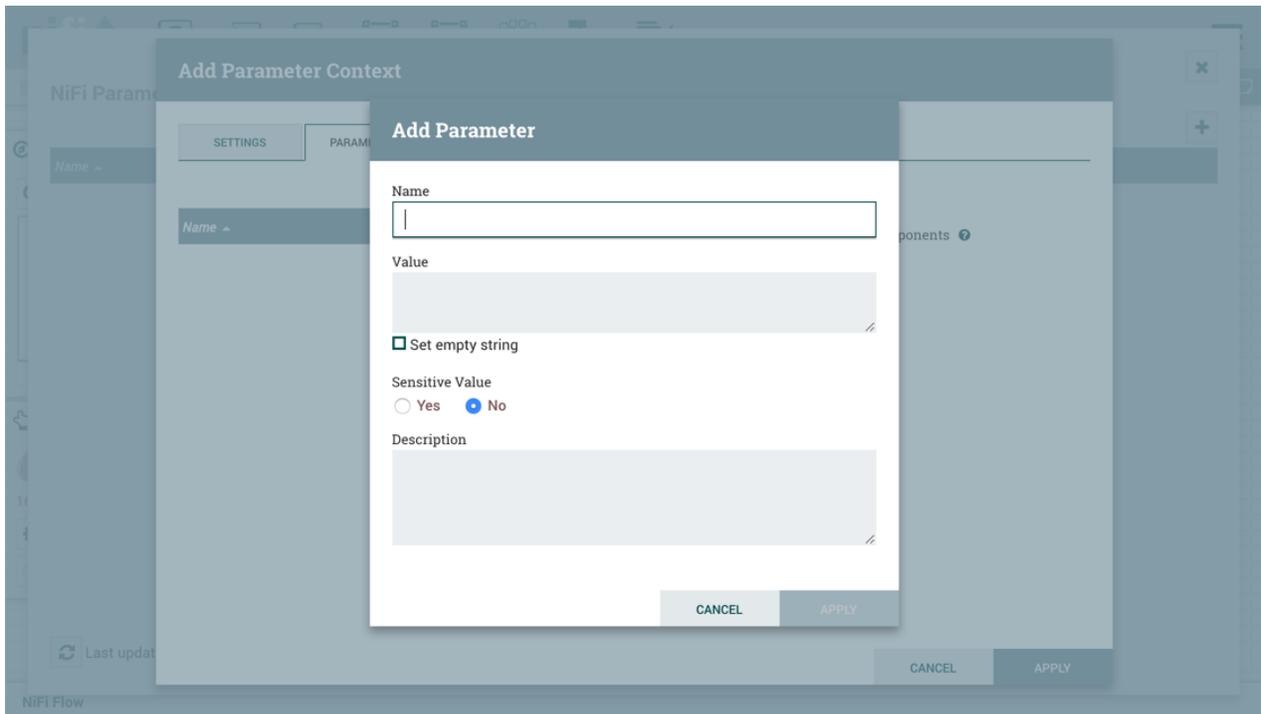


On the "Settings" tab, add a name for the Parameter Context and a description if desired. Select "Apply" to save the Parameter Context or select the "Parameters" tab to add parameters to the context.

Adding a Parameter to a Parameter Context

Parameters can be added during Parameter Context creation or added to existing Parameter Contexts.

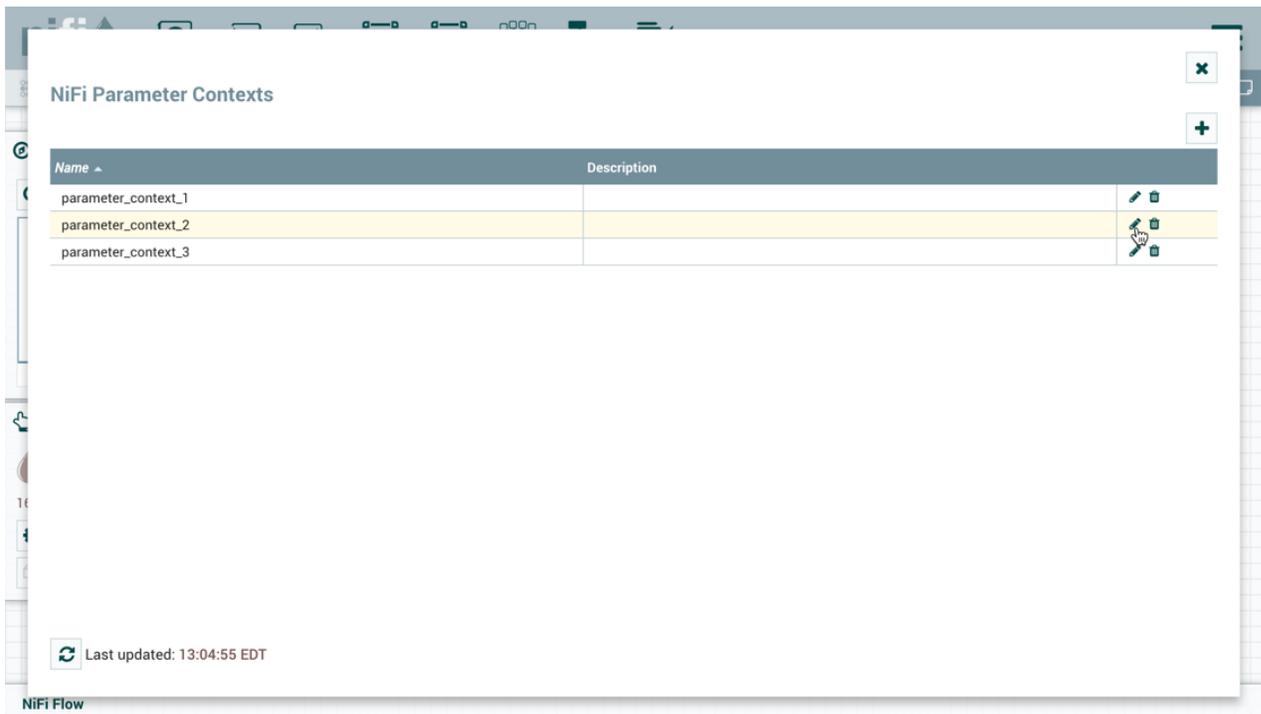
During Parameter Context creation, select the "Parameters" tab. Click the + button to open the Add Parameter window.



To add parameters to an existing Parameter Context, open the Parameter Context window and click the Edit button



in the row of the desired Parameter Context.

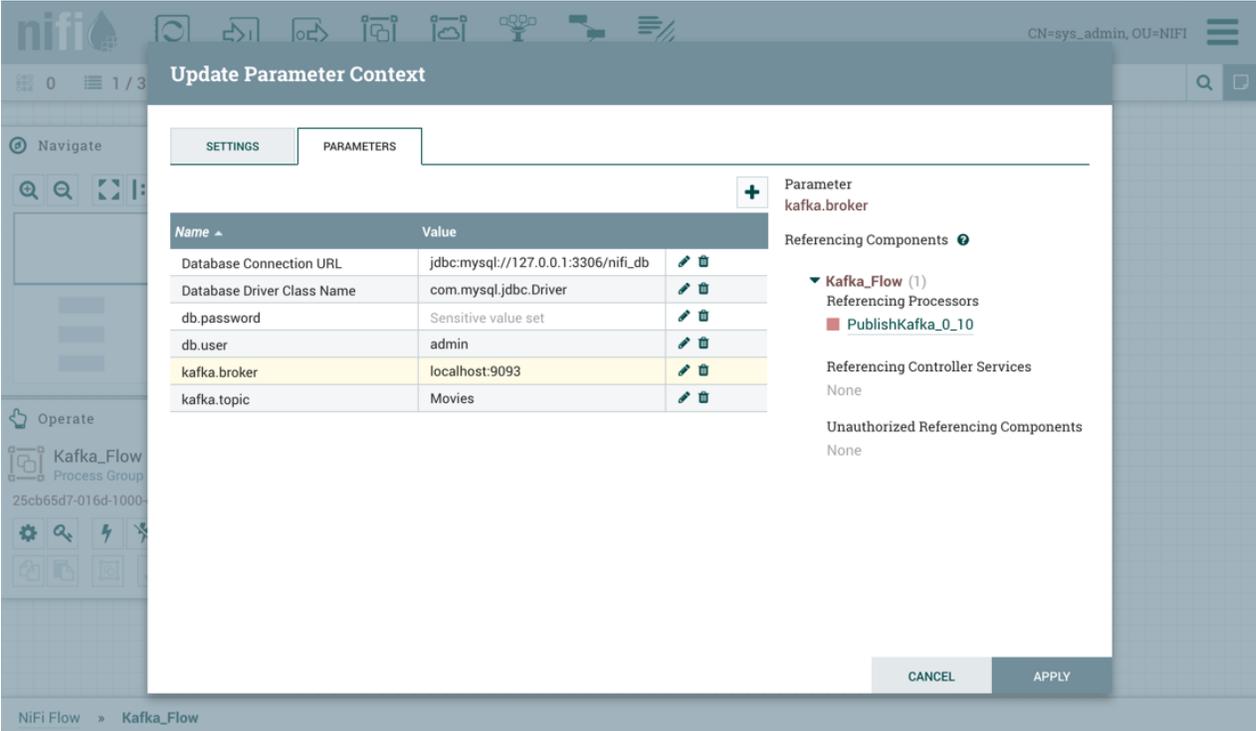


On the "Parameters" tab, click the + button to open the Add Parameter window.

The Add Parameter window has the following settings:

- Name - A name that is used to denote the Parameter. Only alpha-numeric characters (a-z, A-Z, 0-9), hyphens (-), underscores (_), periods (.), and spaces are allowed.
- Value - The value that will be used when the Parameter is referenced. Parameter values do not support Expression Language or embedded parameter references.
- Set empty string - Check to explicitly set the value of the Parameter to an empty string. Unchecked by default. (Note: If checked but a value is set, the checkbox is ignored.)
- Sensitive Value - Set to "Yes" if the Parameter's Value should be considered sensitive. If sensitive, the value of the Parameter will not be shown in the UI once applied. The default setting is "No". Sensitive Parameters can only be referenced by sensitive properties and Non-Sensitive Parameters by non-sensitive properties. Once a Parameter is created, its sensitivity flag cannot be changed.
- Description - A description that explains what the Parameter is, how it is to be used, etc. This field is optional.

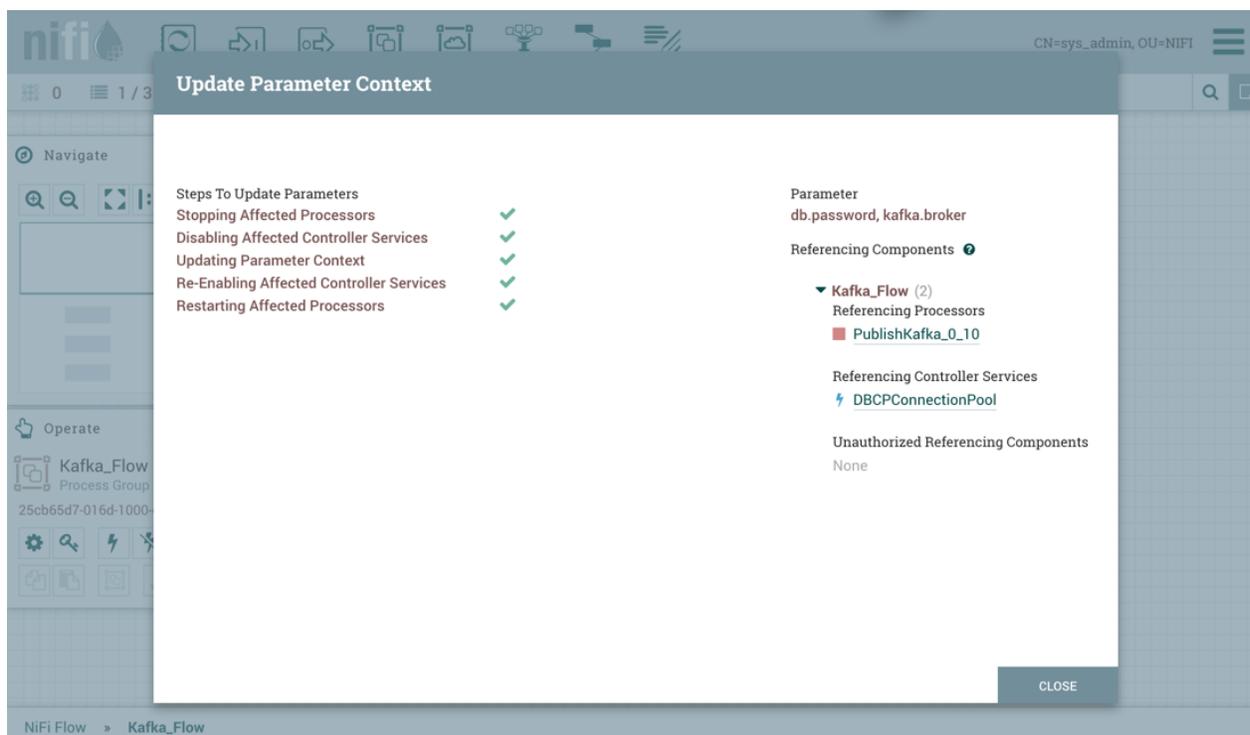
Once these settings are configured, select "Apply". The Referencing Components lists the components referenced by the currently selected parameter. Add additional Parameters or edit any existing Parameters.



The screenshot shows the 'Update Parameter Context' dialog in Apache NiFi. The dialog has two tabs: 'SETTINGS' and 'PARAMETERS'. The 'PARAMETERS' tab is active, displaying a table of parameters. The table has columns for 'Name', 'Value', and edit/delete icons. The 'kafka.broker' parameter is highlighted in yellow. To the right of the table, there is a section for 'Referencing Components' which lists 'Kafka_Flow (1)' and 'Referencing Processors' including 'PublishKafka_0_10'. Below this, it shows 'Referencing Controller Services' as 'None' and 'Unauthorized Referencing Components' as 'None'. At the bottom right of the dialog are 'CANCEL' and 'APPLY' buttons.

| Name | Value | |
|----------------------------|-------------------------------------|---|
| Database Connection URL | jdbc:mysql://127.0.0.1:3306/nifi_db |   |
| Database Driver Class Name | com.mysql.jdbc.Driver |   |
| db.password | Sensitive value set |   |
| db.user | admin |   |
| kafka.broker | localhost:9093 |   |
| kafka.topic | Movies |   |

To complete the process, select "Apply" from the Parameter Context window. The following operations are performed to validate all components that reference the added or modified parameters: Stopping/Restarting affected Processors, Disabling/Re-enabling affected Controller Services, Updating Parameter Context.



The Referencing Components section now lists an aggregation of all the components referenced by the set of parameters added/edited/deleted, organized by process group.

Assigning a Parameter Context to a Process Group

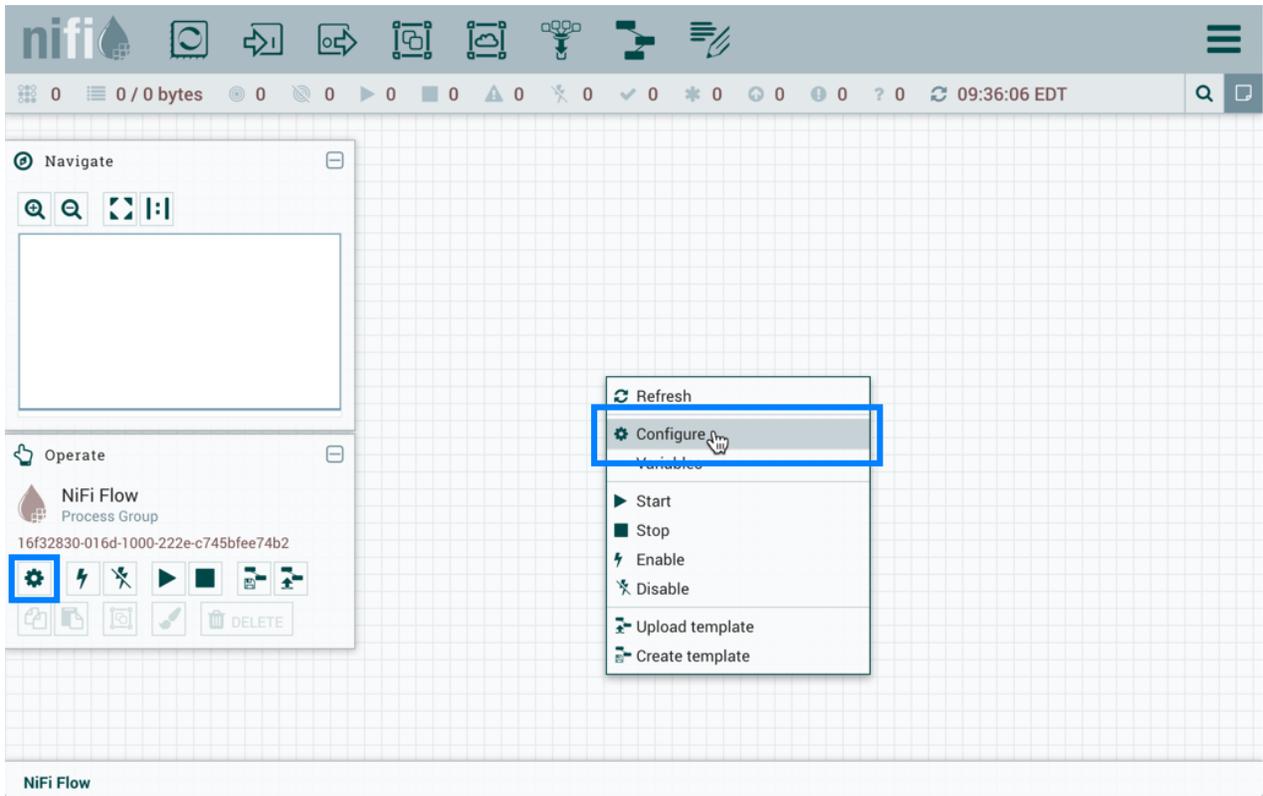
For a component to reference a Parameter, its Process Group must first be assigned a Parameter Context. Once assigned, processors and controller services within that Process Group may only reference Parameters within that Parameter Context.

A Process Group can only be assigned one Parameter Context, while a given Parameter Context can be assigned to multiple Process Groups.

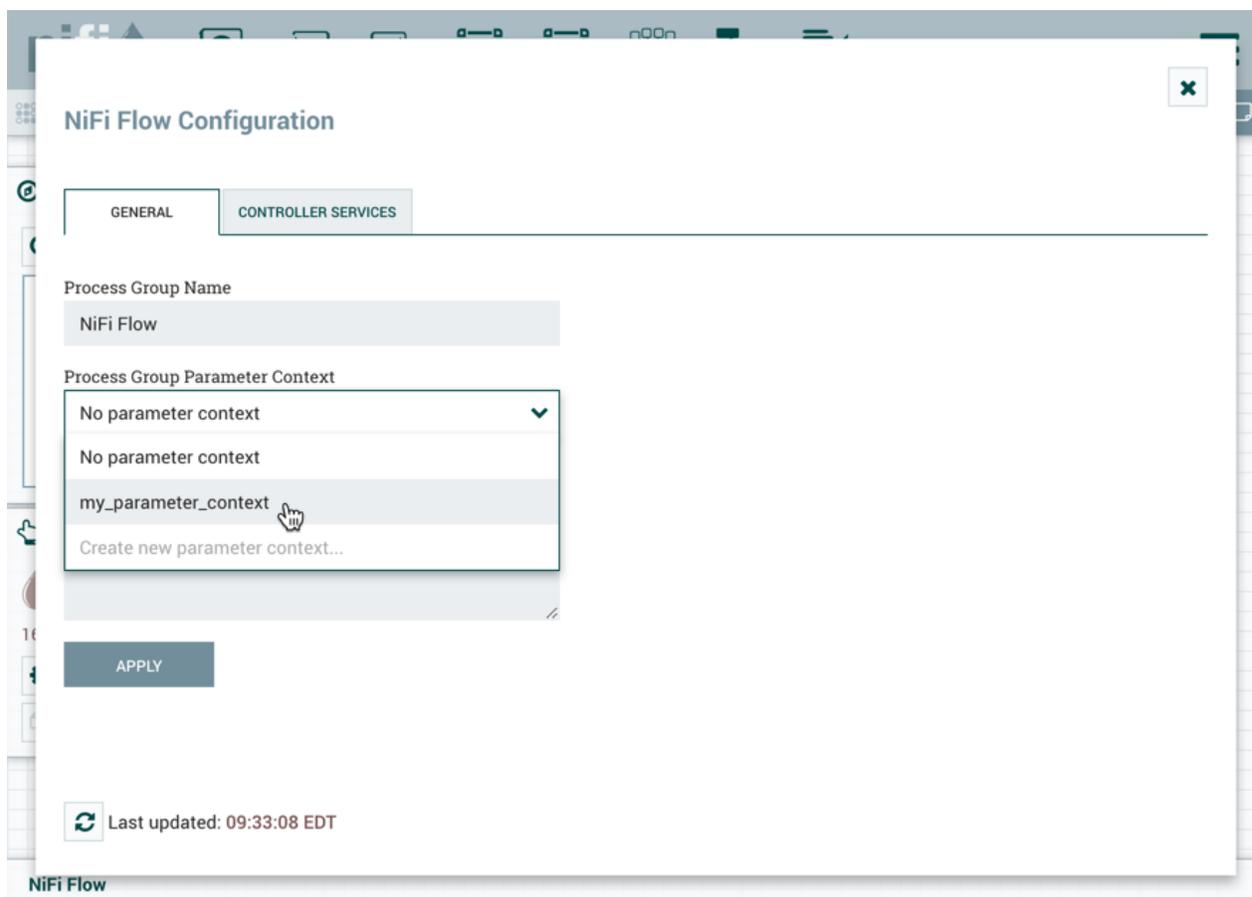


Note: A user can only set the Parameter Context of a Process Group to one of the Parameter Contexts that the user has the view policy for. Additionally, in order to set the Parameter Context, the user must have the modify policy for the Process Group. See [Accessing Parameters](#) for more information.

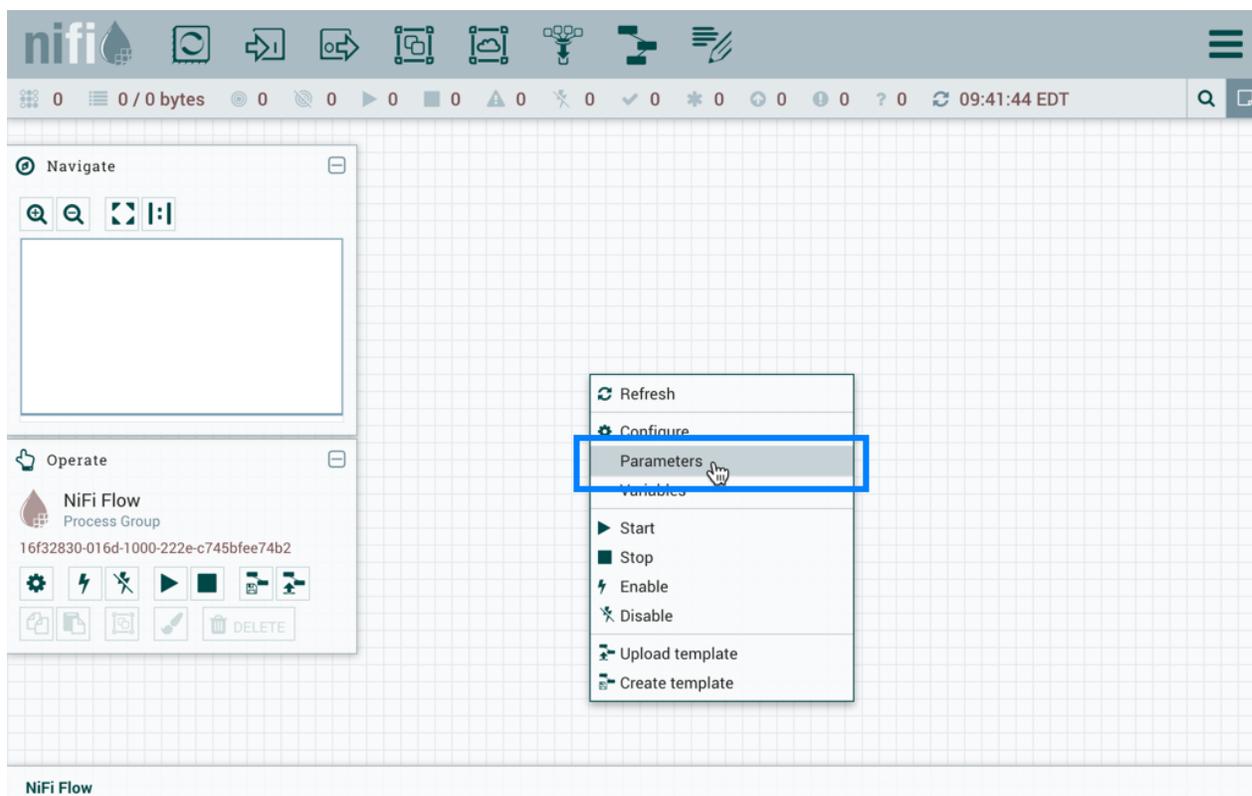
To assign a Parameter Context to a Process Group, click **Configure**, either from the Operate Palette or from the Process Group context menu.



In the Flow Configuration window, select the "General" tab. From the Process Group Parameter Context drop-down menu, select an existing Parameter Context or create a new one.



Select "Apply" to save the configuration changes. The Process Group context menu now includes a "Parameters" option which allows quick access to the Update Parameter Context window for the assigned Parameter Context.



If the Parameter Context for a Process Group is changed, all components that reference any Parameters in that Process Group will be stopped, validated, and restarted assuming the components were previously running and are still valid.



Note: If a Parameter Context is unset from a Process Group, it does NOT inherit the Parameter Context from the parent Process Group. Instead, no Parameters can be referenced. Any component that does already reference a Parameter will become invalid.

Referencing Parameters

Parameter Reference Syntax

To configure an eligible property to reference a Parameter, use the # symbol as the start, with the Parameter's name enclosed in curly braces:

`#{Parameter.Name}`

This can be escaped using an additional # character at the beginning. To illustrate this, assume that the Parameter abc has a value of xxx and Parameter def has a value of yyy. Then, the following user-defined property values will evaluate to these effective values:

| User-Entered Literal Property Value | Effective Property Value | Explanation |
|-------------------------------------|--------------------------|--|
| <code>#{abc}</code> | xxx | Simple substitution |
| <code>#{abc}/data</code> | xxx/data | Simple substitution with additional literal data |
| <code>#{abc}/#{def}</code> | xxx/yyy | Multiple substitution with additional literal data |
| <code>#{abc}</code> | <code>#{abc}</code> | No { } for parameter replacement |
| <code>#abc</code> | <code>#abc</code> | No { } for parameter replacement |
| <code>##{abc}</code> | <code>#{abc}</code> | Escaped # for literal interpretation |

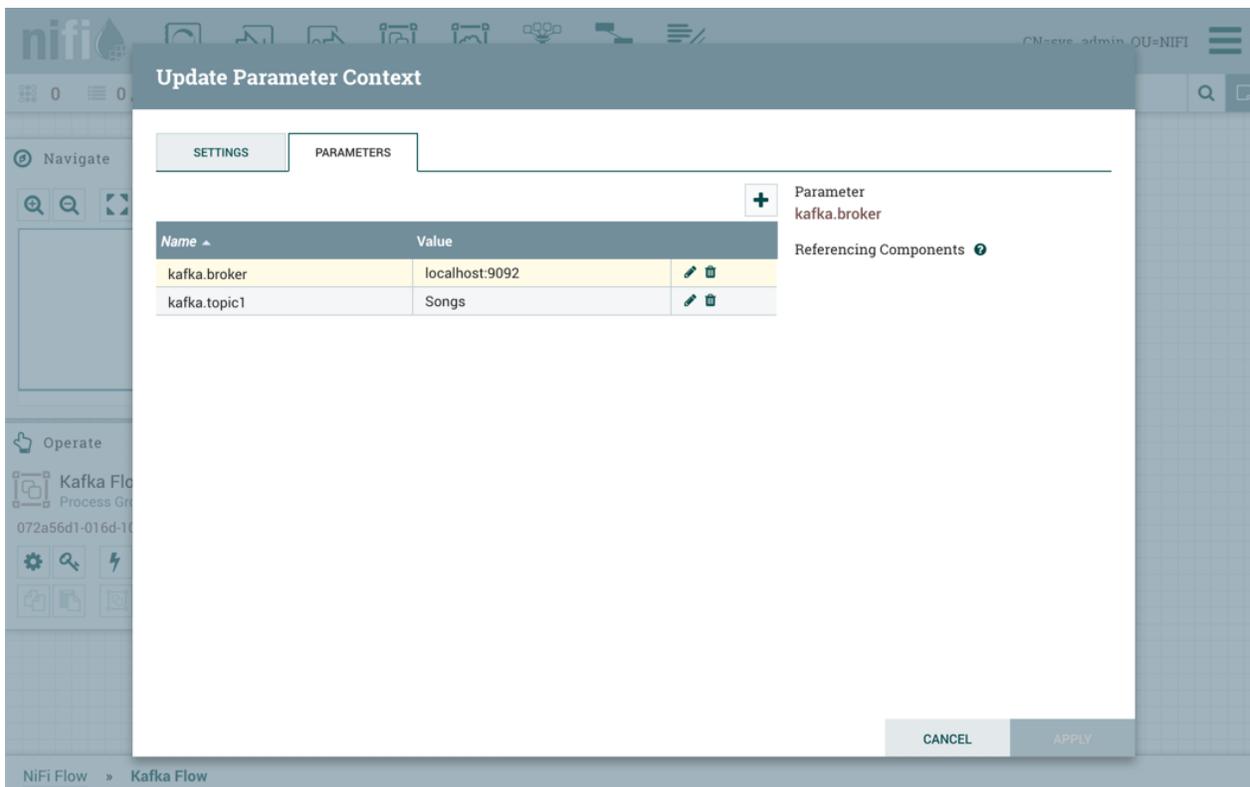
| | | |
|-------------|--|--|
| ###{abc} | #xxx | Escaped # for literal interpretation, followed by simple substitution |
| ####{abc} | #{abc} | Escaped # for literal interpretation, twice |
| #####{abc} | ##xxx | Escaped # for literal interpretation, twice, followed by simple substitution |
| #{abc/data} | Exception thrown on property set operation | / not a valid parameter name character |

When referencing a Parameter from within *Expression Language*, the Parameter reference is evaluated first. As an example, to replace xxx with zzz for the abc Parameter:

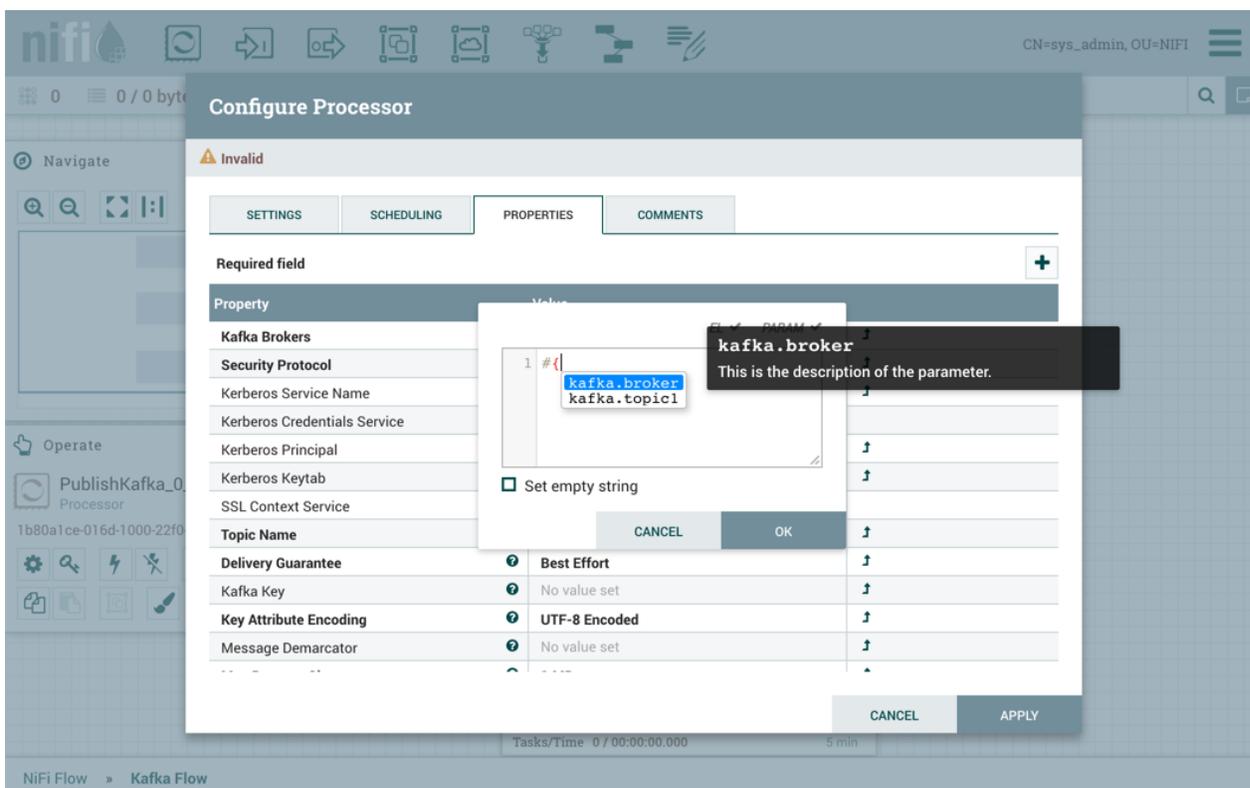
```
$ { #{abc}:replace('xxx', 'zzz') }
```

Referencing and Creating Parameters During Component Configuration

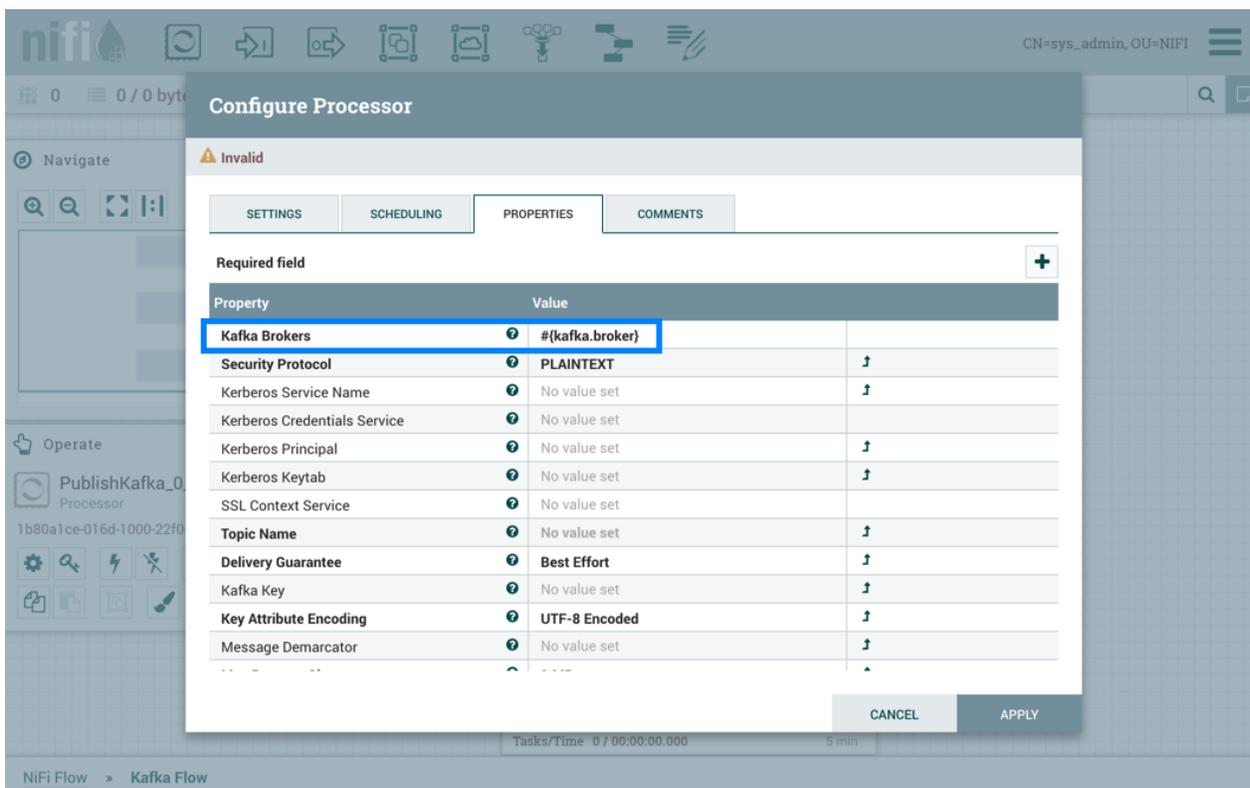
Parameters can be easily referenced or created as you configure the components in your flow. For example, assume a process group has the Parameter Context "Kafka Settings" assigned to it. "Kafka Settings" contains the parameters kafka.broker and kafka.topic1.



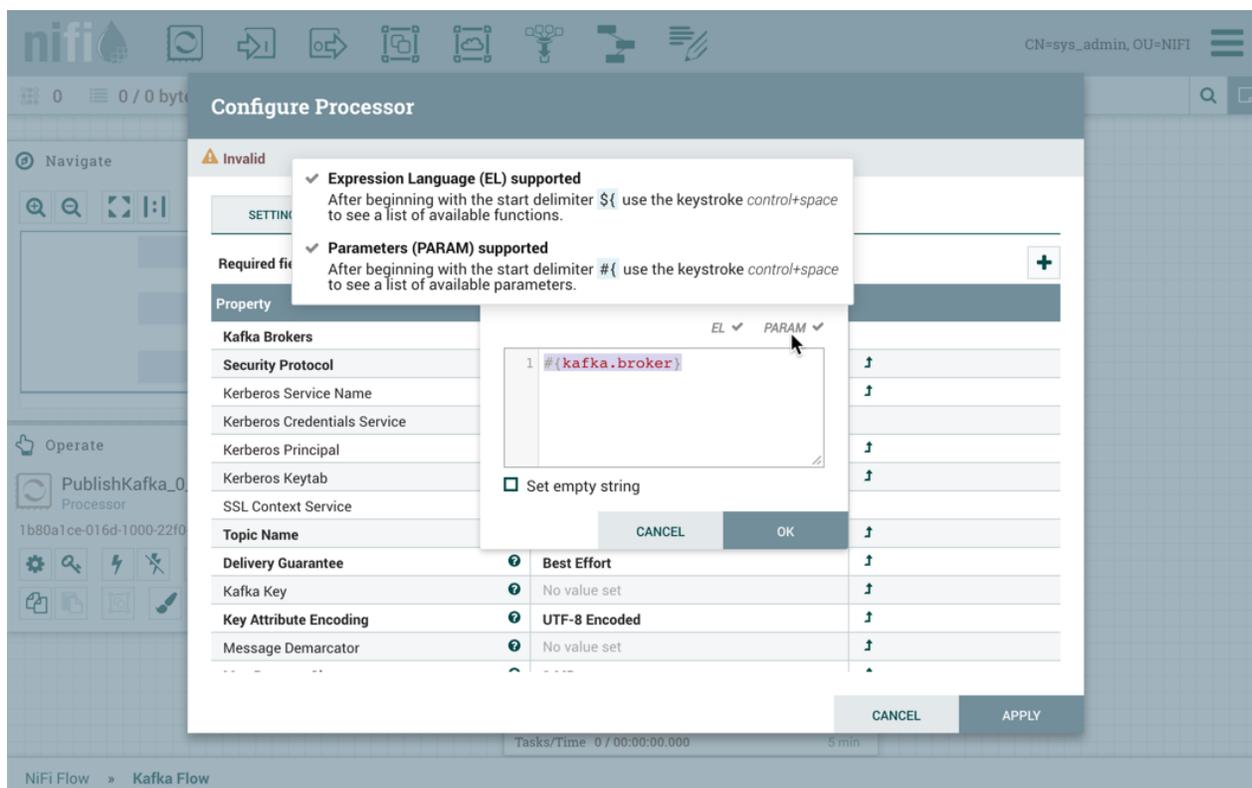
To reference kafka.broker as the value for the "Kafka Brokers" property in the PublishKafka processor, clear the default value and begin a new entry with the start delimiter #{. Next use the keystroke control+space to show the list of available parameters:



Select kafka.broker and complete the entry with a closing curly brace }.



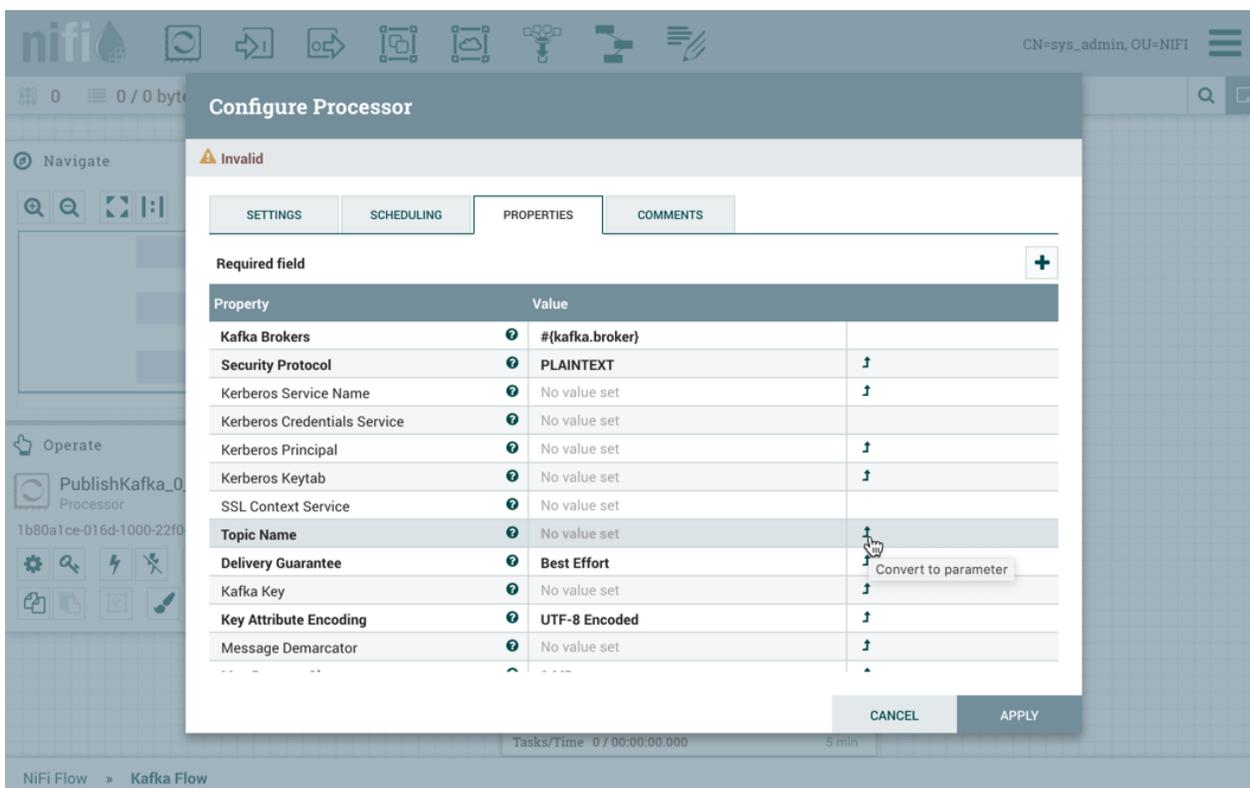
Help text describing this process is displayed when hovering over the Expression Language and Parameters eligibility indicators.



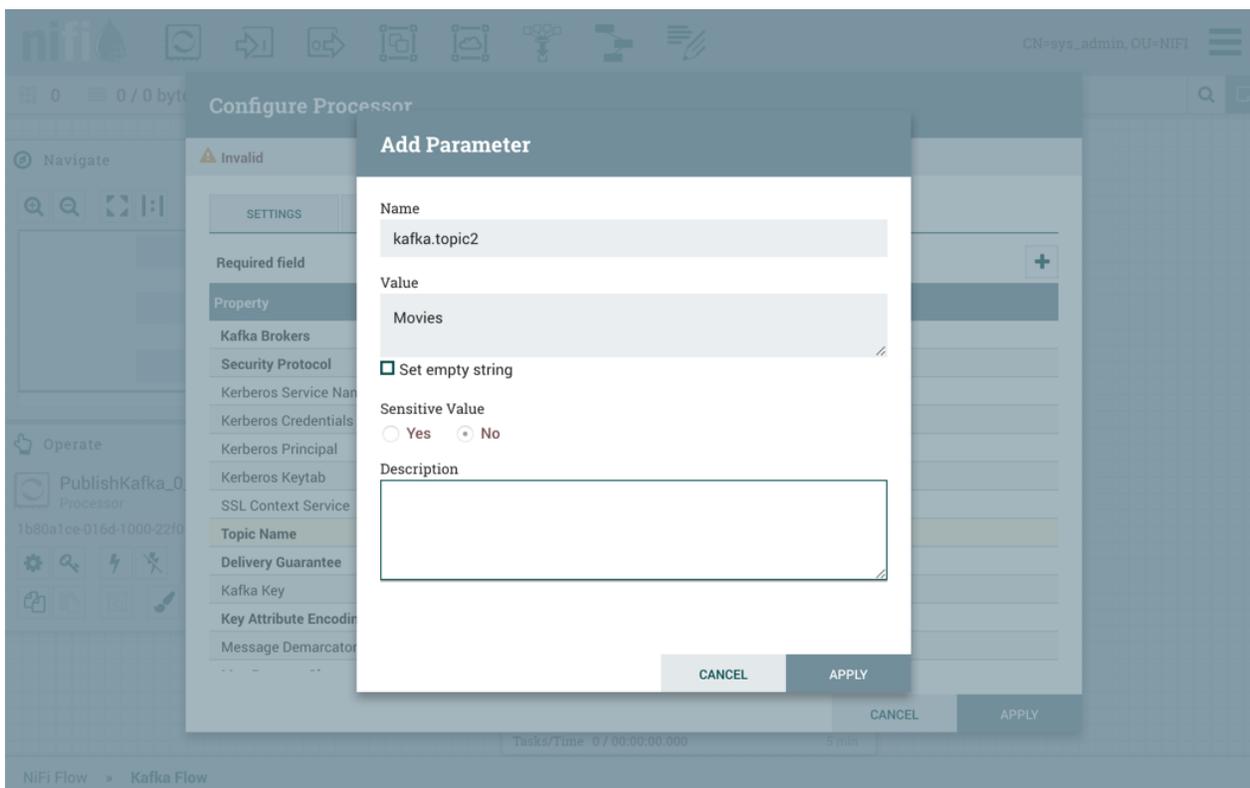
Parameters can also be created on the fly. For example, to create a parameter for the "Topic Name" property, select the "Convert to Parameter" icon



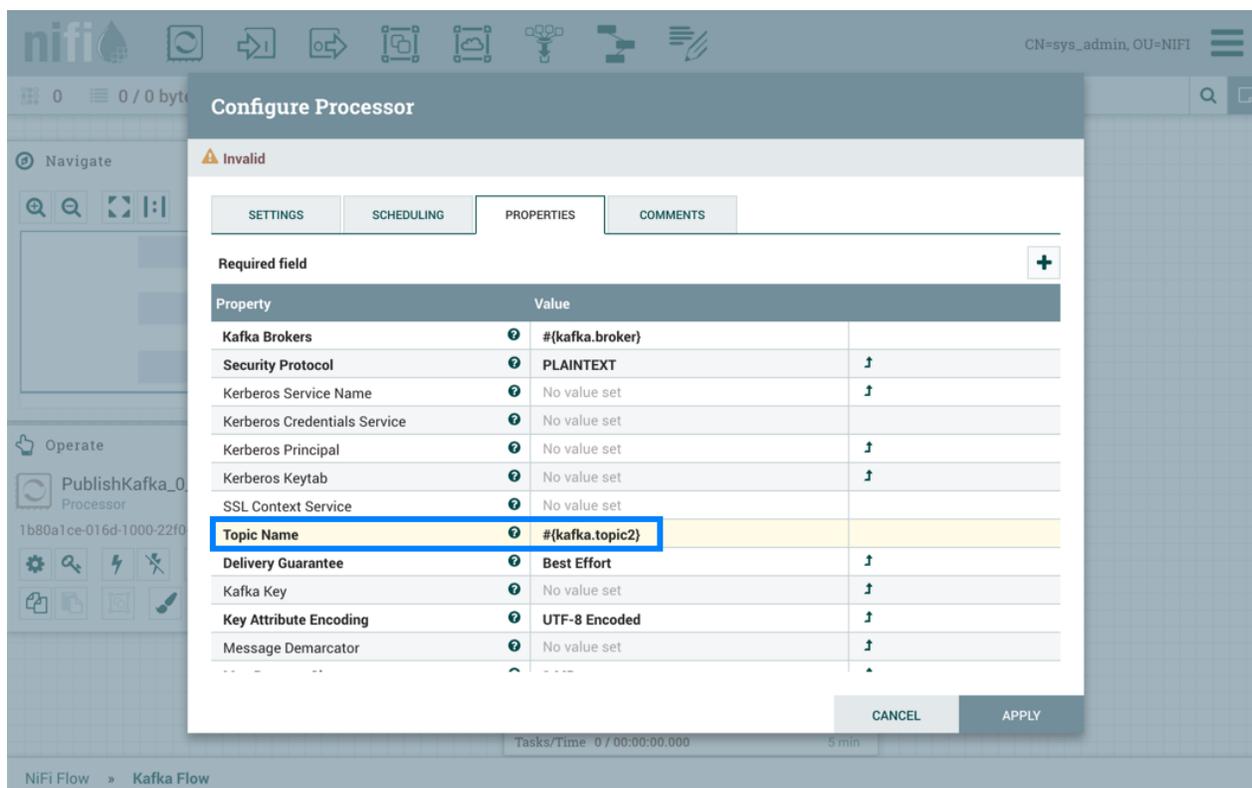
in that property's row. This icon will only be available if the user has appropriate permissions to modify the Parameter Context (see *Accessing Parameters* for more information).



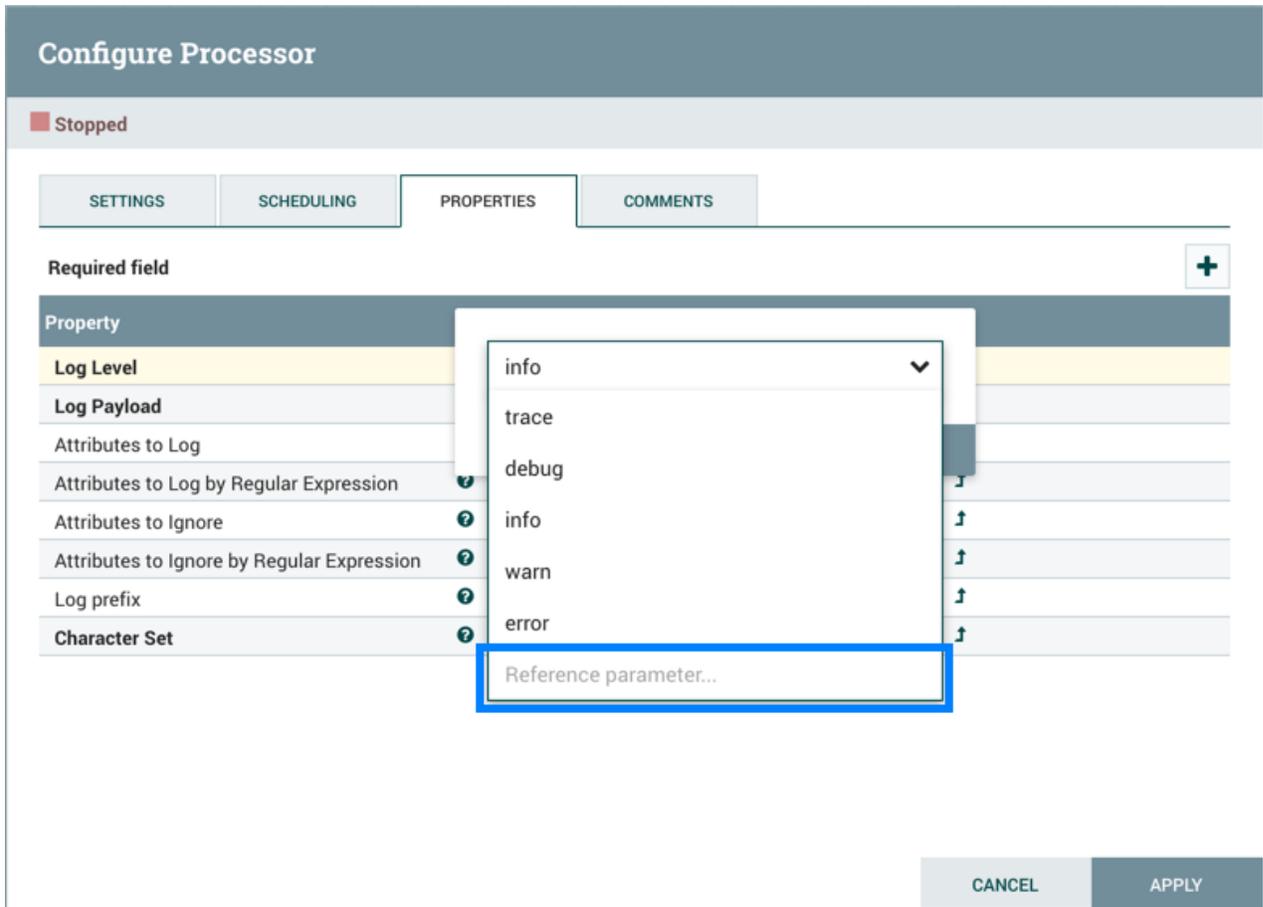
The Add Parameter dialog will open. Configure the new parameter as desired.



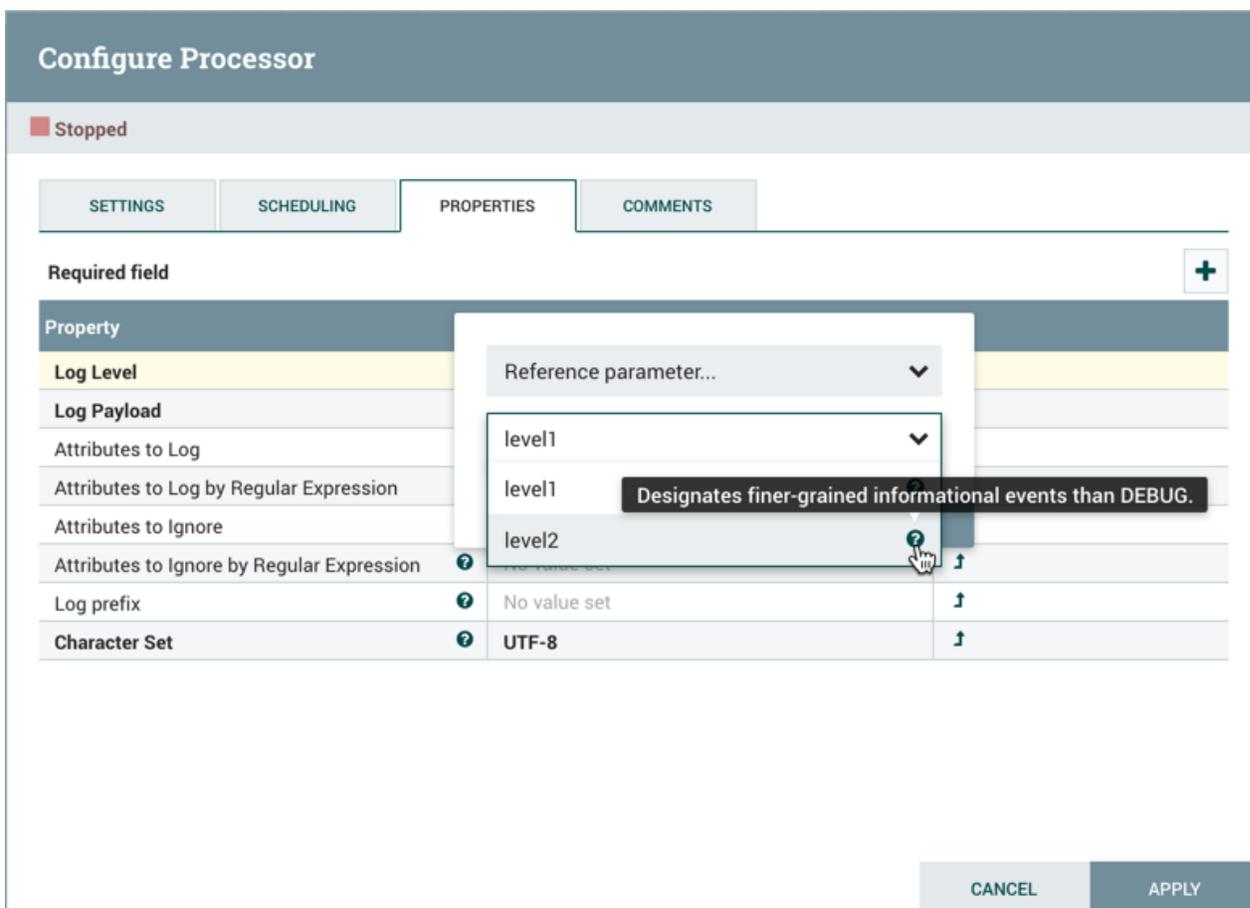
Select "Apply". The process group's Parameter Context will be updated and the new parameter will be referenced by the property with the proper syntax applied automatically.



Properties values that are selectable can also reference parameters. In addition to applying the "Convert to Parameter" method described earlier, the option "Reference parameter.." is available in the value drop-down menu.



Selecting "Reference parameter..." will display a drop-down list of available parameters, determined by the parameter context assigned to the component's process group and the user's access policies.



Hovering over the question mark icon

() displays the parameter's description.

Using Parameters with Sensitive Properties

Sensitive properties may only reference sensitive Parameters. This is important for *Versioning a DataFlow*. The value of the sensitive Parameter itself will NOT be sent to the flow registry, only the fact that the property references the sensitive Parameter. For more information see *Parameters in Versioned Flows*.

The value of a sensitive property must be set to a single Parameter reference. For example, values of `#{password}123` and `#{password}#{suffix}` are not allowed. Sending `#{password}123` would lead to exposing part of the sensitive property's value. This is in contrast to a non-sensitive property, where a value such as `#{path}/child/file.txt` is valid.

Accessing Parameters

User privileges to Parameters are managed via access policies on the following levels:

- Parameter Context
- Process Group
- Component



Note: For additional information on how to configure access policies, see the Access Policies section in the System Administrator's Guide.

Parameter Context Access Policies

For a user to see Parameter Contexts, they must be added to either the "access the controller" view policy or the "access parameter contexts" view policy. For a user to modify Parameter Contexts, they must also be added to the corresponding modify policies. These policies are accessed via "Policies" from the Global Menu. See the *System Administration documentation* in our *Reference* section for more information.

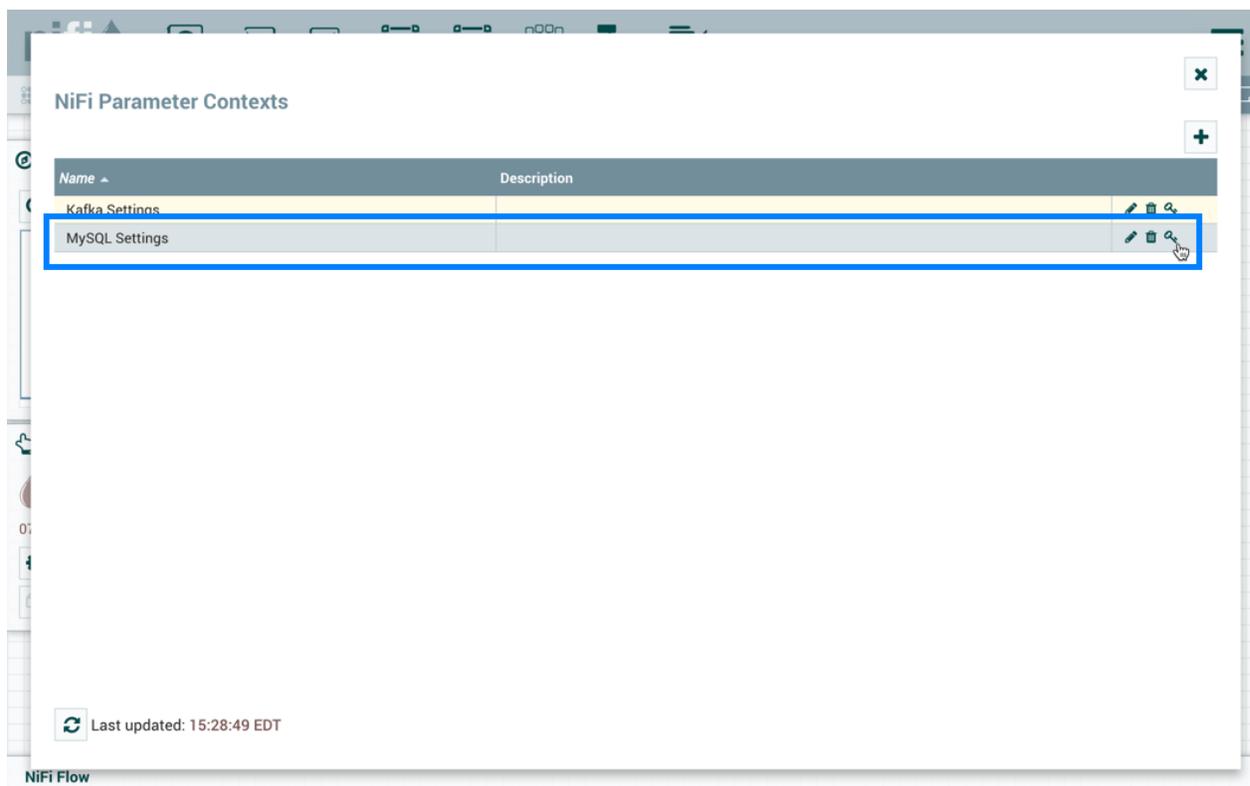


Note: The "access parameter contexts" policies are inherited from the "access the controller" policies unless overridden.

View and modify policies can also be set on individual parameter contexts to determine which users can view or add parameters to the context. Select "Parameter Contexts" from the Global Menu. Select the "Access Policies" button



in the row of the desired parameter context to manage these policies.



See the *System Administration documentation* in our *Reference* section for more information.

Process Group Access Policies

A user can only set the Parameter Context of a Process Group to one of the Parameter Contexts that the user has the view policy for. Additionally, in order to set the Parameter Context, the user must have the modify policy for the Process Group. The Process Group access policies can be managed by highlighting the Process Group and selecting the "Access Policies" button



from the Operate Palette.

Component Access Policies

To reference Parameters or convert properties to a Parameter in a component, a user needs to have the view and modify policies for the component. These policies are inherited if the user has view and modify policies to the component's process group, but these policies can be overridden on the component level.

In order to modify a Parameter, a user must have view and modify policies for any and all components that reference that Parameter. This is needed because changing the Parameter requires that the components be stopped/started and also because by taking that action, the user is modifying the behavior of the component.

See the *System Administration documentation* in our *Reference* section for more information.

Using Custom Properties with Expression Language

You can use NiFi Expression Language to reference FlowFile attributes, compare them to other values, and manipulate their values when you are creating and configuring dataflows. For more information on Expression Language, see the *Expression Language Reference*.

In addition to using FlowFile attributes, system properties, and environment properties within Expression Language, you can also define custom properties for Expression Language use. Defining custom properties gives you more flexibility in handling and processing dataflows. You can also create custom properties for connection, server, and service properties, for easier dataflow configuration.

NiFi properties have resolution precedence of which you should be aware when creating custom properties:

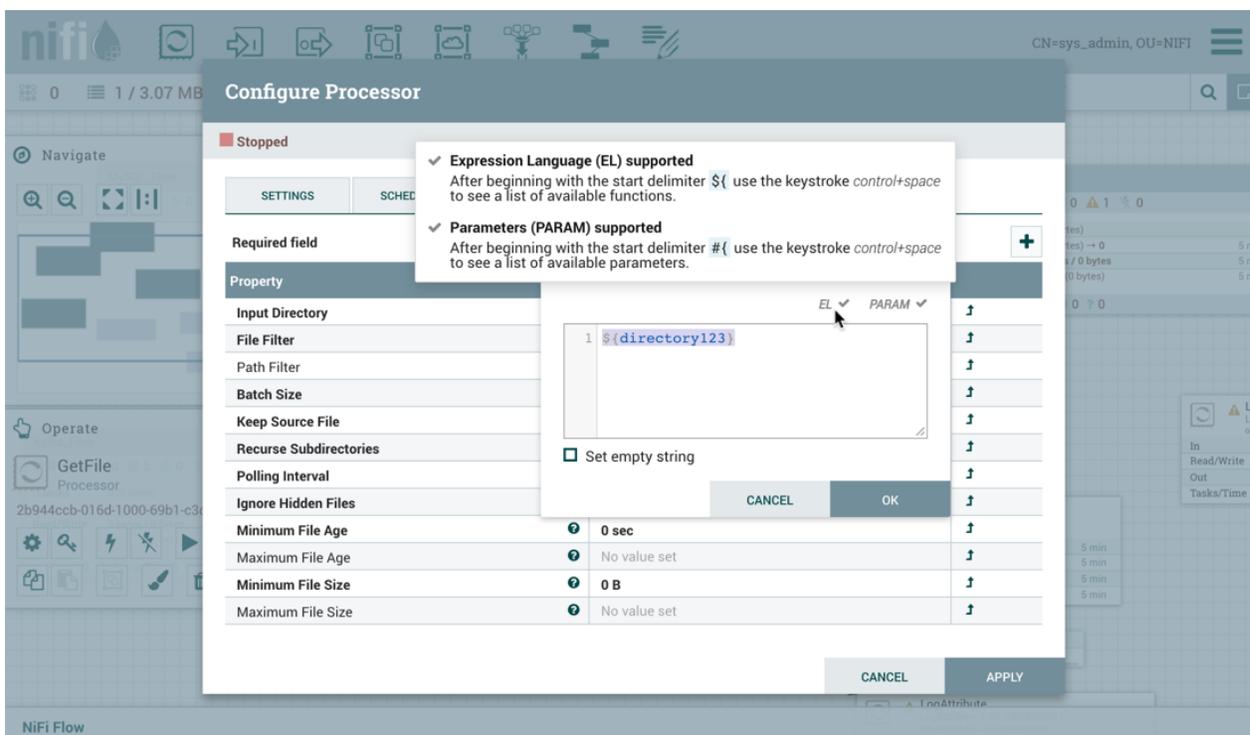
- Processor-specific attributes
- FlowFile properties
- FlowFile attributes
- From Variable Registry:
 - User defined properties (custom properties)
 - System properties
 - Operating System environment variables

When you are creating custom properties, ensure that each custom property contains a distinct property value, so that it is not overridden by existing environment properties, system properties, or FlowFile attributes.

There are two ways to use and manage custom properties with Expression Language:

- Variables: Variables are created and configured within the NiFi UI. They can be used in any field that supports Expression Language. Variables cannot be used for sensitive properties. NiFi automatically picks up new or modified variables. Variables are defined at the Process Group level, as a result, the access policies for viewing and changing variables are derived from the access policies of the Process Group. See *Variables* for more information.
- Custom Properties File: Key/value pairs are defined in a custom properties file that is referenced via the `nifi.variable.registry.properties` in `nifi.properties`. NiFi must be restarted for updates to be picked up. See *Referencing Custom Properties via nifi.properties* for more information.

Expression Language support for a property is indicated in the UI.

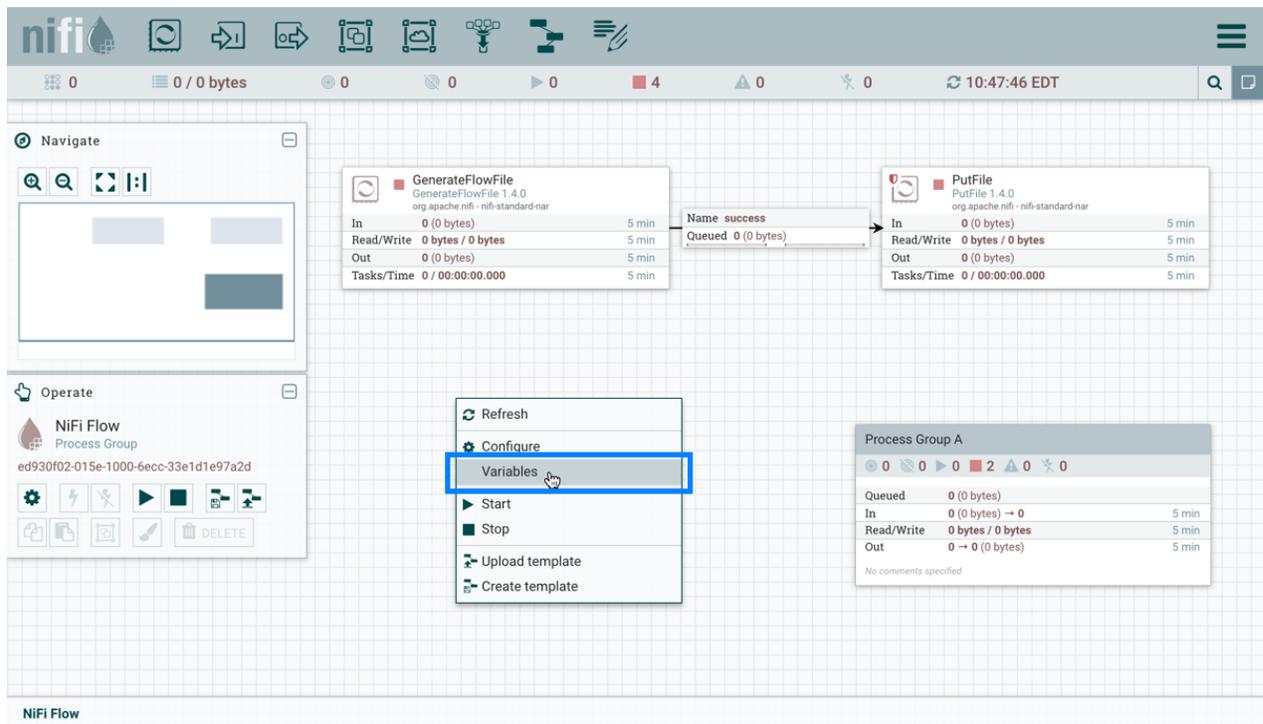


Variables

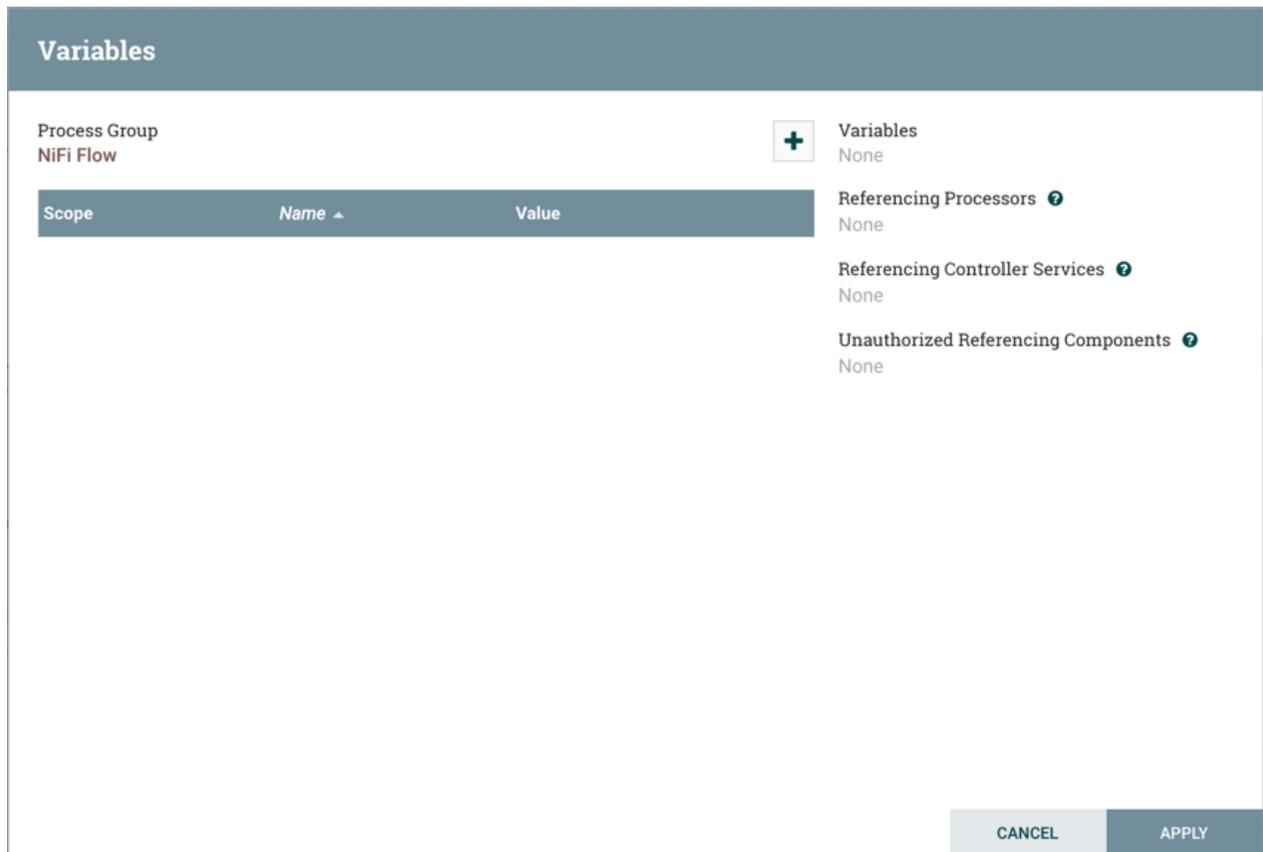
Variables are created and configured within the NiFi UI. They can be used in any field that supports Expression Language. Variables cannot be used for sensitive properties. Variables are defined at the Process Group level, as a result, the access policies for viewing and changing variables are derived from the access policies of the Process Group. Variable values cannot reference other variables or make use of Expression Language.

Variables Window

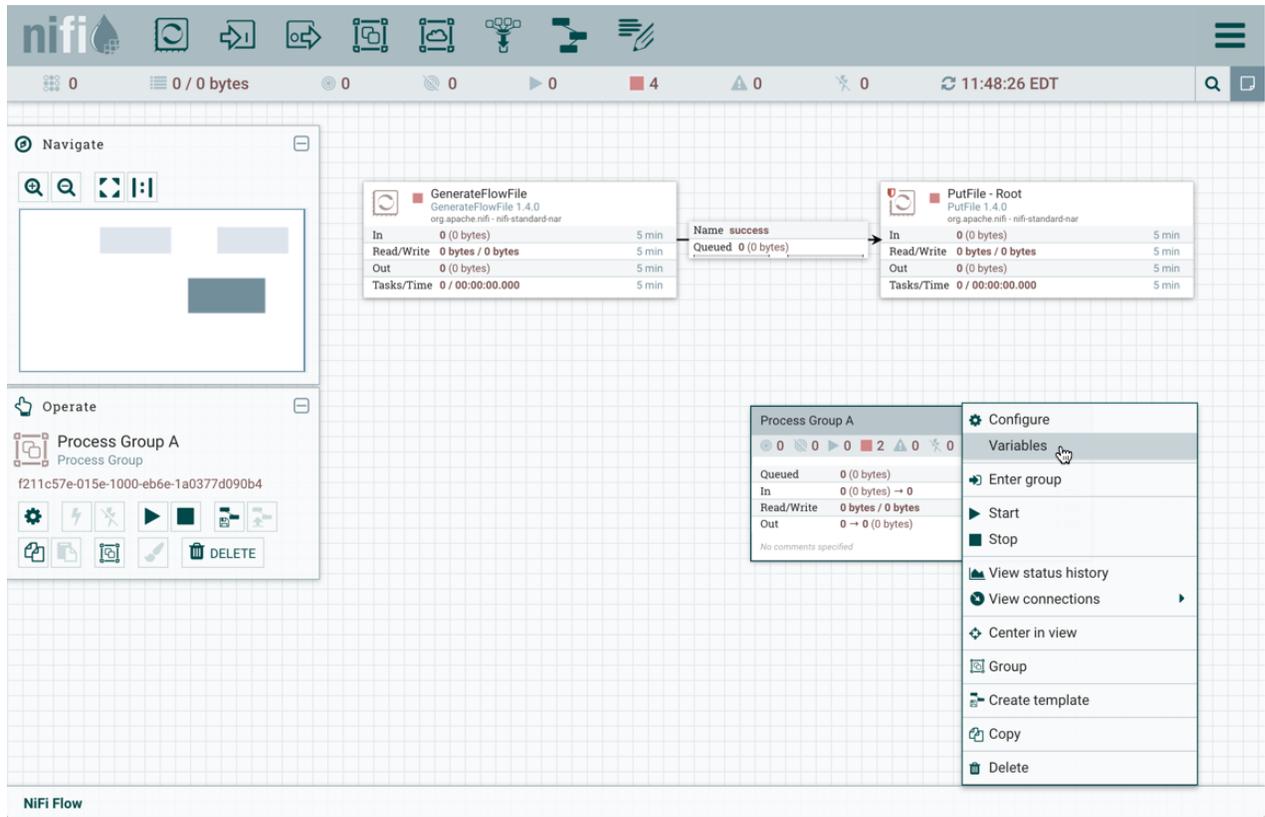
To access the Variables window, right-click on the canvas with nothing selected:



Select "Variables" from the Context Menu:

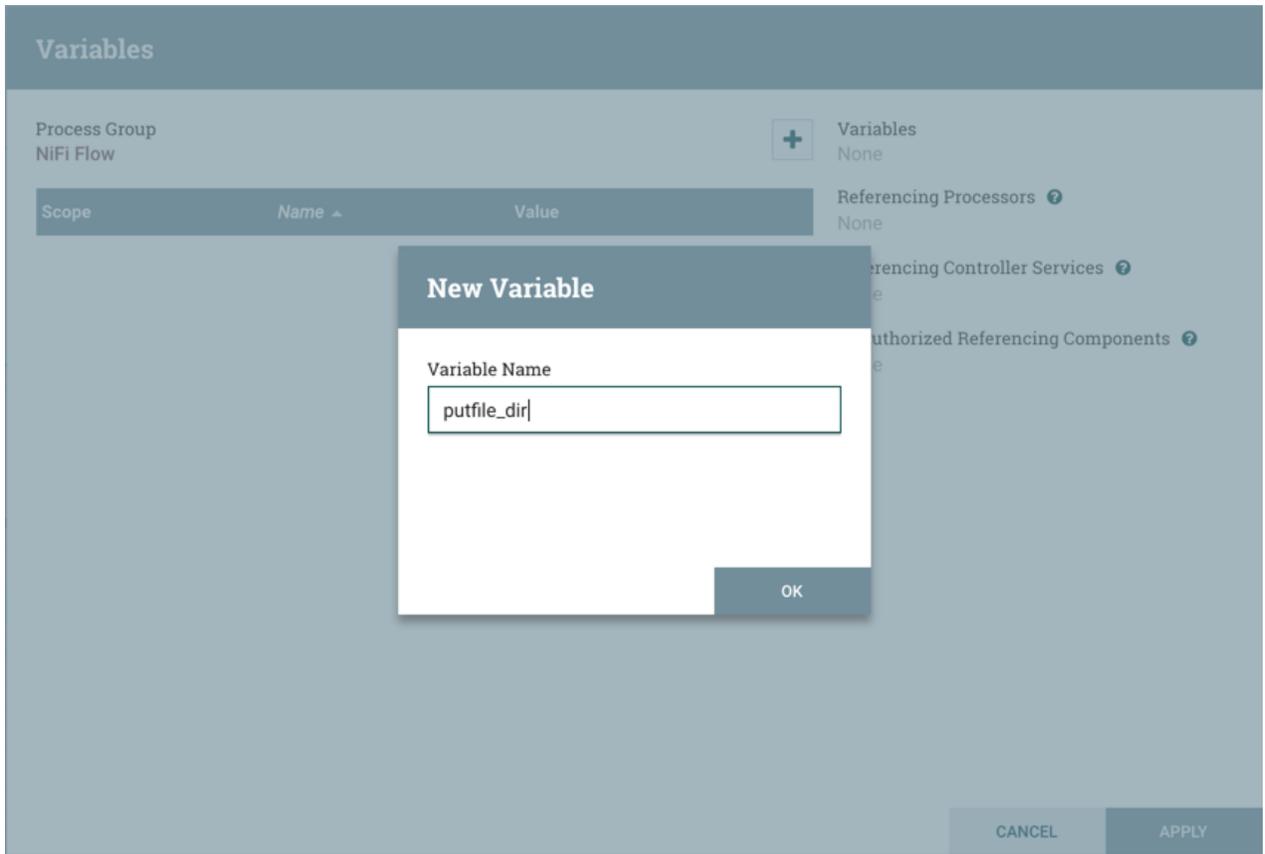


"Variables" is also available in the right-click Context Menu when a process group is selected:

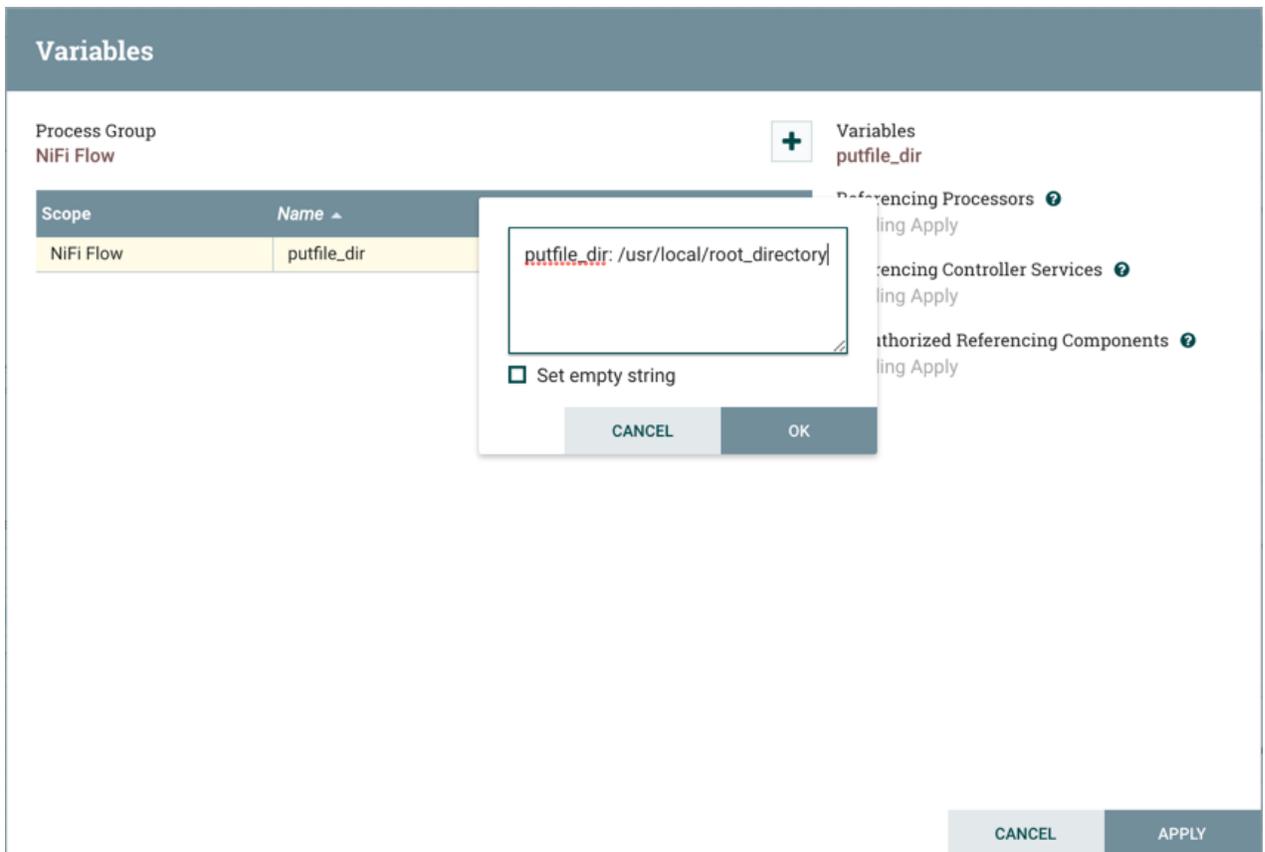


Creating a Variable

In the Variables window, click the + button to create a new variable. Add a name:



and a value:



Select "Apply":

Variables

| | |
|---|---|
| <p>Process Group NiFi Flow</p> <p>Steps To Update Variables</p> <ul style="list-style-type: none"> Identifying components affected ✓ Stopping affected Processors ✓ Disabling affected Controller Services ✓ Applying Updates ✓ Re-Enabling affected Controller Services ✓ Restarting affected Processors ✓ | <p>Variables putfile_dir</p> <p>Referencing Processors ?</p> <ul style="list-style-type: none"> ■ PutFile - Root <p>Referencing Controller Services ?</p> <p>None</p> <p>Unauthorized Referencing Components ?</p> <p>None</p> |
|---|---|

CLOSE

Steps to update the variable are performed (Identifying components affected, Stopping affected Processors, etc.). For example, the Referencing Processors section now lists the "PutFile-Root" processor. Selecting the name of the processor in the list will navigate to that processor on the canvas. Looking at the properties of the processor, `${putfile_dir}` is referenced by the Directory property:

Configure Processor

SETTINGS
SCHEDULING
PROPERTIES
COMMENTS

Required field +

| Property | Value |
|------------------------------|----------------------|
| Directory | ?(\${putfile_dir}) |
| Conflict Resolution Strategy | ?(fail) |
| Create Missing Directories | ?(true) |
| Maximum File Count | ?(No value set) |
| Last Modified Time | ?(No value set) |
| Permissions | ?(No value set) |
| Owner | ?(No value set) |
| Group | ?(No value set) |

CANCEL
APPLY

Variable Scope

Variables are scoped by the Process Group they are defined in and are available to any Processor defined at that level and below (i.e. any descendant Processors).

Variables in a descendant group override the value in a parent group. More specifically, if a variable x is declared at the root group and also declared inside a process group, components inside the process group will use the value of x defined in the process group.

For example, in addition to the putfile_dir variable that exists at the root process group, assume another putfile_dir variable was created within Process Group A. If one of the components within Process Group A references putfile_dir, both variables will be listed, but the putfile_dir from the root group will have a strikethrough indicating that it is being overridden:

Variables

Process Group
Process Group A

| Scope | Name ▲ | Value | |
|-----------------|-------------|--------------------------------------|----|
| Process Group A | putfile_dir | /usr/local/pgA_directory | 🗑️ |
| NiFi Flow | putfile_dir | /usr/local/root_directory | ➔ |

+ Variables
putfile_dir

Referencing Processors ?
■ PutFile - PG-A

Referencing Controller Services ?
None

Unauthorized Referencing Components ?
None

CANCEL
APPLY

A variable can only be modified for the process group it was created in, which is listed at the top of the Variables window. To modify a variable defined in a different process group, select the "arrow" icon in that variable's row:

Variables

Process Group
Process Group A ← **Current Process Group**

| Scope | Name | Value | |
|-----------------|-------------|---------------------------|----|
| Process Group A | putfile_dir | /usr/local/pgA_directory | 🗑️ |
| NiFi Flow | putfile_dir | /usr/local/root_directory | ➔ |

Variables
putfile_dir

Referencing Processors ?
PutFile - Root

Referencing Controller Services ?
None

Unauthorized Referencing Components ?
None

Root Process Group

Go to Variable Window for Root Process Group

CANCEL APPLY

which will navigate to the Variables window for that process group:

Variables

Process Group
NiFi Flow

| Scope | Name | Value | |
|-----------|-------------|--------------------------------|----|
| NiFi Flow | drivers_dir | file:///drivers/jdbc/mysql-... | 🗑️ |
| NiFi Flow | putfile_dir | /usr/local/root_directory | 🗑️ |

Variables
putfile_dir

Referencing Processors ?
PutFile - Root

Referencing Controller Services ?
None

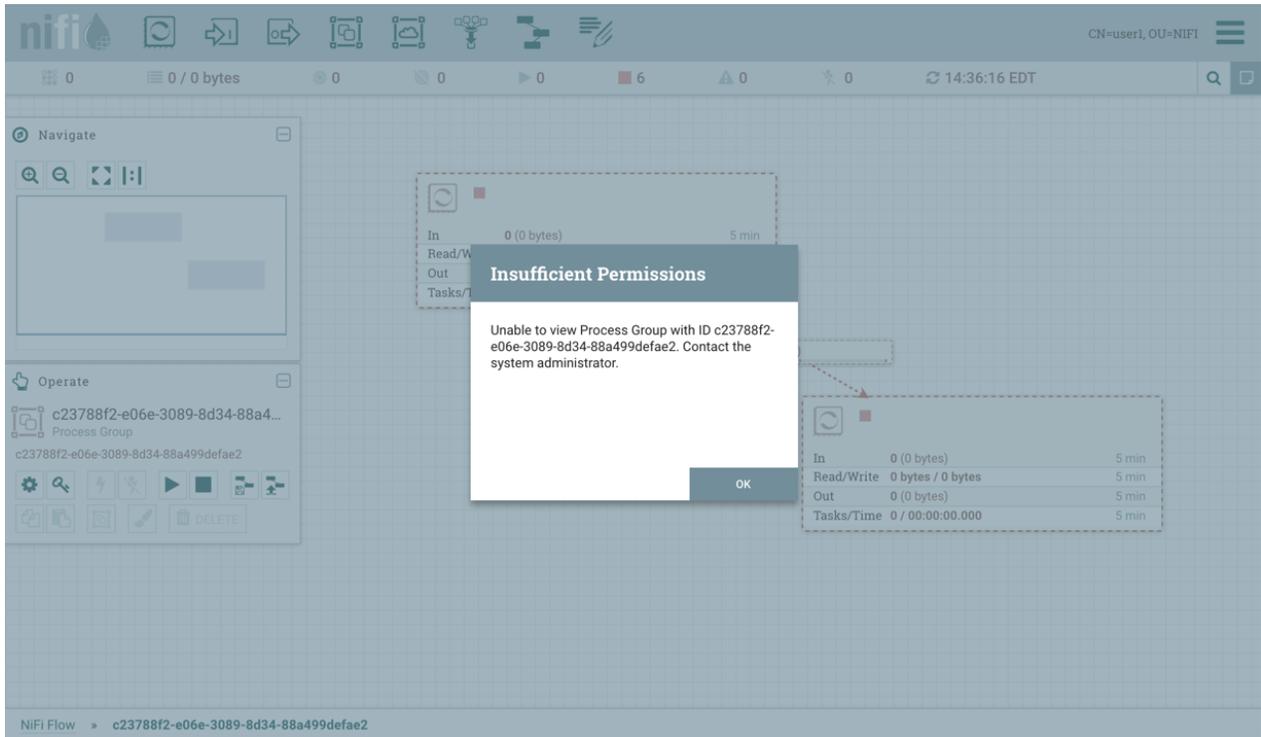
Unauthorized Referencing Components ?
None

CANCEL APPLY

Variable Permissions

Variable permissions are based solely on the privileges configured on the corresponding Process Group.

For example, if a user does not have access to View a process group, the Variables window can not be viewed for that process group:



If a user has access to View a process group but does not have access to Modify the process group, the variables can be viewed but not modified.

For information on how to manage privileges on components, see the *System Administration documentation* in our *Reference* section.

Referencing Controller Services

In addition to Referencing Processors, the Variables window also displays Referencing Controller Services:

Variables

Process Group: NiFi Flow

Variables: drivers_dir

| Scope | Name | Value |
|-----------|-------------|--|
| NiFi Flow | drivers_dir | file:///drivers/jdbc/mysql-connector-java-5.1... |
| NiFi Flow | property1 | ./private |
| NiFi Flow | putfile_dir | /usr/local/root_directory |

Referencing Processors: None

Referencing Controller Services: DBCPConnectionPool

Unauthorized Referencing Components: None

CANCEL APPLY

Selecting the name of the controller service will navigate to that controller service in the Configuration window:

NiFi Flow Configuration

GENERAL CONTROLLER SERVICES

| Name | Type | Bundle | State | Scope |
|--------------------|--------------------------|--|---------|-----------|
| DBCPConnectionPool | DBCPConnectionPool 1.4.0 | org.apache.nifi - nifi-dbc-service-nar | Enabled | NiFi Flow |

Last updated: 12:08:48 EDT

Listed services are available to all descendant Processors and services of this Process Group.

Unauthorized Referencing Components

When View or Modify privileges are not given to a component that references a variable, the UUID of the component will be displayed in the Variables window:

Variables

Process Group
NiFi Flow

| Scope | Name | Value |
|-----------|-------------|--|
| NiFi Flow | drivers_dir | file:///drivers/jdbc/mysql-connector-java-5.1... |
| NiFi Flow | property1 | ./private |
| NiFi Flow | putfile_dir | /usr/local/root_directory |

Variables
property1

Referencing Processors ?
None

Referencing Controller Services ?
None

Unauthorized Referencing Components ?
424d4e96-ae1c-310a-48cc-c7f45b8b74e6

CANCEL APPLY

In the above example, the variable property1 is referenced by a processor that "user1" is not able to view:

The screenshot shows the NiFi web interface with a data flow diagram. The top navigation bar includes the NiFi logo, a user menu (CN=user1, OU=NIFI), and various status indicators. The main workspace displays a data flow with the following components:

- GenerateFlowFile** (top left): GenerateFlowFile 1.4.0, org.apache.nifi-nifi-standard-nar. In: 0 (0 bytes), Read/Write: 0 bytes / 0 bytes, Out: 0 (0 bytes), Tasks/Time: 0 / 00:00:00.000.
- PutFile - Root** (top right): PutFile 1.4.0, org.apache.nifi-nifi-standard-nar. In: 0 (0 bytes), Read/Write: 0 bytes / 0 bytes, Out: 0 (0 bytes), Tasks/Time: 0 / 00:00:00.000.
- Process Group A** (middle right): Summary statistics showing 2 tasks and 0 bytes.
- GenerateFlowFile** (bottom left): GenerateFlowFile 1.4.0, org.apache.nifi-nifi-standard-nar. In: 0 (0 bytes), Read/Write: 0 bytes / 0 bytes, Out: 0 (0 bytes), Tasks/Time: 0 / 00:00:00.000.
- Unauthorized Component** (bottom right): A processor box with a dashed border and a red 'X' icon, indicating it is unauthorized for the current user. It has the same statistics as the other GenerateFlowFile processors.

The unauthorized component's ID, 424d4e96-ae1c-310a-48cc-c7f45b8b74e6, is highlighted in a blue box in the 'Operate' sidebar on the left.

Referencing Custom Properties via `nifi.properties`

Identify one or more sets of key/value pairs, and give them to your system administrator.

Once the new custom properties have been added, ensure that the `nifi.variable.registry.properties` field in the `nifi.properties` file is updated with the custom properties location.



Note: NiFi must be restarted for these updates to be picked up.

For more information, see the *System Administration documentation* in the *Reference* section.

Controller Services

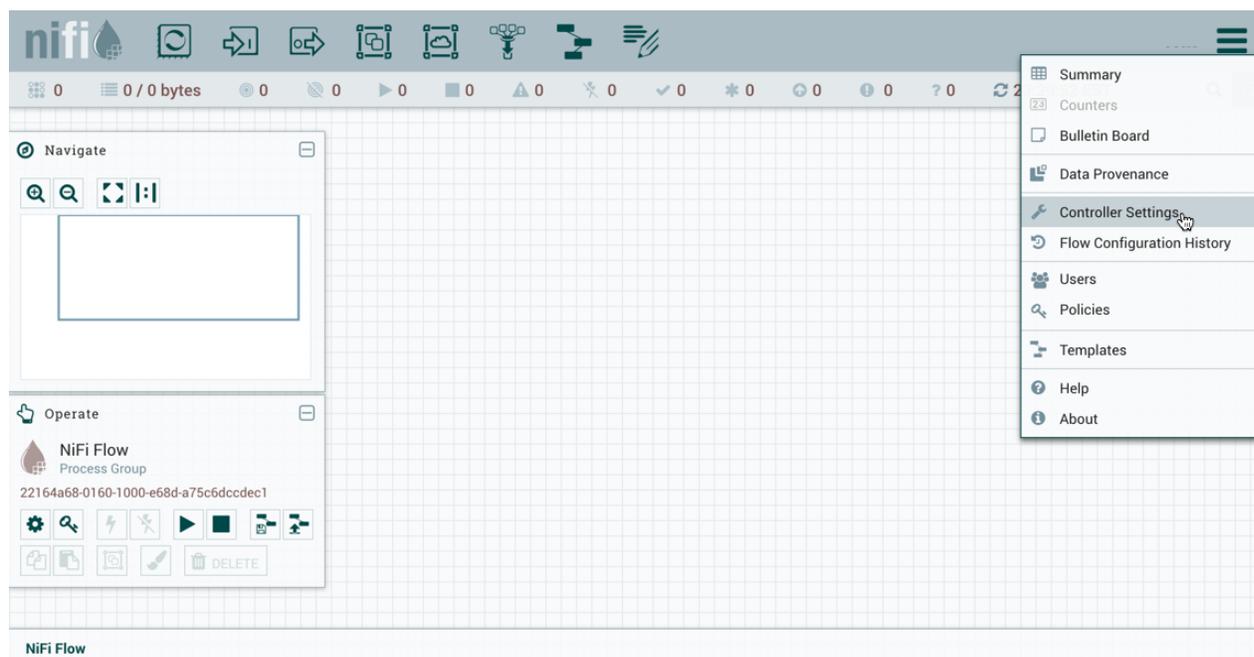
Controller Services are shared services that can be used by reporting tasks, processors, and other services to utilize for configuration or task execution.



Note: If your NiFi instance is secured, your ability to view and add Controller Services is dependent on the privileges assigned to you. If you do not have access to one or more Controller Services, you are not able to see or access it in the UI. Access privileges can be assigned on a global or Controller Service-specific basis (see *Accessing the UI with Multi-Tenant Authorization* for more information).

Adding Controller Services for Reporting Tasks

To add a Controller Service for a reporting task, select Controller Settings from the Global Menu.



This displays the NiFi Settings window. The window has four tabs: General, Reporting Task Controller Services, Reporting Tasks and Registry Clients. The General tab provides settings for the overall maximum thread counts of the instance.

NiFi Settings

GENERAL

REPORTING TASK CONTROLLER SERVICES

REPORTING TASKS

REGISTRY CLIENTS

Maximum Timer Driven Thread Count ?

10

Maximum Event Driven Thread Count ?

5

APPLY

To the right of the General tab is the Reporting Task Controller Services tab. From this tab, the DFM may click the + button in the upper-right corner to create a new Controller Service.

NiFi Settings ← NiFi context of this configuration dialog

GENERAL
REPORTING TASK CONTROLLER SERVICES
REPORTING TASKS
REGISTRY CLIENTS

Click to add a Reporting Task Controller Service → +

| Name | Type | Bundle | State | Scope |
|--------------------|------------------------------|--|----------|------------|
| DBCPConnectionPool | DBCPCConnectionPool 1.5.0... | org.apache.nifi - nifi-dbcpc-service-nar | Disabled | Controller |

↑ Scope of Controller Service

Description of availability
↓

Last updated: 13:21:34 EST Listed services are available to all Reporting Tasks and services defined in the Controller Settings.

The Add Controller Service window opens. This window is similar to the Add Processor window. It provides a list of the available Controller Services on the right and a tag cloud, showing the most common category tags used for Controller Services, on the left. The DFM may click any tag in the tag cloud in order to narrow down the list of Controller Services to those that fit the categories desired. The DFM may also use the Filter field at the top-right of the window to search for the desired Controller Service or use the Source drop-down at the top-left to filter the list by the group who created them. Upon selecting a Controller Service from the list, the DFM can see a description of the service below. Select the desired controller service and click Add, or simply double-click the name of the service to add it.

Add Controller Service

Source: all groups | Displaying 10 of 28 | Serv

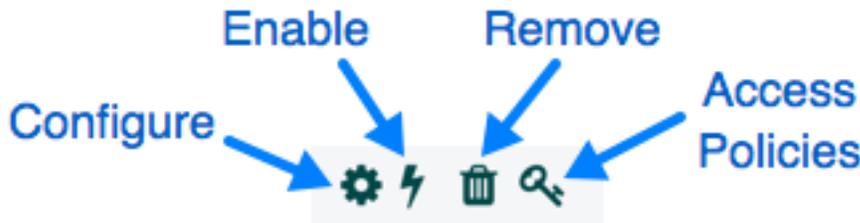
| Type | Version | Tags |
|---|---------|--|
| AWSCredentialsProviderControllerService | 1.2.0 | credentials, provider, aws |
| CouchbaseClusterService | 1.2.0 | database, couchbase, connection, ... |
| DistributedMapCacheClientService | 1.2.0 | cluster, cache, distributed, state, m... |
| DistributedMapCacheServer | 1.2.0 | cluster, server, cache, key/value, dis... |
| DistributedSetCacheClientService | 1.2.0 | cluster, cache, set, distributed, state |
| DistributedSetCacheServer | 1.2.0 | server, cache, set, distributed, disti... |
| GCPCredentialsControllerService | 1.2.0 | gcp, credentials, provider |
| HBase_1_1_2_ClientService | 1.2.0 | client, hbase |
| StandardSSLContextService | 1.2.0 | p12, jks, certificate, keystore, trusts... |
| JettyWebSocketServer | 1.2.0 | server, Jetty, WebSocket |

StandardSSLContextService 1.2.0 org.apache.nifi - nifi-ssl-context-service-nar

Standard implementation of the SSLContextService. Provides the ability to configure keystore and/or truststore properties once and reuse that configuration throughout the application

CANCEL ADD

Once you have added a Controller Service, you can configure it by clicking the "Configure" button in the far-right column. Other buttons in this column include "Enable", "Remove" and "Access Policies".



You can obtain information about Controller Services by clicking the "Usage" and "Alerts" buttons in the left-hand column.



When the DFM clicks the "Configure" button, a Configure Controller Service window opens. It has three tabs: Settings, Properties, and Comments. This window is similar to the Configure Processor window. The Settings tab provides a place for the DFM to give the Controller Service a unique name (if desired). It also lists the UUID, Type, Bundle and Support information for the service and provides a list of other components (reporting tasks or other controller services) that reference the service.

Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Name
StandardSSLContextService

Id
172438f9-015c-1000-ecd6-ea699f013e59

Type
StandardSSLContextService 1.2.0

Bundle
org.apache.nifi - nifi-ssl-context-service-nar

Supports Controller Service

- SSLContextService 1.2.0 from org.apache.nifi - nifi-standard-services-api-nar

Referencing Components ⓘ

▼ **Reporting Tasks (1)**

- ⚠ SiteToSiteProvenanceReportingTask

CANCEL

APPLY

The Properties tab lists the various properties that apply to the particular controller service. As with configuring processors, the DFM may hover over the question mark icons to see more information about each property.

Configure Controller Service

SETTINGS
PROPERTIES
COMMENTS

Required field +

| Property | Value | |
|---------------------|-------|---------------------|
| Keystore Filename | ? | No value set |
| Keystore Password | ? | Sensitive value set |
| Key Password | ? | Sensitive value set |
| Keystore Type | ? | No value set |
| Truststore Filename | ? | No value set |
| Truststore Password | ? | Sensitive value set |
| Truststore Type | ? | No value set |
| SSL Protocol | ? | TLS |

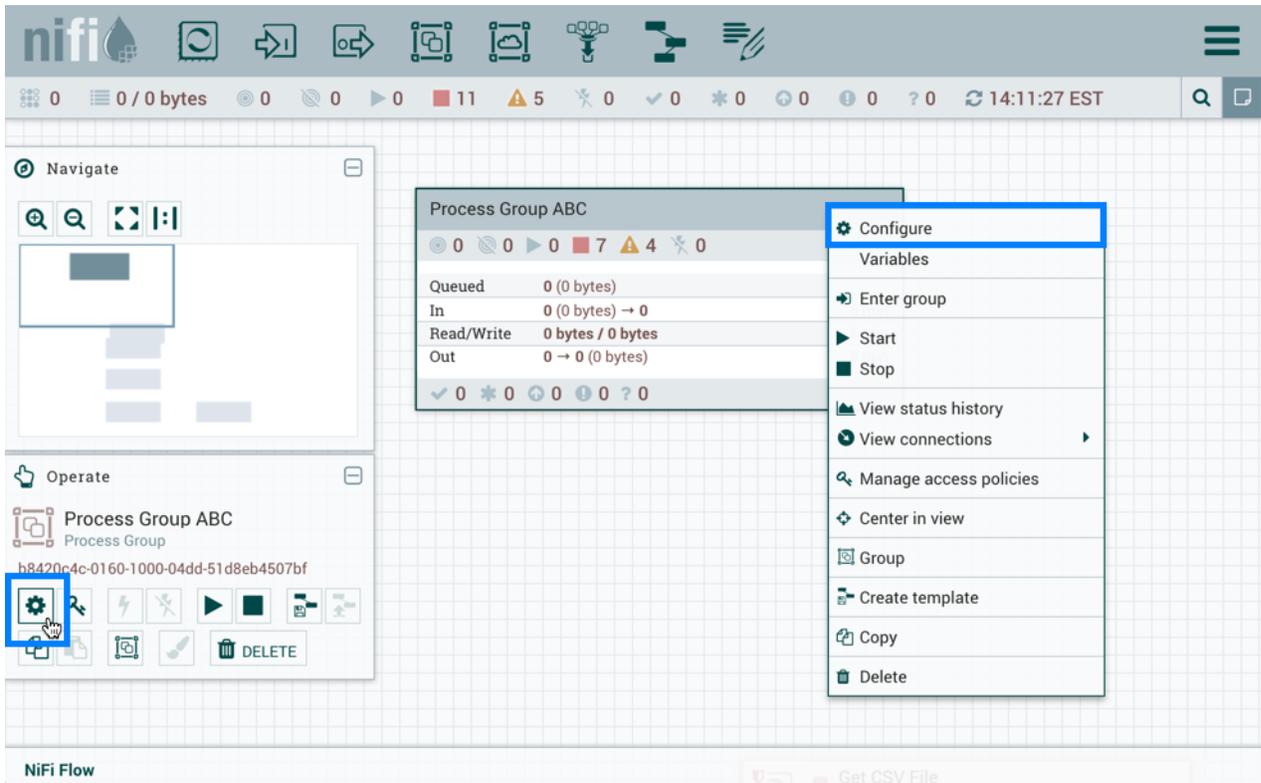
The algorithm to use for this SSL context
 Default value: TLS
 Supports expression language: false

CANCEL
APPLY

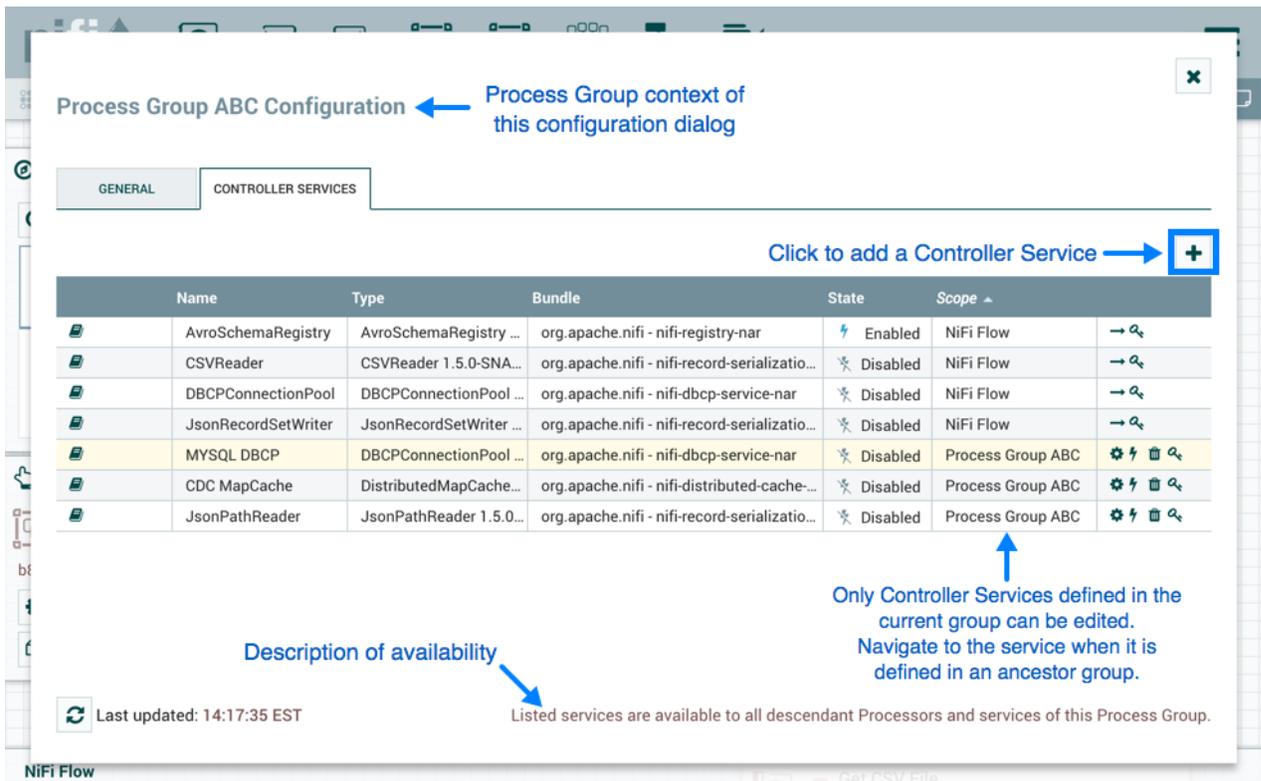
The Comments tab is just an open-text field, where the DFM may include comments about the service. After configuring a Controller Service, click "Apply" to save the configuration and close the window, or click "Cancel" to discard the changes and close the window.

Adding Controller Services for Dataflows

To add a Controller Service for a dataflow, you can either right click a Process Group and select Configure, or click Configure from the Operate Palette.

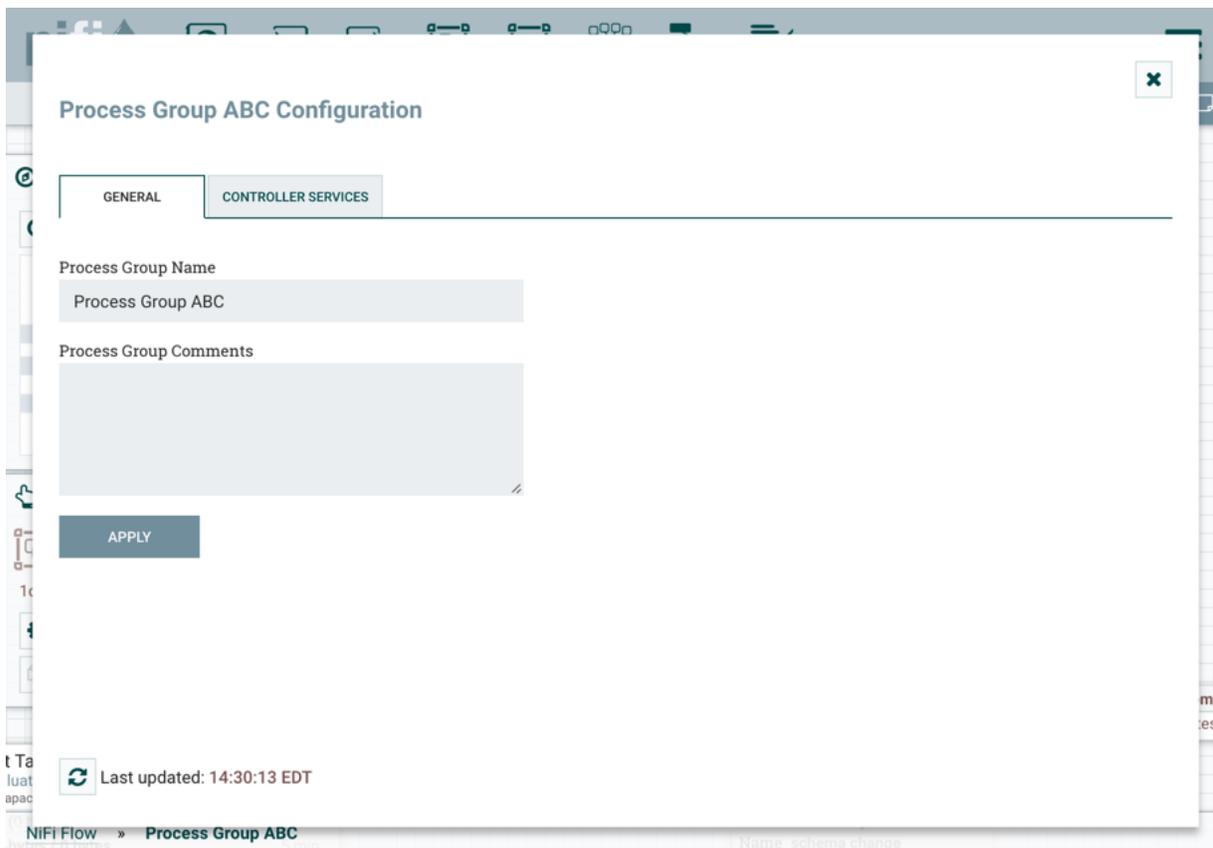


When you click Configure from the Operate Palette with nothing selected on your canvas, you add a Controller Service for your Root Process Group. That Controller Service is then available to all nested Process Groups in your dataflow. When you select a Process Group on the canvas and then click Configure from either the Operate Palette or the Process Group context menu, the service will be available to all Processors and Controller Services defined in that Process Group and below.



Use the following steps to add a Controller Service:

1. Click Configure, either from the Operate Palette, or from the Process Group context menu. This displays the process group Configuration window. The window has two tabs: General and Controller Services. The General tab is for settings that pertain to general information about the process group. For example, if configuring the root process group, the DFM can provide a unique name for the overall dataflow, as well as comments that describe the flow (Note: this information is visible to any other NiFi instance that connects remotely to this instance (using Remote Process Groups, a.k.a., Site-to-Site)).



2. From the Process Group Configuration page, select the Controller Services tab.
3. Click the + button to display the Add Controller Service dialog.
4. Select the Controller Service desired, and click Add.
5. Perform any necessary Controller Service configuration tasks by clicking the Configure icon

()
 in the right-hand column.

Enabling/Disabling Controller Services

After a Controller Service has been configured, it must be enabled in order to run. Do this using the "Enable" button

()
 in the far-right column of the Controller Services tab. In order to modify an existing/running controller service, the DFM needs to stop/disable it (as well as all referencing reporting tasks and controller services). Do this using the "Disable" button

()
 Rather than having to hunt down each component that is referenced by that controller service, the DFM has the ability

to stop/disable them when disabling the controller service in question. When enabling a controller service, the DFM has the option to either start/enable the controller service and all referencing components or start/enable only the controller service itself.

Enable Controller Service

Service
StandardSSLContextService

Scope

- Service only ✓
- Service only ?
- Service and referencing components ?

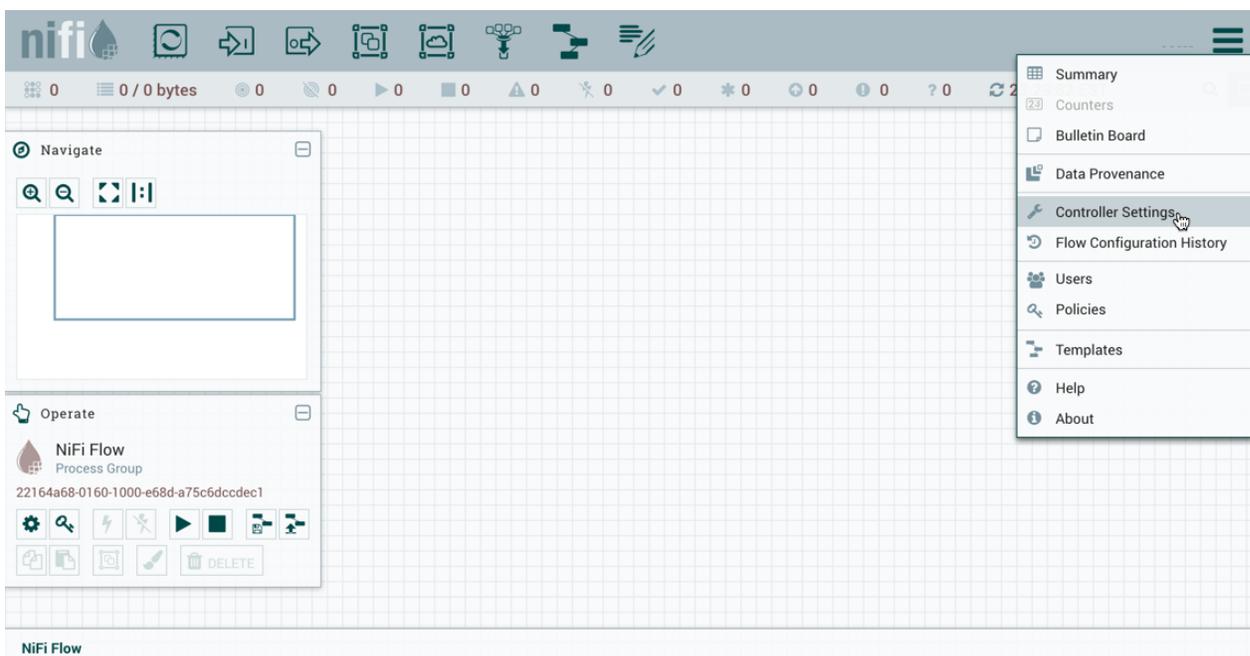
Referencing Components ?

- ▼ Reporting Tasks (1)
 - ⚠ SiteToSiteProvenanceReportingTask

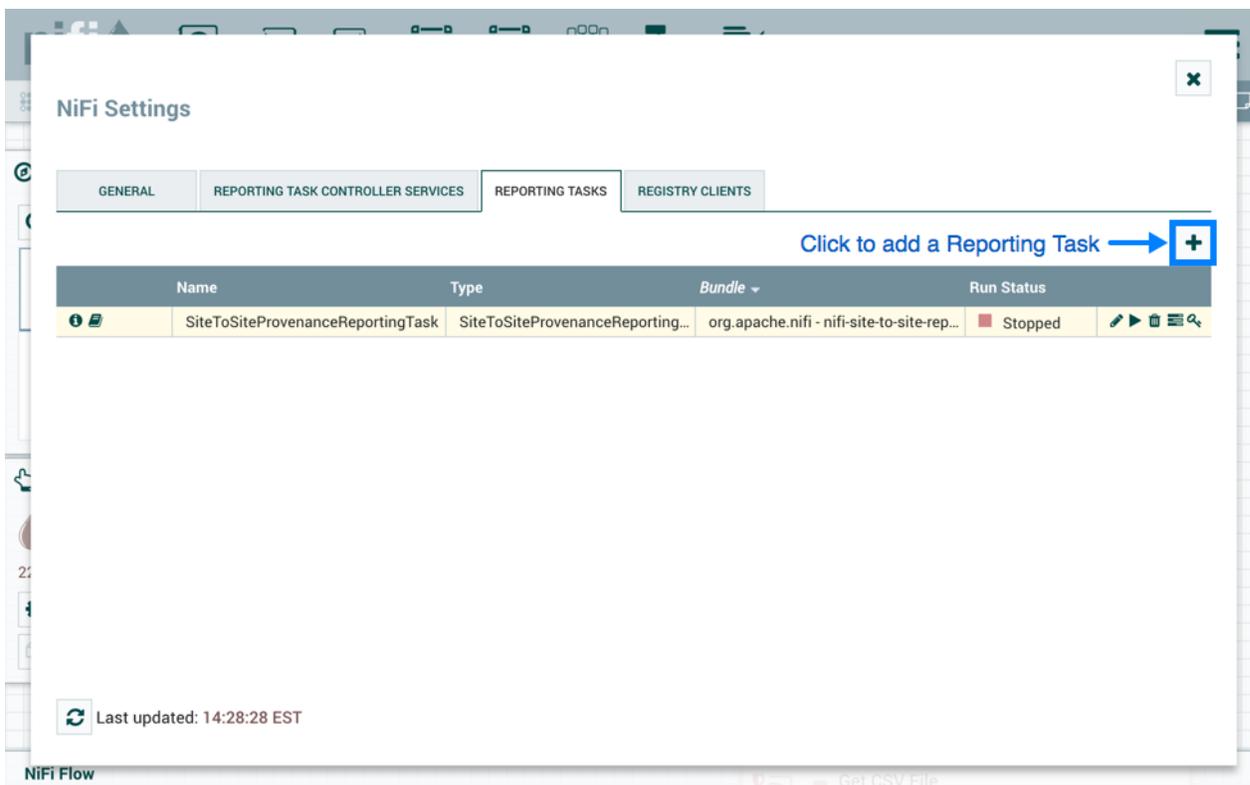
CANCEL ENABLE

Reporting Tasks

Reporting Tasks run in the background to provide statistical reports about what is happening in the NiFi instance. The DFM adds and configures Reporting Tasks similar to the process for Controller Services. To add a Reporting Task, select Controller Settings from the Global Menu.



This displays the NiFi Settings window. Select the Reporting Tasks tab and click the + button in the upper-right corner to create a new Reporting Task.



The Add Reporting Task window opens. This window is similar to the Add Processor window. It provides a list of the available Reporting Tasks on the right and a tag cloud, showing the most common category tags used for Reporting Tasks, on the left. The DFM may click any tag in the tag cloud in order to narrow down the list of Reporting Tasks to those that fit the categories desired. The DFM may also use the Filter field at the top-right of the window to search for the desired Reporting Task or use the Source drop-down at the top-left to filter the list by the group who created them. Upon selecting a Reporting Task from the list, the DFM can see a description of the task below. Select the desired reporting task and click Add, or simply double-click the name of the service to add it.

Add Reporting Task

Source: all groups

Displaying 7 of 10

| Type | Version | Tags |
|-----------------------------------|---------|---|
| AmbariReportingTask | 1.2.0 | ambari, metrics, reporting |
| ControllerStatusReportingTask | 1.2.0 | stats, log |
| DataDogReportingTask | 1.2.0 | datadog, metrics, reporting |
| ScriptedReportingTask | 1.2.0 | lua, python, groovy, jython, js, lua, execute, rep... |
| SiteToSiteBulletinReportingTask | 1.2.0 | site, restricted, bulletin, site to site |
| SiteToSiteProvenanceReportingTask | 1.2.0 | lineage, site, provenance, restricted, tracking, ... |
| SiteToSiteStatusReportingTask | 1.2.0 | site, metrics, history, status, site to site |

SiteToSiteStatusReportingTask 1.2.0 org.apache.nifi - nifi-site-to-site-reporting-nar

Publishes Status events using the Site To Site protocol. The component type and name filter regexes form a union: only components matching both regexes will be reported. However, all process groups are recursively searched for matching components, regardless of whether the process group matches the component filters.

CANCEL ADD

Once a Reporting Task has been added, the DFM may configure it by clicking the "Edit" button in the far-right column. Other buttons in this column include "Start", "Remove", "State" and "Access Policies".



You can obtain information about Reporting Tasks by clicking the "View Details", "Usage", and "Alerts" buttons in the left-hand column.



When the DFM clicks the "Edit" button, a Configure Reporting Task window opens. It has three tabs: Settings, Properties, and Comments. This window is similar to the Configure Processor window. The Settings tab provides a place for the DFM to give the Reporting Task a unique name (if desired). It also lists the UUID, Type, and Bundle information for the task and provides settings for the task's Scheduling Strategy and Run Schedule (similar to the same settings in a processor). The DFM may hover the mouse over the question mark icons to see more information about each setting.

Configure Reporting Task

SETTINGS

PROPERTIES

COMMENTS

Name
SiteToSiteStatusReportingTask Enabled

Id
1d1f5ecd-015c-1000-bff7-40b54c50db99

Type
SiteToSiteStatusReportingTask 1.2.0

Bundle
org.apache.nifi - nifi-site-to-site-reporting-nar

Scheduling Strategy ?
Timer driven The strategy used to schedule this reporting

Run Schedule ?
5 mins

CANCEL
APPLY

The Properties tab lists the various properties that may be configured for the task. The DFM may hover the mouse over the question mark icons to see more information about each property.

Configure Reporting Task

SETTINGS
PROPERTIES
COMMENTS

+

Required field

| Property | Value |
|-----------------------------|---|
| Destination URL | No value set |
| Input Port Name | No value set |
| SSL Context Service | No value set |
| Instance URL | http://\${hostname(true)}:8080/nifi |
| Compress Events | true |
| Communications Timeout | 30 secs |
| Batch Size | 1000 |
| Platform | nifi |
| Component Type Filter Regex | (Processor ProcessGroup RemoteProcessGrou.. |
| Component Name Filter Regex | .* |

A regex specifying which component names to report. Any component name matching this regex will be included.
 Default value: .*
 Supports expression language: true

CANCEL
APPLY

The Comments tab is just an open-text field, where the DFM may include comments about the task. After configuring the Reporting Task, click "Apply" to save the configuration and close the window, or click "Cancel" to discard the changes and close the window.

When you want to run the Reporting Task, click the "Start" button



Connecting Components

Once processors and other components have been added to the canvas and configured, the next step is to connect them to one another so that NiFi knows what to do with each FlowFile after it has been processed. This is accomplished by creating a Connection between each component. When the user hovers the mouse over the center of a component, a new Connection icon (



) appears:

| | | |
|---|--|---|
|  | GenerateFlowFile GenerateFlowFile 1.2.0 org.apache.nifi - nifi-standard-nar |  |
| In | 0 (0 bytes) | 5 min |
| Read/Write | 0 bytes / 0 bytes | 5 min |
| Out | 0 (0 bytes) | 5 min |
| Tasks/Time | 0 / 00:00:00.000 | 5 min |

The user drags the Connection bubble from one component to another until the second component is highlighted. When the user releases the mouse, a 'Create Connection' dialog appears. This dialog consists of two tabs: 'Details' and 'Settings'. They are discussed in detail below. Note that it is possible to draw a connection so that it loops back on the same processor. This can be useful if the DFM wants the processor to try to re-process FlowFiles if they go down a failure Relationship. To create this type of looping connection, simply drag the connection bubble away and then back to the same processor until it is highlighted. Then release the mouse and the same 'Create Connection' dialog appears.

Details Tab

The Details tab of the 'Create Connection' dialog provides information about the source and destination components, including the component name, the component type, and the Process Group in which the component lives:

Create Connection

DETAILS

SETTINGS

| | |
|---|---|
| <p>From Processor GenerateFlowFile GenerateFlowFile</p> <p>Within Group NiFi Flow</p> <p>For Relationships <input checked="" type="checkbox"/> success</p> | <p>To Processor LogAttribute LogAttribute</p> <p>Within Group NiFi Flow</p> |
|---|---|

CANCEL

ADD

Additionally, this tab provides the ability to choose which Relationships should be included in this Connection. At least one Relationship must be selected. If only one Relationship is available, it is automatically selected.



Note: If multiple Connections are added with the same Relationship, any FlowFile that is routed to that Relationship will automatically be 'cloned', and a copy will be sent to each of those Connections.

Settings

The Settings tab provides the ability to configure the Connection's Name, FlowFile Expiration, Back Pressure Thresholds, Load Balance Strategy and Prioritization:

Configure Connection

DETAILS | **SETTINGS**

Name

Id
208bf461-0166-1000-ffff-fffffb88d70

FlowFile Expiration ⓘ
0 sec

Back Pressure Object Threshold ⓘ | Size Threshold ⓘ
10000 | 1 GB

Load Balance Strategy ⓘ
Do not load balance

Available Prioritizers ⓘ

- FirstInFirstOutPrioritizer
- NewestFlowFileFirstPrioritizer
- OldestFlowFileFirstPrioritizer
- PriorityAttributePrioritizer

Selected Prioritizers ⓘ

CANCEL | APPLY

The Connection name is optional. If not specified, the name shown for the Connection will be names of the Relationships that are active for the Connection.

FlowFile Expiration

FlowFile expiration is a concept by which data that cannot be processed in a timely fashion can be automatically removed from the flow. This is useful, for example, when the volume of data is expected to exceed the volume that can be sent to a remote site. In this case, the expiration can be used in conjunction with Prioritizers to ensure that the highest priority data is processed first and then anything that cannot be processed within a certain time period (one hour, for example) can be dropped. The expiration period is based on the time that the data entered the NiFi instance. In other words, if the file expiration on a given connection is set to '1 hour', and a file that has been in the NiFi instance for one hour reaches that connection, it will expire. The default value of 0 sec indicates that the data will never expire. When a file expiration other than '0 sec' is set, a small clock icon appears on the connection label, so the DFM can see it at-a-glance when looking at a flow on the canvas.



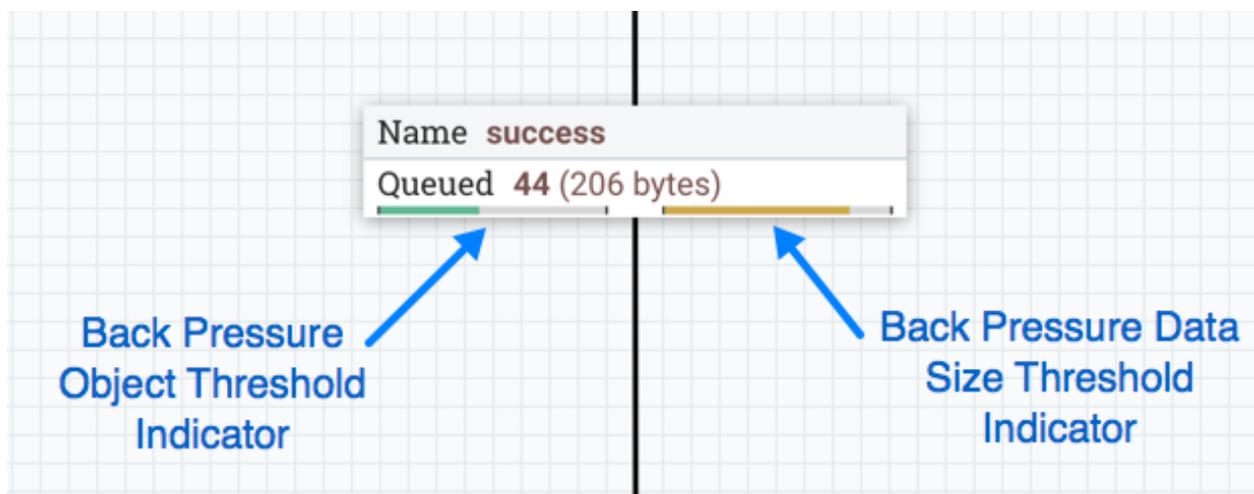
Back Pressure

NiFi provides two configuration elements for Back Pressure. These thresholds indicate how much data should be allowed to exist in the queue before the component that is the source of the Connection is no longer scheduled to run. This allows the system to avoid being overrun with data. The first option provided is the "Back pressure object threshold." This is the number of FlowFiles that can be in the queue before back pressure is applied. The second configuration option is the "Back pressure data size threshold." This specifies the maximum amount of data (in size) that should be queued up before applying back pressure. This value is configured by entering a number followed by a data size (B for bytes, KB for kilobytes, MB for megabytes, GB for gigabytes, or TB for terabytes).

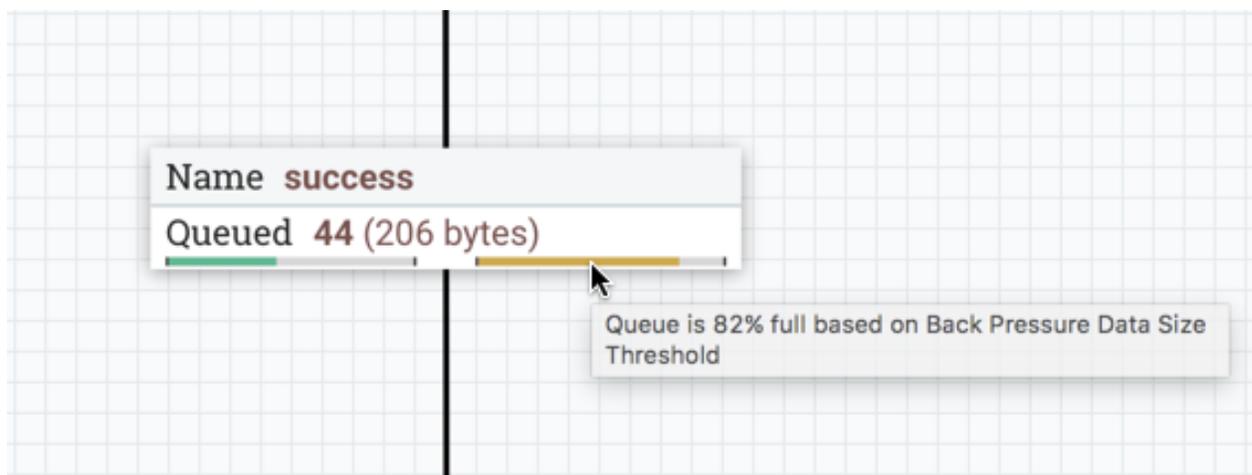


Note: By default each new connection added will have a default Back Pressure Object Threshold of 10,000 objects and Back Pressure Data Size Threshold of 1 GB. These defaults can be changed by modifying the appropriate properties in the nifi.properties file.

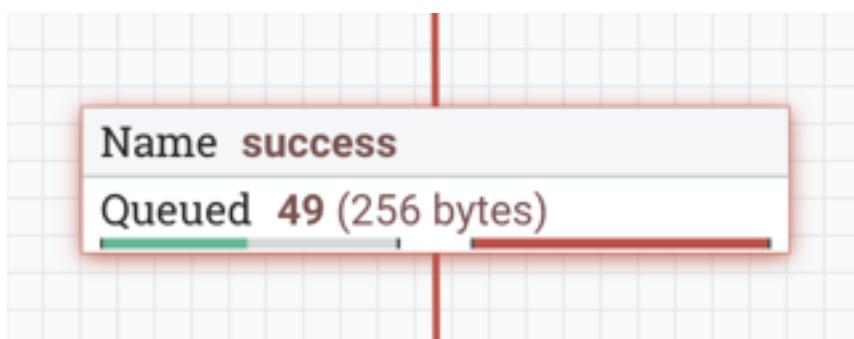
When back pressure is enabled, small progress bars appear on the connection label, so the DFM can see it at-a-glance when looking at a flow on the canvas. The progress bars change color based on the queue percentage: Green (0-60%), Yellow (61-85%) and Red (86-100%).



Hovering your mouse over a bar displays the exact percentage.



When the queue is completely full, the Connection is highlighted in red.



Load Balancing

Load Balancing Strategy

To distribute the data in a flow across the nodes in the cluster, NiFi offers the following load balance strategies:

- Do not load balance: Do not load balance FlowFiles between nodes in the cluster. This is the default.
- Partition by attribute: Determines which node to send a given FlowFile to based on the value of a user-specified FlowFile Attribute. All FlowFiles that have the same value for the Attribute will be sent to the same node in the cluster. If the destination node is disconnected from the cluster or if unable to communicate, the data does not fail over to another node. The data will queue, waiting for the node to be available again. Additionally, if a node joins or leaves the cluster necessitating a rebalance of the data, consistent hashing is applied to avoid having to redistribute all of the data.
- Round robin: FlowFiles will be distributed to nodes in the cluster in a round-robin fashion. If a node is disconnected from the cluster or if unable to communicate with a node, the data that is queued for that node will be automatically redistributed to another node(s). If a node is not able to receive the data as fast other nodes in the cluster, the node may also be skipped for one or more iterations in order to maximize throughput of data distribution across the cluster.
- Single node: All FlowFiles will be sent to a single node in the cluster. Which node they are sent to is not configurable. If the node is disconnected from the cluster or if unable to communicate with the node, the data that is queued for that node will remain queued until the node is available again.



Note: In addition to the UI settings, there are Cluster Node Properties related to load balancing that must also be configured in `nifi.properties`.

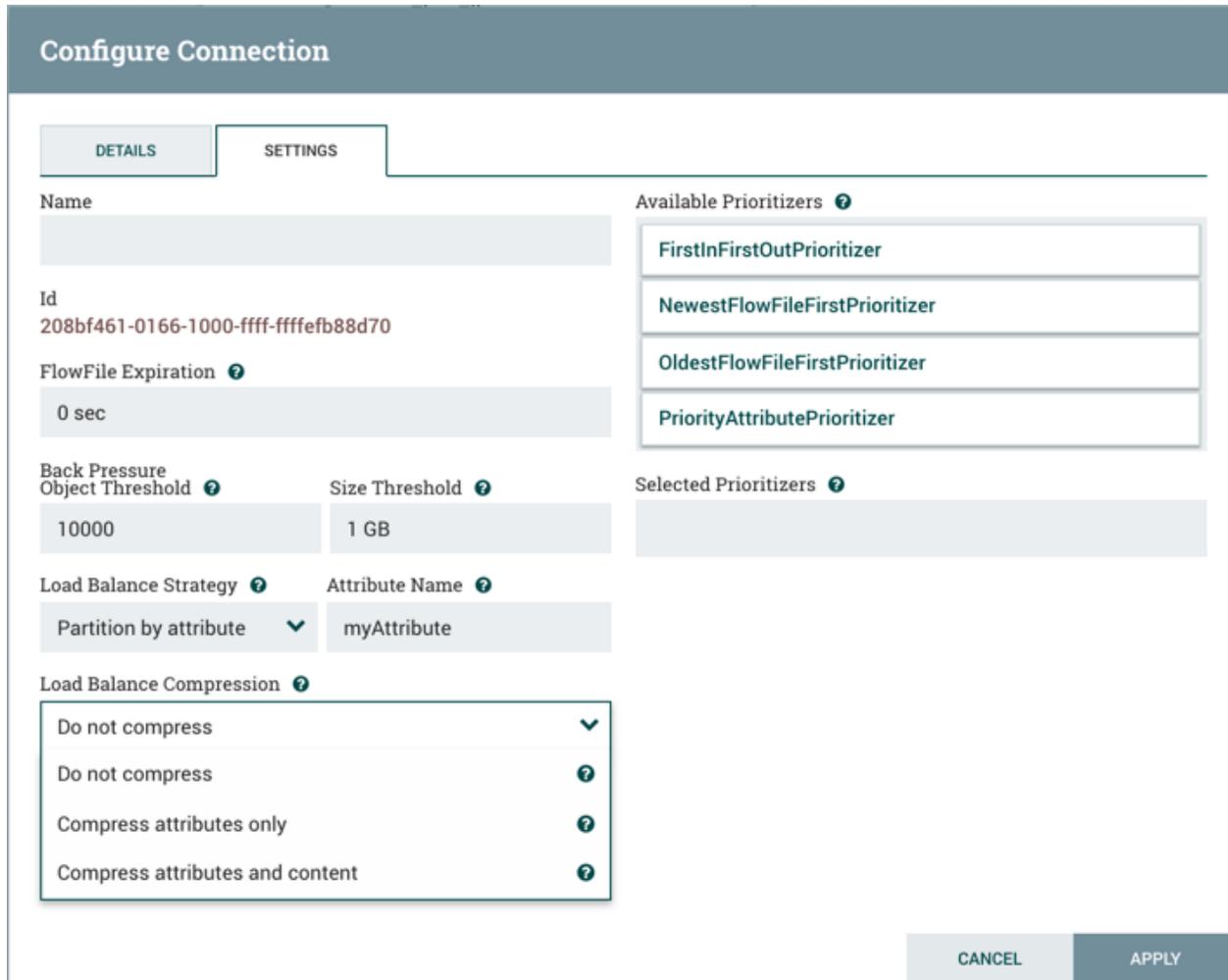


Note: NiFi persists the nodes that are in a cluster across restarts. This prevents the redistribution of data until all of the nodes have connected. If the cluster is shutdown and a node is not intended to be brought

back up, the user is responsible for removing the node from the cluster via the "Cluster" dialog in the UI (see Managing Nodes for more information).

Load Balance Compression

After selecting the load balance strategy, the user can configure whether or not data should be compressed when being transferred between nodes in the cluster.



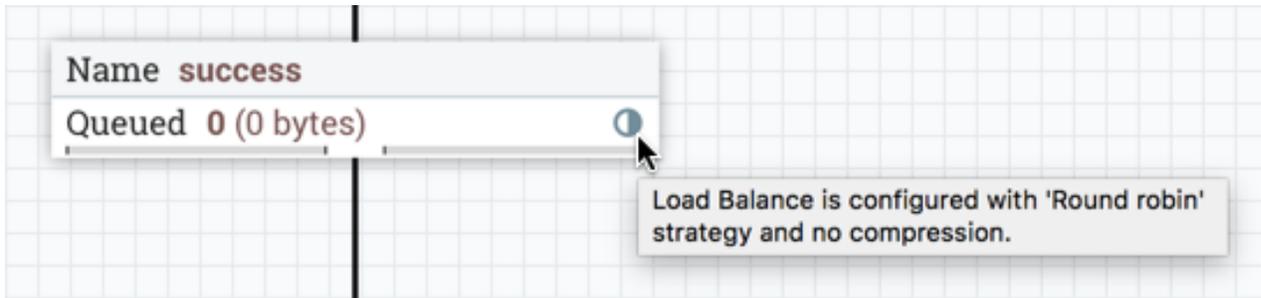
The following compression options are available:

- Do not compress: FlowFiles will not be compressed. This is the default.
- Compress attributes only: FlowFile attributes will be compressed, but FlowFile contents will not.
- Compress attributes and content: FlowFile attributes and contents will be compressed.

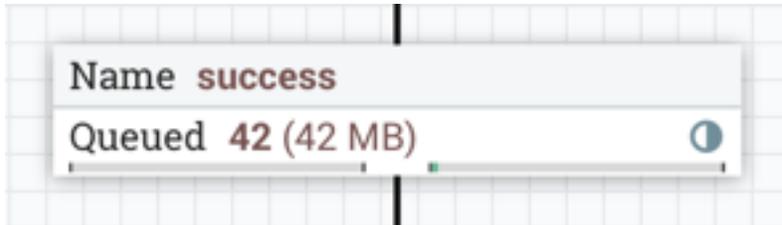
Load Balance Indicator

When a load balance strategy has been implemented for a connection, a load balance indicator

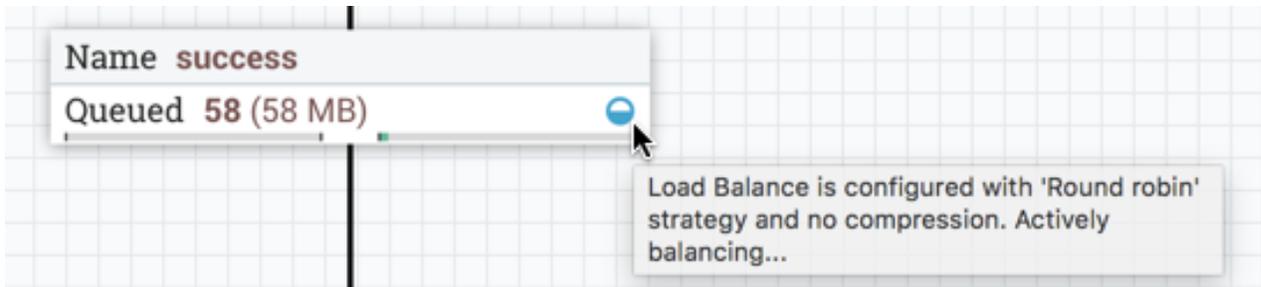
() will appear on the connection:



Hovering over the icon will display the connection's load balance strategy and compression configuration. The icon in this state also indicates that all data in the connection has been distributed across the cluster.

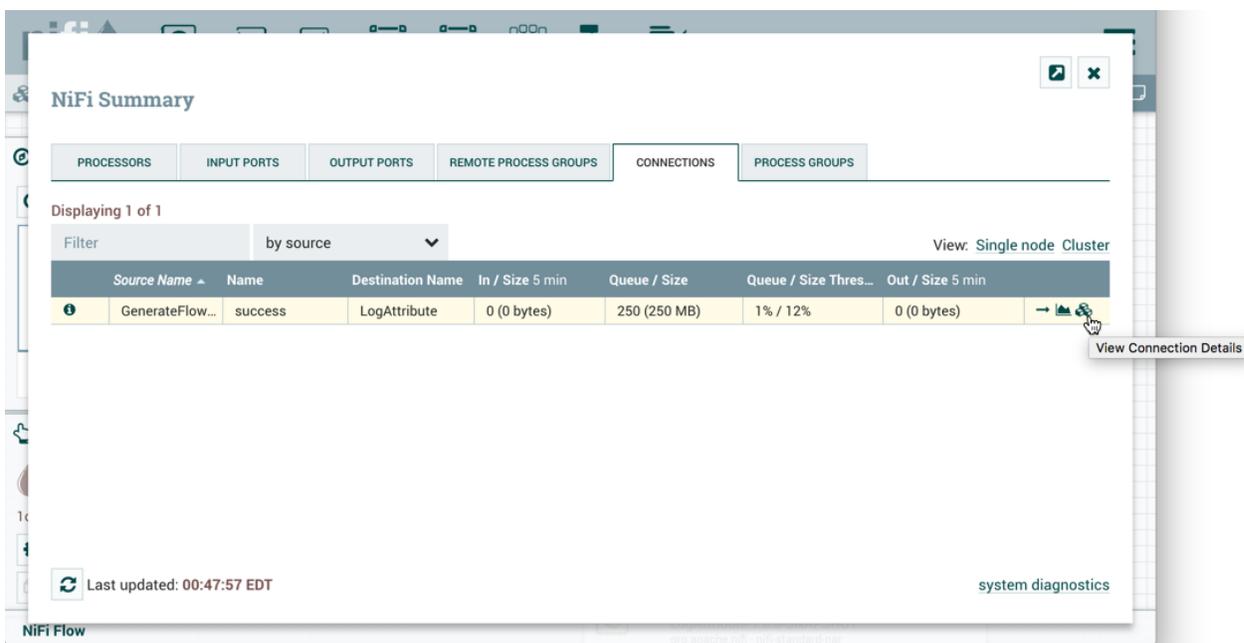


When data is actively being transferred between the nodes in the cluster, the load balance indicator will change orientation and color:

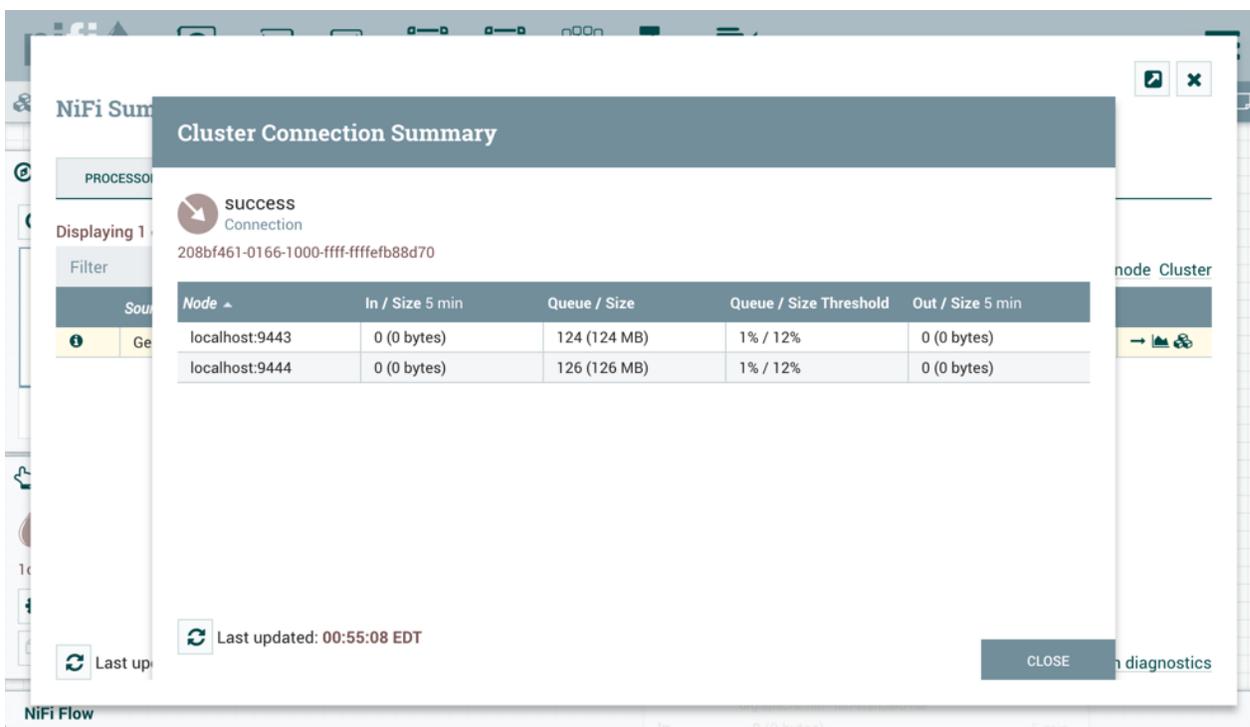


Cluster Connection Summary

To see where data has been distributed among the cluster nodes, select Summary from the Global Menu. Then select the "Connections" tab and the "View Connection Details" icon for a source:



This will open the Cluster Connection Summary dialog, which shows the data on each node in the cluster:



Prioritization

The right-hand side of the tab provides the ability to prioritize the data in the queue so that higher priority data is processed first. Prioritizers can be dragged from the top ('Available prioritizers') to the bottom ('Selected prioritizers'). Multiple prioritizers can be selected. The prioritizer that is at the top of the 'Selected prioritizers' list is the highest priority. If two FlowFiles have the same value according to this prioritizer, the second prioritizer will determine which FlowFile to process first, and so on. If a prioritizer is no longer desired, it can then be dragged from the 'Selected prioritizers' list to the 'Available prioritizers' list.

The following prioritizers are available:

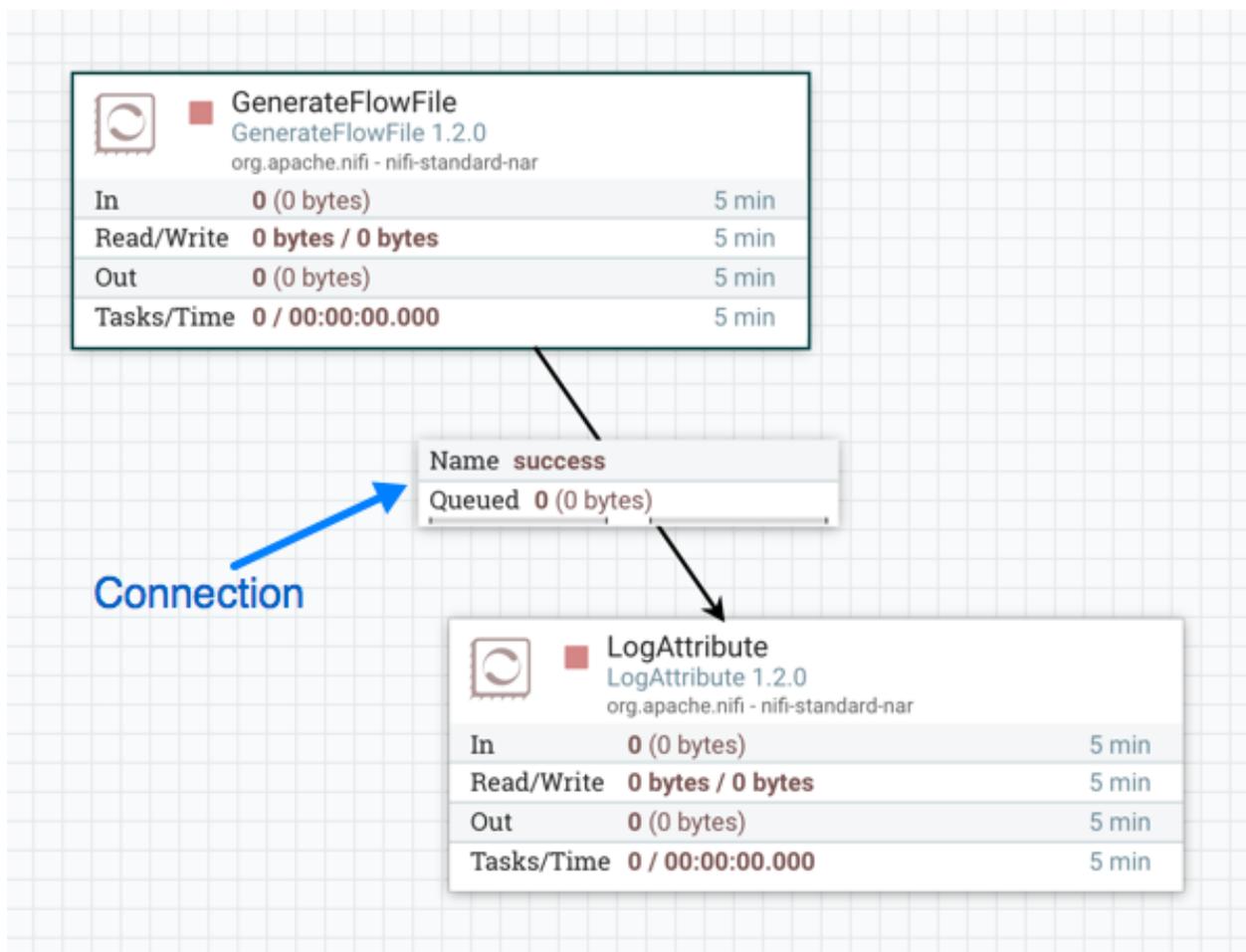
- FirstInFirstOutPrioritizer: Given two FlowFiles, the one that reached the connection first will be processed first.
- NewestFlowFileFirstPrioritizer: Given two FlowFiles, the one that is newest in the dataflow will be processed first.
- OldestFlowFileFirstPrioritizer: Given two FlowFiles, the one that is oldest in the dataflow will be processed first. 'This is the default scheme that is used if no prioritizers are selected'.
- PriorityAttributePrioritizer: Given two FlowFiles, an attribute called "priority" will be extracted. The one that has the lowest priority value will be processed first.
 - Note that an UpdateAttribute processor should be used to add the "priority" attribute to the FlowFiles before they reach a connection that has this prioritizer set.
 - If only one has that attribute it will go first.
 - Values for the "priority" attribute can be alphanumeric, where "a" will come before "z" and "1" before "9"
 - If "priority" attribute cannot be parsed as a long, unicode string ordering will be used. For example: "99" and "100" will be ordered so the flowfile with "99" comes first, but "A-99" and "A-100" will sort so the flowfile with "A-100" comes first.



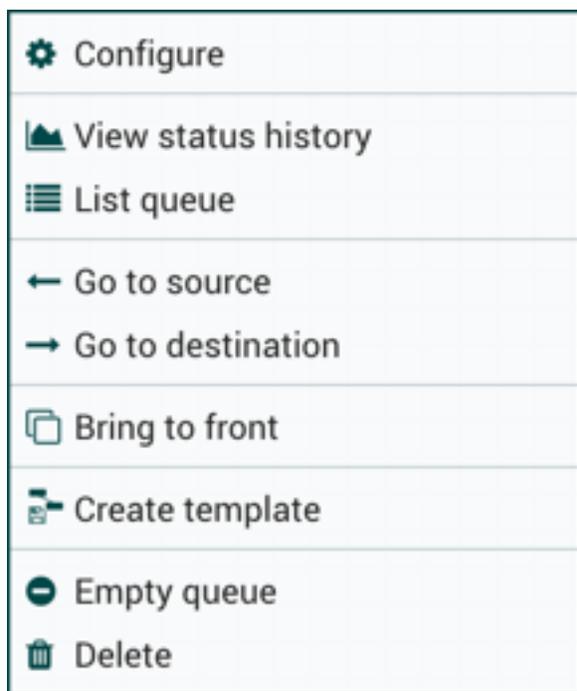
Note: With a Load Balance Strategy configured, the connection has a queue per node in addition to the local queue. The prioritizer will sort the data in each queue independently.

Changing Configuration and Context Menu Options

After a connection has been drawn between two components, the connection's configuration may be changed, and the connection may be moved to a new destination; however, the processors on either side of the connection must be stopped before a configuration or destination change may be made.



To change a connection's configuration or interact with the connection in other ways, right-click on the connection to open the connection context menu.

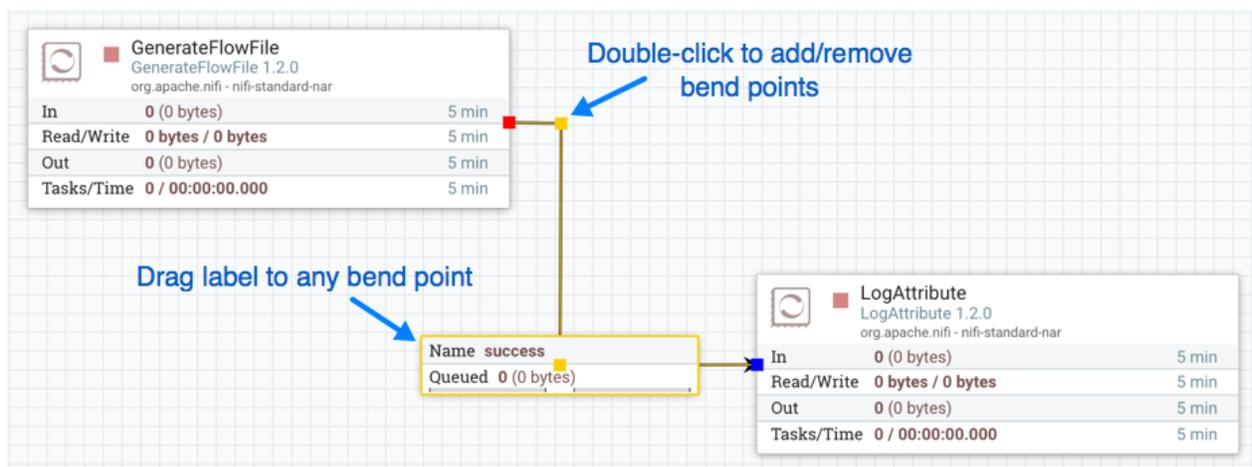


The following options are available:

- **Configure:** This option allows the user to change the configuration of the connection.
- **View status history:** This option opens a graphical representation of the connection's statistical information over time.
- **List queue:** This option lists the queue of FlowFiles that may be waiting to be processed.
- **Go to source:** This option can be useful if there is a long distance between the connection's source and destination components on the canvas. By clicking this option, the view of the canvas will jump to the source of the connection.
- **Go to destination:** Similar to the "Go to source" option, this option changes the view to the destination component on the canvas and can be useful if there is a long distance between two connected components.
- **Bring to front:** This option brings the connection to the front of the canvas if something else (such as another connection) is overlapping it.
- **Empty queue:** This option allows the DFM to clear the queue of FlowFiles that may be waiting to be processed. This option can be especially useful during testing, when the DFM is not concerned about deleting data from the queue. When this option is selected, users must confirm that they want to delete the data in the queue.
- **Delete:** This option allows the DFM to delete a connection between two components. Note that the components on both sides of the connection must be stopped and the connection must be empty before it can be deleted.

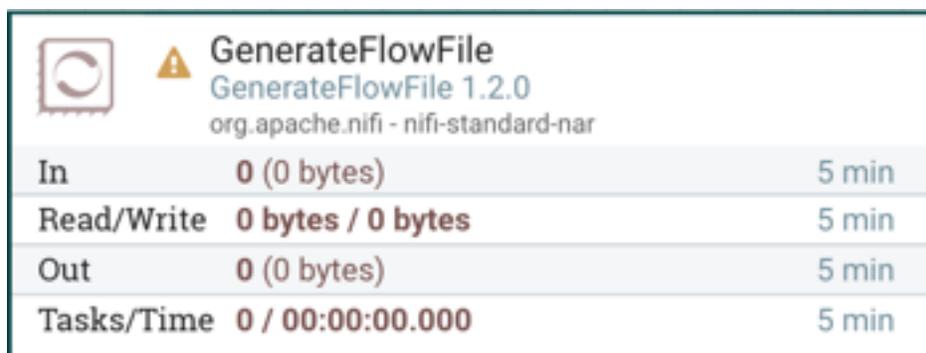
Bending Connections

To add a bend point (or elbow) to an existing connection, simply double-click on the connection in the spot where you want the bend point to be. Then, you can use the mouse to grab the bend point and drag it so that the connection is bent in the desired way. You can add as many bend points as you want. You can also use the mouse to drag and move the label on the connection to any existing bend point. To remove a bend point, simply double-click it again.

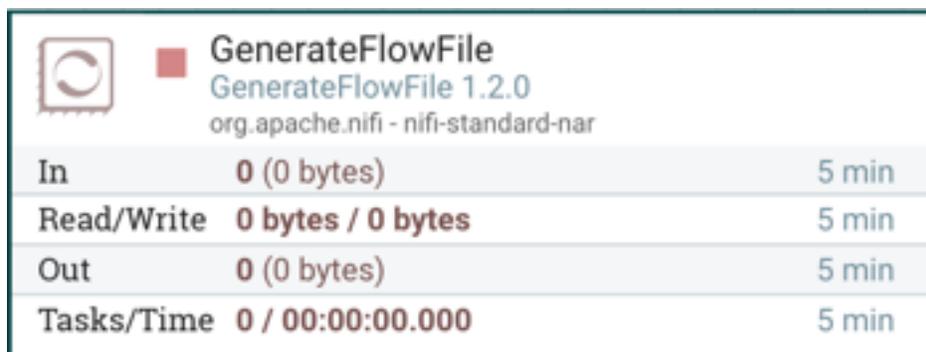


Processor Validation

Before trying to start a Processor, it's important to make sure that the Processor's configuration is valid. A status indicator is shown in the top-left of the Processor. If the Processor is invalid, the indicator will show a yellow Warning indicator with an exclamation mark indicating that there is a problem:



In this case, hovering over the indicator icon with the mouse will provide a tooltip showing all of the validation errors for the Processor. Once all of the validation errors have been addressed, the status indicator will change to a Stop icon, indicating that the Processor is valid and ready to be started but currently is not running:



Site-to-Site

When sending data from one instance of NiFi to another, there are many different protocols that can be used. The preferred protocol, though, is the NiFi Site-to-Site Protocol. Site-to-Site makes it easy to securely and efficiently

transfer data to/from nodes in one NiFi instance or data producing application to nodes in another NiFi instance or other consuming application.

Using Site-to-Site provides the following benefits:

- Easy to configure
 - After entering the URL(s) of the remote NiFi instance/cluster, the available ports (endpoints) are automatically discovered and provided in a drop-down list.
- Secure
 - Site-to-Site optionally makes use of Certificates in order to encrypt data and provide authentication and authorization. Each port can be configured to allow only specific users, and only those users will be able to see that the port even exists. For information on configuring the Certificates, see the *System Administration documentation* in the *Reference* section.
- Scalable
 - As nodes in the remote cluster change, those changes are automatically detected and data is scaled out across all nodes in the cluster.
- Efficient
 - Site-to-Site allows batches of FlowFiles to be sent at once in order to avoid the overhead of establishing connections and making multiple round-trip requests between peers.
- Reliable
 - Checksums are automatically produced by both the sender and receiver and compared after the data has been transmitted, in order to ensure that no corruption has occurred. If the checksums don't match, the transaction will simply be canceled and tried again.
- Automatically load balanced
 - As nodes come online or drop out of the remote cluster, or a node's load becomes heavier or lighter, the amount of data that is directed to that node will automatically be adjusted.
- FlowFiles maintain attributes
 - When a FlowFile is transferred over this protocol, all of the FlowFile's attributes are automatically transferred with it. This can be very advantageous in many situations, as all of the context and enrichment that has been determined by one instance of NiFi travels with the data, making for easy routing of the data and allowing users to easily inspect the data.
- Adaptable
 - As new technologies and ideas emerge, the protocol for handling Site-to-Site communications are able to change with them. When a connection is made to a remote NiFi instance, a handshake is performed in order to negotiate which protocol and which version of the protocol will be used. This allows new capabilities to be added while still maintaining backward compatibility with all older instances. Additionally, if a vulnerability or deficiency is ever discovered in a protocol, it allows a newer version of NiFi to forbid communication over the compromised versions of the protocol.

Site-to-Site is a protocol transferring data between two NiFi instances. Both end can be a standalone NiFi or a NiFi cluster. In this section, the NiFi instance initiates the communications is called Site-to-Site client NiFi instance and the other end as Site-to-Site server NiFi instance to clarify what configuration needed on each NiFi instances.

A NiFi instance can be both client and server for Site-to-Site protocol, however, it can only be a client or server within a specific Site-to-Site communication. For example, if there are three NiFi instances A, B and C. A pushes data to B, and B pulls data from C. A - push # B # pull - C. Then B is not only a server in the communication between A and B, but also a client in B and C.

It is important to understand which NiFi instance will be the client or server in order to design your data flow, and configure each instance accordingly. Here is a summary of what components run on which side based on data flow direction:

- Push: a client sends data to a Remote Process Group, the server receives it with an Input Port
- Pull: a client receives data from a Remote Process Group, the server sends data through an Output Port

Configure Site-to-Site client NiFi instance

Remote Process Group: In order to communicate with a remote NiFi instance via Site-to-Site, simply drag a *Remote Process Group* onto the canvas and enter the URL(s) of the remote NiFi instance (for more information on the components of a Remote Process Group, see the *Remote Process Group* section of this guide.) The URL is the same URL you would use to go to that instance's User Interface or in the case of a cluster, the URLs of the cluster nodes. At this point, you can drag a connection to or from the Remote Process Group in the same way you would drag a connection to or from a Processor or a local Process Group. When you drag the connection, you will have a chance to choose which Port to connect to. Note that it may take up to one minute for the Remote Process Group to determine which ports are available.

If the connection is dragged starting from the Remote Process Group, the ports shown will be the Output Ports of the remote group, as this indicates that you will be pulling data from the remote instance. If the connection instead ends on the Remote Process Group, the ports shown will be the Input Ports of the remote group, as this implies that you will be pushing data to the remote instance.



Note: If the remote instance is configured to use secure data transmission, you will see only ports that you are authorized to communicate with. For information on configuring NiFi to run securely, see the System Administrator's Guide.

Transport Protocol: On a Remote Process Group creation or configuration dialog, you can choose Transport Protocol to use for Site-to-Site communication as shown in the following image:

Configure Remote Process Group

Name
http://localhost:8080/nifi/

Id
a43a9b4e-015a-1000-6997-39a4e265a8b8

URLs
http://localhost:8080/nifi/

| | |
|---|---|
| <p>Transport Protocol ?</p> <div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between; align-items: center;"> RAW ▼ </div> | <p>Local Network Interface ?</p> <div style="border: 1px solid #ccc; height: 20px;"></div> |
| <p>HTTP Proxy Server Hostname ?</p> <div style="border: 1px solid #ccc; height: 20px;"></div> | <p>HTTP Proxy Server Port ?</p> <div style="border: 1px solid #ccc; height: 20px;"></div> |
| <p>HTTP Proxy User ?</p> <div style="border: 1px solid #ccc; height: 20px;"></div> | <p>HTTP Proxy Password ?</p> <div style="border: 1px solid #ccc; height: 20px;"></div> |
| <p>Communications Timeout ?</p> <div style="border: 1px solid #ccc; padding: 2px;">30 sec</div> | <p>Yield Duration ?</p> <div style="border: 1px solid #ccc; padding: 2px;">10 sec</div> |

CANCEL
APPLY

By default, it is set to RAW which uses raw socket communication using a dedicated port. HTTP transport protocol is especially useful if the remote NiFi instance is in a restricted network that only allow access through HTTP(S) protocol or only accessible from a specific HTTP Proxy server. For accessing through a HTTP Proxy Server, BASIC and DIGEST authentication are supported.

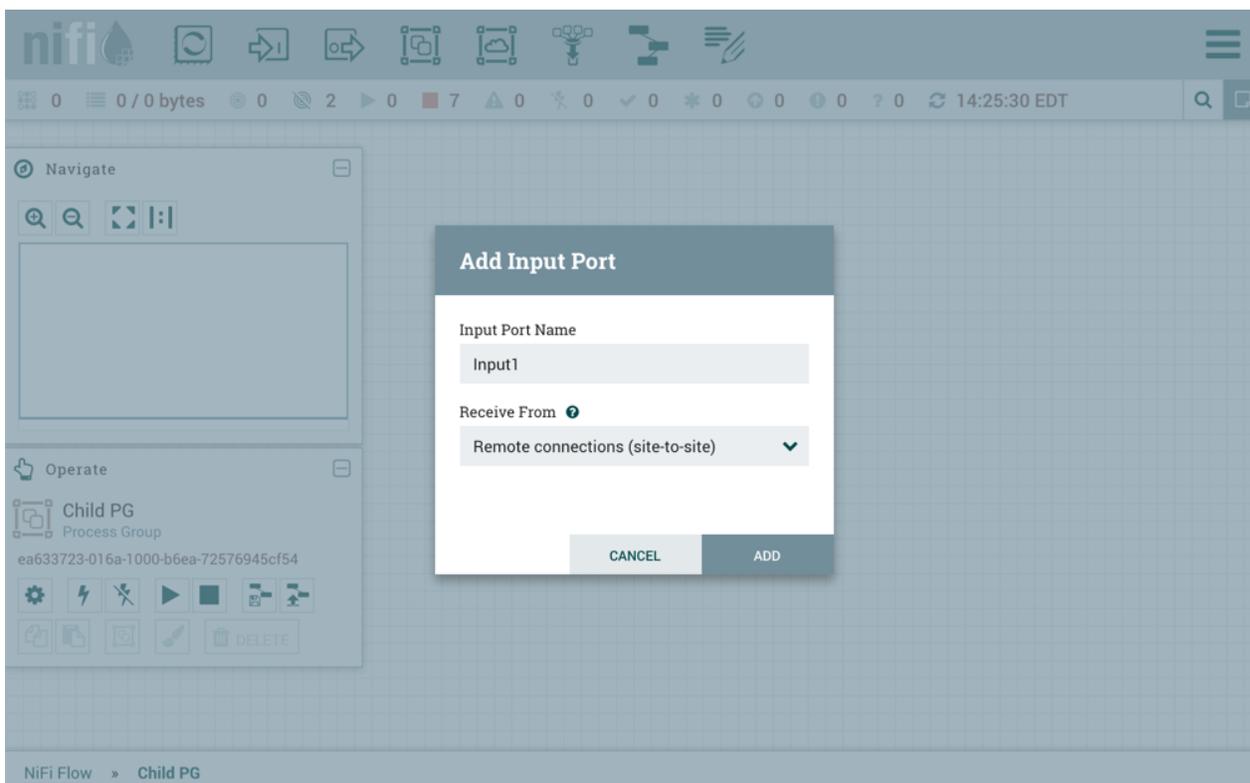
Local Network Interface: In some cases, it may be desirable to prefer one network interface over another. For example, if a wired interface and a wireless interface both exist, the wired interface may be preferred. This can be configured by specifying the name of the network interface to use in this box. If the value entered is not valid, the Remote Process Group will not be valid and will not communicate with other NiFi instances until this is resolved.

Configure Site-to-Site Server NiFi Instance

Retrieve Site-to-Site Details: If your NiFi is running securely, in order for another NiFi instance to retrieve information from your instance, it needs to be added to the Global Access "retrieve site-to-site details" policy. This will allow the other instance to query your instance for details such as name, description, available peers (nodes when clustered), statistics, OS port information and available Input and Output ports. Utilizing Input and Output ports in a secured instance requires additional policy configuration as described below.

Input Port: In order to allow another NiFi instance to push data to your local instance, you can simply drag an *input port* onto the Root Process Group of your canvas. After entering a name for the port, it will be added to your flow. You can now right-click on the Input Port and choose Configure in order to adjust the name and the number of concurrent tasks that are used for the port.

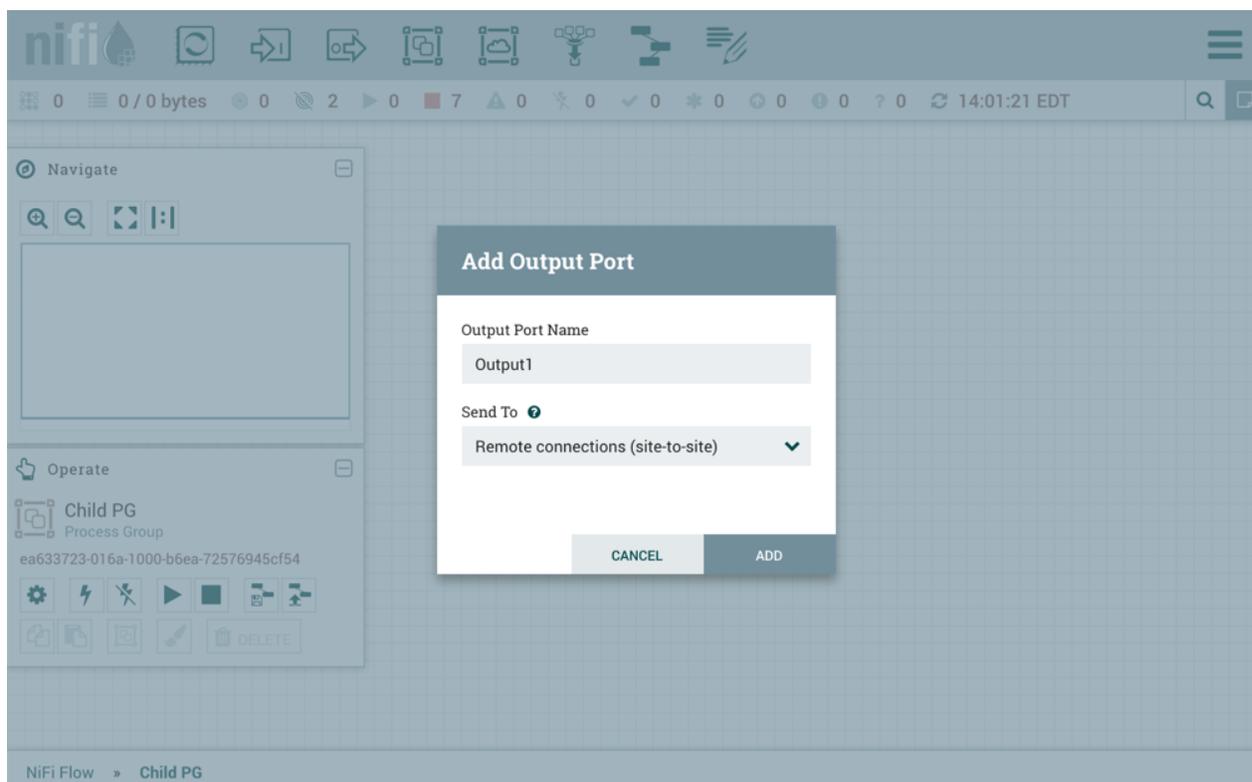
To create an Input Port for Site-to-Site in a child Process Group, enter the name for the port and select "Remote connections (site-to-site)" from the Receive From drop-down menu.



If Site-to-Site is configured to run securely, you will need to manage the input port's "receive data via site-to-site" component access policy. Only those users who have been added to the policy will be able to communicate with the port.

Output Port: Similar to an Input Port, a DataFlow Manager may choose to add an *output port* to the Root Process Group. The Output Port allows an authorized NiFi instance to remotely connect to your instance and pull data from the Output Port. After dragging an Output Port onto the canvas, right-click and choose Configure to adjust the name and how many concurrent tasks are allowed. Manage the output port's "receive data via site-to-site" component access policy to control which users are authorized to pull data from the instance being configured.

To create an Output Port for Site-to-Site in a child Process Group, enter the name for the port and select "Remote connections (site-to-site)" from the Send To drop-down menu.



In addition to other instances of NiFi, some other applications may use a Site-to-Site client in order to push data to or receive data from a NiFi instance. For example, NiFi provides an Apache Storm spout and an Apache Spark Receiver that are able to pull data from NiFi's Output Ports for Site-to-Site connections.



Note: For information on how to enable and configure Site-to-Site on a NiFi instance, see the Site-to-Site Properties section of the System Administrator's Guide.



Note: For information on how to configure access policies, see the Access Policies section of the System Administrator's Guide.

Example Dataflow

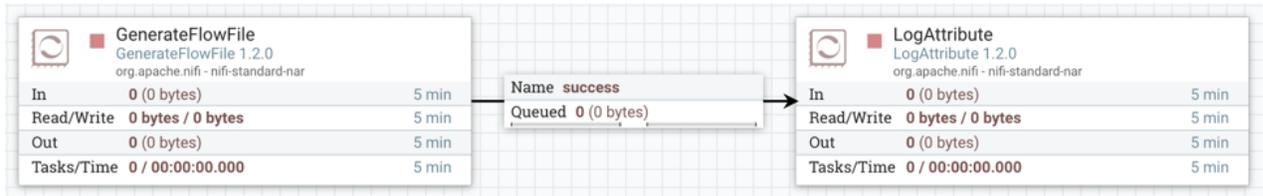
This section has described the steps required to build a dataflow. Now, to put it all together. The following example dataflow consists of just two processors: GenerateFlowFile and LogAttribute. These processors are normally used for testing, but they can also be used to build a quick flow for demonstration purposes and see NiFi in action.

After you drag the GenerateFlowFile and LogAttribute processors to the canvas and connect them (using the guidelines provided above), configure them as follows:

- Generate FlowFile
 - On the Scheduling tab, set Run schedule to: 5 sec. Note that the GenerateFlowFile processor can create many FlowFiles very quickly; that's why setting the Run schedule is important so that this flow does not overwhelm the system NiFi is running on.
 - On the Properties tab, set File Size to: 10 KB
- Log Attribute
 - On the Settings tab, under Auto-terminate relationships, select the checkbox next to Success. This will terminate FlowFiles after this processor has successfully processed them.

- Also on the Settings tab, set the Bulletin level to Info. This way, when the dataflow is running, this processor will display the bulletin icon (see *Anatomy of a Processor*), and the user may hover over it with the mouse to see the attributes that the processor is logging.

The dataflow should look like the following:



Now see the following section on how to start and stop the dataflow. When the dataflow is running, be sure to note the statistical information that is displayed on the face of each processor (see *Anatomy of a Processor*).