

Hortonworks DataFlow

User Guide

(July 27, 2017)

Hortonworks DataFlow: User Guide

Copyright © 2012-2017 Hortonworks, Inc. Some rights reserved.



Except where otherwise noted, this document is licensed under
Creative Commons Attribution ShareAlike 4.0 License.
<http://creativecommons.org/licenses/by-sa/4.0/legalcode>

Table of Contents

1. Apache NiFi User Guide	1
1.1. Introduction	1
1.2. Browser Support	1
1.2.1. Unsupported Browsers	2
1.2.2. Viewing the UI in Variably Sized Browsers	2
1.3. Terminology	2
1.4. NiFi User Interface	4
1.5. Accessing the UI with Multi-Tenant Authorization	6
1.6. Logging In	7
1.7. Building a DataFlow	8
1.7.1. Adding Components to the Canvas	8
1.7.2. Component Versions	17
1.7.3. Configuring a Processor	19
1.7.4. Additional Help	28
1.7.5. Using Custom Properties with Expression Language	28
1.7.6. Controller Services	29
1.7.7. Reporting Tasks	34
1.7.8. Connecting Components	36
1.7.9. Processor Validation	42
1.7.10. Site-to-Site	43
1.7.11. Example Dataflow	47
1.8. Command and Control of the DataFlow	48
1.8.1. Starting a Component	48
1.8.2. Stopping a Component	49
1.8.3. Enabling/Disabling a Component	49
1.8.4. Remote Process Group Transmission	50
1.9. Navigating within a DataFlow	52
1.9.1. Component Linking	52
1.10. Monitoring of DataFlow	53
1.10.1. Anatomy of a Processor	53
1.10.2. Anatomy of a Process Group	56
1.10.3. Anatomy of a Remote Process Group	58
1.10.4. Queue Interaction	60
1.10.5. Summary Page	60
1.10.6. Historical Statistics of a Component	63
1.11. Templates	64
1.11.1. Creating a Template	64
1.11.2. Importing a Template	65
1.11.3. Instantiating a Template	65
1.11.4. Managing Templates	66
1.12. Data Provenance	66
1.12.1. Provenance Events	67
1.12.2. Searching for Events	68
1.12.3. Details of an Event	70
1.12.4. Replaying a FlowFile	72
1.12.5. Viewing FlowFile Lineage	73
1.12.6. Encrypted Provenance Repository	76
1.13. Other Management Features	79

1. Apache NiFi User Guide

- [Introduction \[1\]](#)
- [Browser Support \[1\]](#)
- [Terminology \[2\]](#)
- [NiFi User Interface \[4\]](#)
- [Accessing the UI with Multi-Tenant Authorization \[6\]](#)
- [Logging In \[7\]](#)
- [Building a DataFlow \[8\]](#)
- [Command and Control of the DataFlow \[48\]](#)
- [Navigating within a DataFlow \[52\]](#)
- [Monitoring of DataFlow \[53\]](#)
- [Templates \[64\]](#)
- [Data Provenance \[66\]](#)
- [Other Management Features \[79\]](#)

1.1. Introduction

Apache NiFi is a dataflow system based on the concepts of flow-based programming. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic. NiFi has a web-based user interface for design, control, feedback, and monitoring of dataflows. It is highly configurable along several dimensions of quality of service, such as loss-tolerant versus guaranteed delivery, low latency versus high throughput, and priority-based queuing. NiFi provides fine-grained data provenance for all data received, forked, joined cloned, modified, sent, and ultimately dropped upon reaching its configured end-state.

See the [System Administrator's Guide](#) for information about system requirements, installation, and configuration. Once NiFi is installed, use a supported web browser to view the UI.

1.2. Browser Support

Browser	Version
Chrome	Current and Current - 1
FireFox	Current and Current - 1
Edge	Current and Current - 1
Safari	Current and Current - 1

Current and Current - 1 indicates that the UI is supported in the current stable release of that browser and the preceding one. For instance, if the current stable release is 45.X then the officially supported versions will be 45.X and 44.X.

For Safari, which releases major versions much less frequently, Current and Current - 1 simply represent the two latest releases.

The supported browser versions are driven by the capabilities the UI employs and the dependencies it uses. UI features will be developed and tested against the supported browsers. Any problem using a supported browser should be reported to Apache NiFi.

1.2.1. Unsupported Browsers

While the UI may run successfully in unsupported browsers, it is not actively tested against them. Additionally, the UI is designed as a desktop experience and is not currently supported in mobile browsers.

1.2.2. Viewing the UI in Variably Sized Browsers

In most environments, all of the UI is visible in your browser. However, the UI has a responsive design that allows you to scroll through screens as needed, in smaller sized browsers or tablet environments.

In environments where your browser width is less than 800 pixels and the height less than 600 pixels, portions of the UI may become unavailable.

1.3. Terminology

DataFlow Manager: A DataFlow Manager (DFM) is a NiFi user who has permissions to add, remove, and modify components of a NiFi dataflow.

FlowFile: The FlowFile represents a single piece of data in NiFi. A FlowFile is made up of two components: FlowFile Attributes and FlowFile Content. Content is the data that is represented by the FlowFile. Attributes are characteristics that provide information or context about the data; they are made up of key-value pairs. All FlowFiles have the following Standard Attributes:

- **uuid:** A unique identifier for the FlowFile
- **filename:** A human-readable filename that may be used when storing the data to disk or in an external service
- **path:** A hierarchically structured value that can be used when storing data to disk or an external service so that the data is not stored in a single directory

Processor: The Processor is the NiFi component that is used to listen for incoming data; pull data from external sources; publish data to external sources; and route, transform, or extract information from FlowFiles.

Relationship: Each Processor has zero or more Relationships defined for it. These Relationships are named to indicate the result of processing a FlowFile. After a Processor has finished processing a FlowFile, it will route (or "transfer") the FlowFile to one of the Relationships. A DFM is then able to connect each of these Relationships to other components in order to specify where the FlowFile should go next under each potential processing result.

Connection: A DFM creates an automated dataflow by dragging components from the Components part of the NiFi toolbar to the canvas and then connecting the components together via Connections. Each connection consists of one or more Relationships. For each Connection that is drawn, a DFM can determine which Relationships should be used for the Connection. This allows data to be routed in different ways based on its processing outcome. Each connection houses a FlowFile Queue. When a FlowFile is transferred to a particular Relationship, it is added to the queue belonging to the associated Connection.

Controller Service: Controller Services are extension points that, after being added and configured by a DFM in the User Interface, will start up when NiFi starts up and provide information for use by other components (such as processors or other controller services). A common Controller Service used by several components is the StandardSSLContextService. It provides the ability to configure keystore and/or truststore properties once and reuse that configuration throughout the application. The idea is that, rather than configure this information in every processor that might need it, the controller service provides it for any processor to use as needed.

Reporting Task: 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 in the User Interface as desired. Common reporting tasks include the ControllerStatusReportingTask, MonitorDiskUsage reporting task, MonitorMemory reporting task, and the StandardGangliaReporter.

Funnel: A funnel is a NiFi component that is used to combine the data from several Connections into a single Connection.

Process Group: When a dataflow becomes complex, it often is beneficial to reason about the dataflow at a higher, more abstract level. NiFi allows multiple components, such as Processors, to be grouped together into a Process Group. The NiFi User Interface then makes it easy for a DFM to connect together multiple Process Groups into a logical dataflow, as well as allowing the DFM to enter a Process Group in order to see and manipulate the components within the Process Group.

Port: Dataflows that are constructed using one or more Process Groups need a way to connect a Process Group to other dataflow components. This is achieved by using Ports. A DFM can add any number of Input Ports and Output Ports to a Process Group and name these ports appropriately.

Remote Process Group: Just as data is transferred into and out of a Process Group, it is sometimes necessary to transfer data from one instance of NiFi to another. While NiFi provides many different mechanisms for transferring data from one system to another, Remote Process Groups are often the easiest way to accomplish this if transferring data to another instance of NiFi.

Bulletin: The NiFi User Interface provides a significant amount of monitoring and feedback about the current status of the application. In addition to rolling statistics and the current status provided for each component, components are able to report Bulletins. Whenever a component reports a Bulletin, a bulletin icon is displayed on that component. System-level bulletins are displayed on the Status bar near the top of the page. Using the mouse to hover over that icon will provide a tool-tip that shows the time and severity (Debug, Info, Warning, Error) of the Bulletin, as well as the message of the Bulletin. Bulletins from all components can also be viewed and filtered in the Bulletin Board Page, available in the Global Menu.

Template: Often times, a dataflow is comprised of many sub-flows that could be reused. NiFi allows DFMs to select a part of the dataflow (or the entire dataflow) and create a Template. This Template is given a name and can then be dragged onto the canvas just like the other components. As a result, several components may be combined together to make a larger building block from which to create a dataflow. These templates can also be exported as XML and imported into another NiFi instance, allowing these building blocks to be shared.

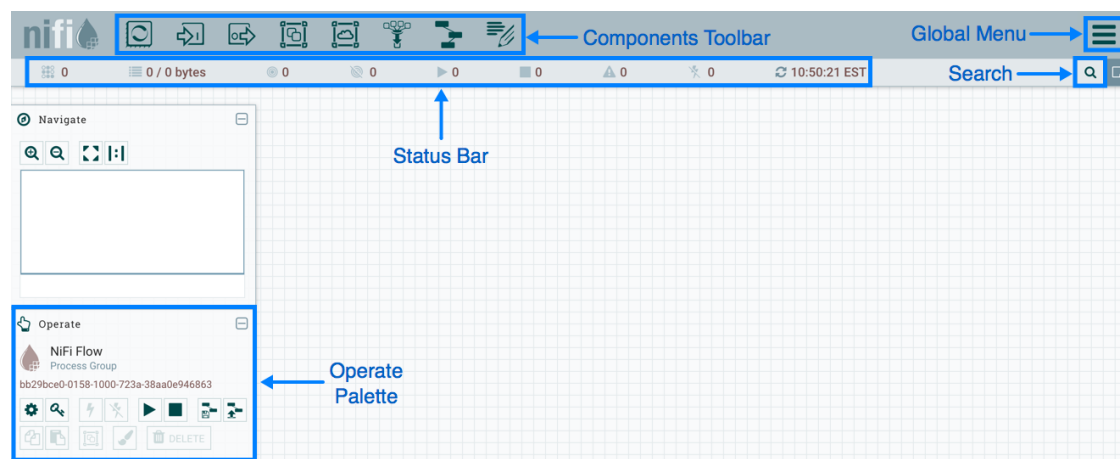
flow.xml.gz: Everything the DFM puts onto the NiFi User Interface canvas is written, in real time, to one file called the flow.xml.gz. This file is located in the nifi/conf directory by default. Any change made on the canvas is automatically saved to this file, without the user needing to click a "save" button. In addition, NiFi automatically creates a backup copy of this file in the archive directory when it is updated. You can use these archived files to rollback flow configuration. To do so, stop NiFi, replace flow.xml.gz with a desired backup copy, then restart NiFi. In a clustered environment, stop the entire NiFi cluster, replace the flow.xml.gz of one of nodes, and restart the node. Remove flow.xml.gz from other nodes. Once you confirmed the node starts up as a one-node cluster, start the other nodes. The replaced flow configuration will be synchronized across the cluster. The name and location of flow.xml.gz, and auto archive behavior are configurable. See the [System Administrator's Guide](#) for further details.

1.4. NiFi User Interface

The NiFi UI provides mechanisms for creating automated dataflows, as well as visualizing, editing, monitoring, and administering those dataflows. The UI can be broken down into several segments, each responsible for different functionality of the application. This section provides screenshots of the application and highlights the different segments of the UI. Each segment is discussed in further detail later in the document.

When the application is started, the user is able to navigate to the UI by going to the default address of `http://<hostname>:8080/nifi` in a web browser. There are no permissions configured by default, so anyone is able to view and modify the dataflow. For information on securing the system, see the [System Administrator's Guide](#).

When a DFM navigates to the UI for the first time, a blank canvas is provided on which a dataflow can be built:

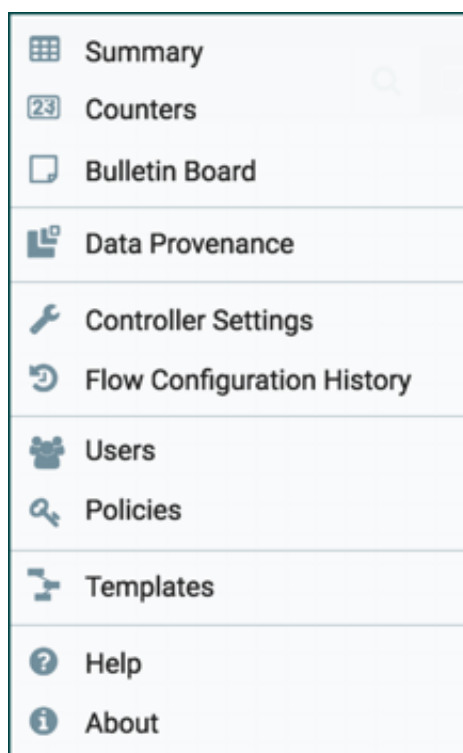


The Components Toolbar runs across the top left portion of your screen. It consists of the components you can drag onto the canvas to build your dataflow. Each component is described in more detail in [Building a Dataflow](#).

The Status Bar is under the Components Toolbar. The Status bar provides information about how many Processors exist on the canvas in each state (Stopped, Running, Invalid, Disabled), how many Remote Process Groups exist on the canvas in each state (Transmitting, Not Transmitting), the number of threads that are currently active in the flow, the amount of data that currently exists in the flow, and the timestamp at which all of this information was last refreshed. Additionally, if the instance of NiFi is clustered, the Status bar shows how many nodes are in the cluster and how many are currently connected.

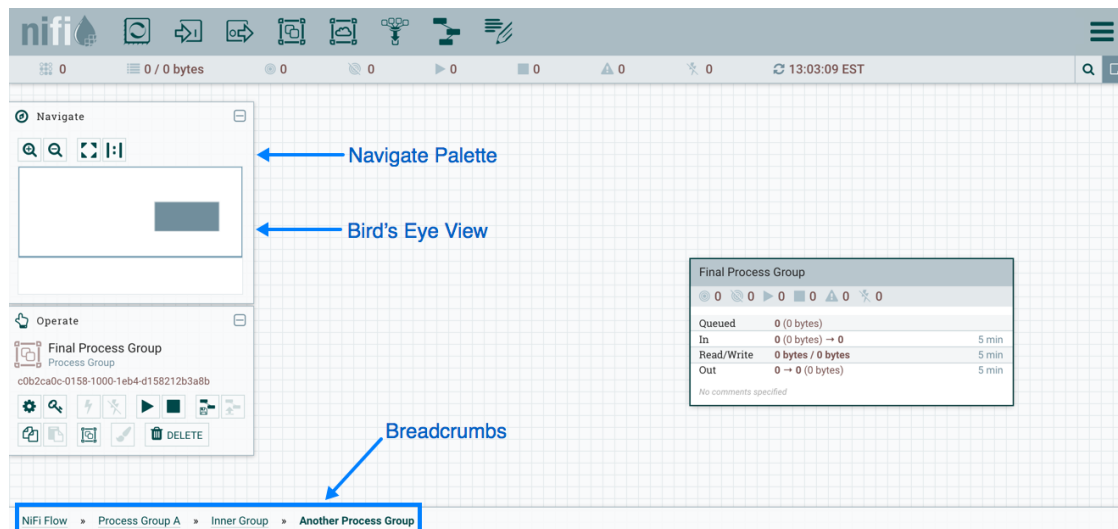
The Operate Palette sits to the left-hand side of the screen. It consists of buttons that are used by DFMs to manage the flow, as well as by administrators who manage user access and configure system properties, such as how many system resources should be provided to the application.

On the right side of the canvas is Search, and the Global Menu. You can use Search to easily find components on the canvas and can to search by component name, type, identifier, configuration properties, and their values. The Global Menu contains options that allow you to manipulate existing components on the canvas:



Additionally, the UI has some features that allow you to easily navigate around the canvas. You can use the Navigate Palette to pan around the canvas, and to zoom in and out. The "Birds Eye View" of the dataflow provides a high-level view of the dataflow and allows you to pan across large portions of the dataflow. You can also find breadcrumbs along the bottom of the screen. As you navigate into and out of Process Groups, the breadcrumbs show the depth in the flow, and each Process Group that you entered to reach this depth.

Each of the Process Groups listed in the breadcrumbs is a link that will take you back up to that level in the flow.



1.5. Accessing the UI with Multi-Tenant Authorization

Multi-tenant authorization enables multiple groups of users (tenants) to command, control, and observe different parts of the dataflow, with varying levels of authorization. When an authenticated user attempts to view or modify a NiFi resource, the system checks whether the user has privileges to perform that action. These privileges are defined by policies that you can apply system wide or to individual components. What this means from a Dataflow Manager perspective is that once you have access to the NiFi canvas, a range of functionality is visible and available to you, depending on the privileges assigned to you.

The available global access policies are:

Policy	Privilege
view the UI	Allows users to view the UI
access the controller	Allows users to view and modify the controller including reporting tasks, Controller Services, and nodes in the cluster
query provenance	Allows users to submit a provenance search and request even lineage
access restricted components	Allows users to create/modify restricted components assuming otherwise sufficient permissions
access all policies	Allows users to view and modify the policies for all components
access users/groups	Allows users view and modify the users and user groups
retrieve site-to-site details	Allows other NiFi instances to retrieve Site-To-Site details
view system diagnostics	Allows users to view System Diagnostics
proxy user requests	Allows proxy machines to send requests on the behalf of others
access counters	Allows users to view and modify counters

The available component-level access policies are:

Policy	Privilege
view the component	Allows users to view component configuration details
modify the component	Allows users to modify component configuration details
view the data	Allows users to view metadata and content for this component through provenance data and flowfile queues in outbound connection
modify the data	Allows users to empty flowfile queues in outbound connections and to submit replays
view the policies	Allows users to view the list of users who can view and modify a component
modify the policies	Allows users to modify the list of users who can view and modify a component
retrieve data via site-to-site	Allows a port to receive data from NiFi instances
send data via site-to-site	Allows a port to send data from NiFi instances

If you are unable to view or modify a NiFi resource, contact your System Administrator or see [Configuring Users and Access Policies](#) in the *System Administrator's Guide* for more information.

1.6. Logging In

If NiFi is configured to run securely, users will be able to request access to the DataFlow. For information on configuring NiFi to run securely, see the [System Administrator's Guide](#). If NiFi supports anonymous access, users will be given access accordingly and given an option to log in.

Clicking the *login* link will open the log in page. If the user is logging in with their username/password they will be presented with a form to do so. If NiFi is not configured to support anonymous access and the user is logging in with their username/password, they will be immediately sent to the login form bypassing the canvas.

The screenshot shows a web interface for logging into NiFi. On the left is the NiFi logo, which consists of a blue water drop with a white grid pattern inside. To the right of the logo, the text 'Log In' is displayed in a bold, dark blue font. Further to the right, there is a link labeled 'home' in a smaller, lighter blue font. Below the 'Log In' heading, there are two input fields. The first is labeled 'Username' and contains the placeholder text 'username'. The second is labeled 'Password' and contains the placeholder text 'password'. At the bottom right of the form, there is a dark blue button with the text 'LOG IN' in white capital letters.

1.7. 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.

1.7.1. 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

Tag Cloud:

amazon
archive
attributes
avro
aws
database
fetch
files
filesystem
get
hadoop
http
ingest
insert
json
listen
logs
message
put
remote
source
split
sql
text
update

Displaying 165 of 165 Filter processor list

Type ^	Tags
AttributesToJSON	flowfile, json, attributes
Base64EncodeContent	encode, base64
CompressContent	lzma, decompress, compress, snappy framed, ...
ConsumeAMQP	receive, amqp, rabbit, get, consume, message
ConsumeJMS	jms, receive, get, consume, message
ConsumeKafka	PubSub, Consume, Ingest, Get, Kafka, Ingress, ...
ConsumeMQTT	MQTT, subscribe, consume, listen, IOT
ConsumeWindowsEventLog	event, windows, ingest
ControlRate	throttle, rate, rate control, throughput
ConvertAvroSchema	convert, kite, avro
ConvertAvroToJSON	json, convert, avro
ConvertAvroToORC	hive orc convert avro

Selected Processor:

AttributesToJSON

Generates a JSON representation of the input FlowFile Attributes. The resulting JSON can be written to either a new Attribute 'JSONAttributes' or written to the FlowFile as content.

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 data via HTTP, we can select both the `http` Tag and the `ingest` Tag:

Add Processor

Tag Cloud:

amazonarchiveattributesavroawsdatabasefetchfilesfilesystemgethadoophttpingestinputinsertjsonlistenlogsmessageputremotesourcetcptextupdate

ingest ✕

http ✕

Displaying 2 of 159 Filter processor list

Type ^	Tags
GetHTTP	input, get, fetch, http, poll, https, source, ingest
ListenHTTP	rest, http, https, listen, ingest

Selected Processor:

GetHTTP

Fetches data from an HTTP or HTTPS URL and writes the data to the content of a FlowFile. Once the content has been fetched, the ETag and Last Modified dates are remembered (if the web server supports these concepts). This allows the Processor to fetch new data only if the remote data has changed or until the state is cleared. That is, once the content has been fetched from the given URL, it

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.

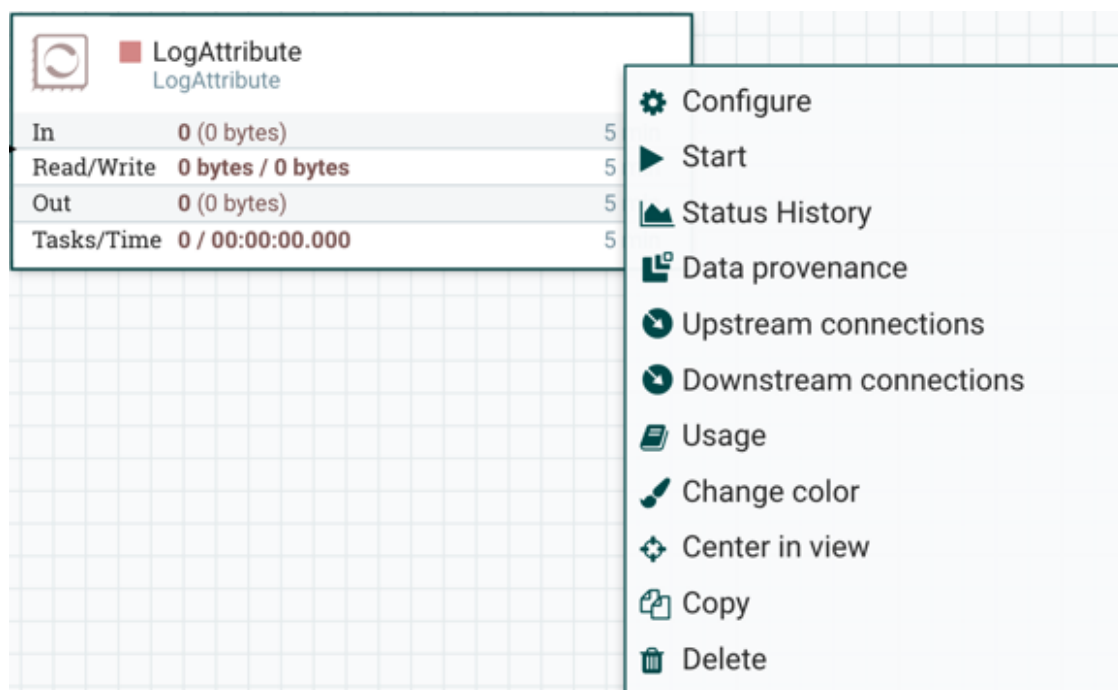
Before a user is allowed to create and modify restricted components they must be granted access to restricted components. Refer to [multi-tenant](#) documentation.

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.

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](#)).
- **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.
- **Status History:** This option opens a graphical representation of the Processor's statistical information over time.
- **Upstream connections:** 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.
- **Downstream connections:** 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.
- **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](#)).

- **Usage:** This option takes the user to the Processor's usage documentation.
- **Change color:** This option allows the user to change the color of the Processor, which can make the visual management of large flows easier.
- **Center in view:** This option centers the view of the canvas on the given 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 Administrator's Guide](#).



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 Administrator's Guide](#).

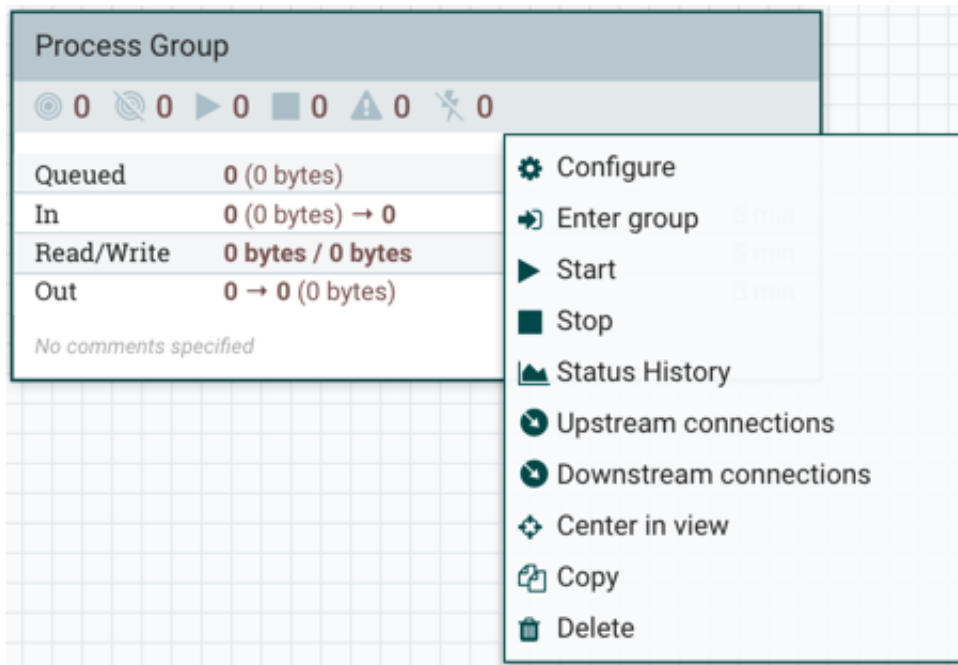


Process

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 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.
- **Enter group:** This option allows the user to enter the Process Group. 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.
- **Status History:** This option opens a graphical representation of the Process Group's statistical information over time.
- **Upstream connections:** This option allows the user to see and "jump to" upstream connections that are coming into the Process Group.
- **Downstream connections:** 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.

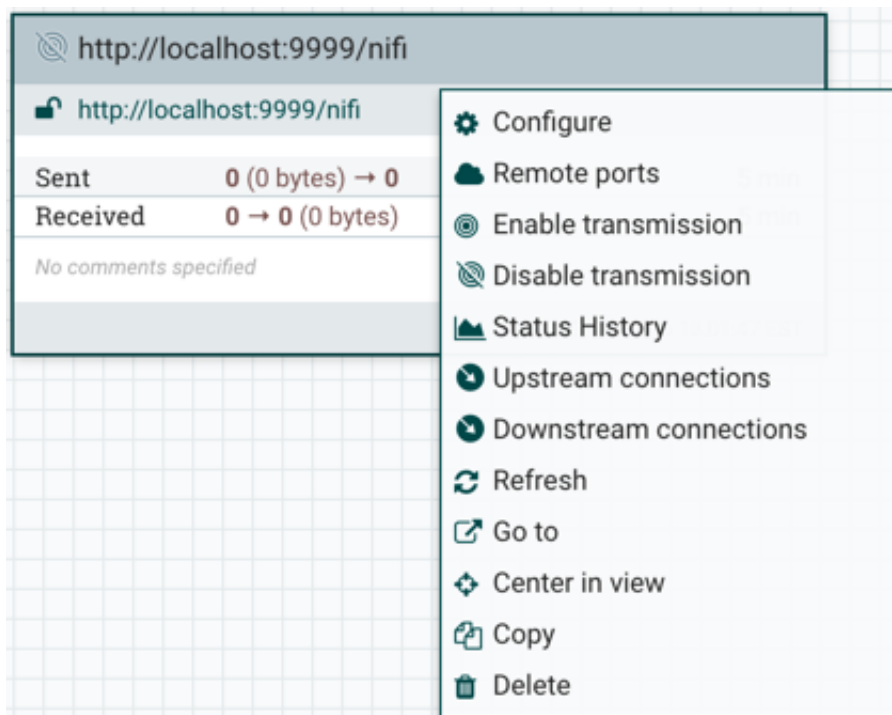
- **Copy:** This option places a copy of the selected Process Group on the clipboard, so that it may be pasted elsewhere on the canvas by right-clicking on the canvas and selecting Paste. The Copy/Paste actions also may be done using the keystrokes Ctrl-C (Command-C) and Ctrl-V (Command-V).
- **Delete:** 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, the URL that should be used is the URL of any NiFi instance in that cluster. 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 may interact with it by right-clicking on the Remote Process Group and selecting an option from 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 Remote Process Group:

- **Configure:** This option allows the user to establish or change the configuration of the Remote Process Group.
- **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.
- **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.
- **Status History:** This option opens a graphical representation of the Remote Process Group's statistical information over time.
- **Upstream connections:** This option allows the user to see and "jump to" upstream connections that are coming into the Remote Process Group.
- **Downstream connections:** This option allows the user to see and "jump to" downstream connections that are going out of the Remote Process Group.
- **Refresh:** This option refreshes the view of the status of the remote NiFi instance.
- **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.
- **Center in view:** This option centers the view of the canvas on the given 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.



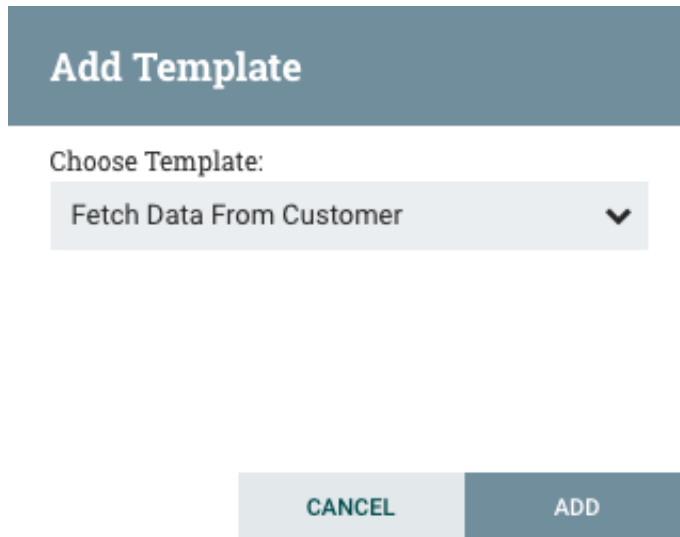
Funnel:

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.

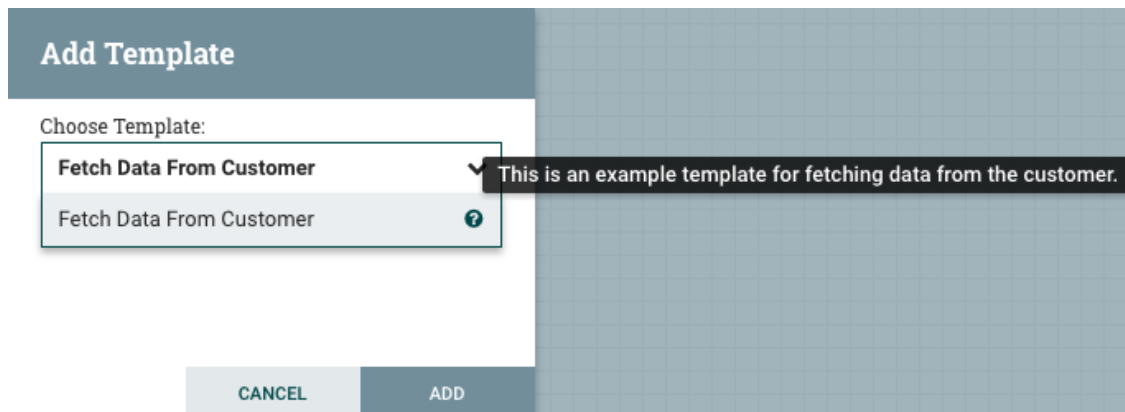


Template:

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:



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`

1.7.2. 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 processor. Each component displayed on the canvas also contains this information.

1.7.2.1. Sorting and Filtering Components

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

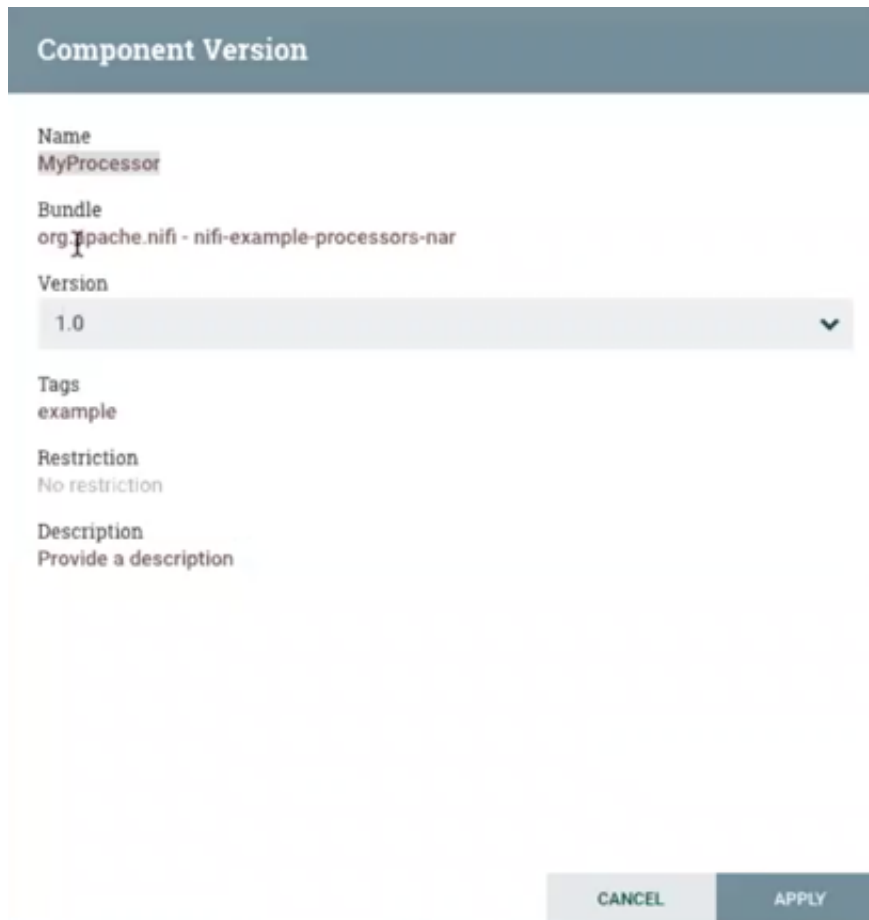
To sort based on version, click the version column to display in ascending or descending version order.

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

1.7.2.2. 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

Tags
example

Restriction
No restriction

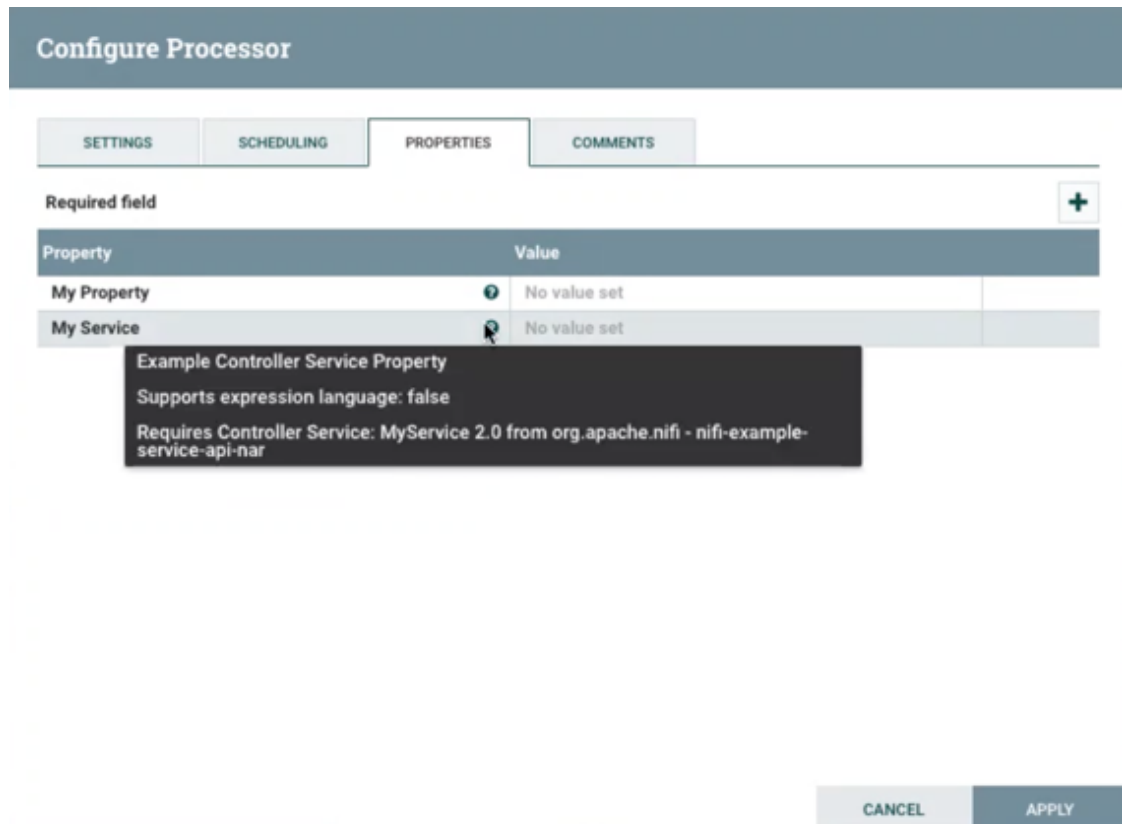
Description
Provide a description

CANCEL APPLY

1.7.2.3. 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.



1.7.3. Configuring a Processor

To configure a processor, right-click on the Processor and select the `Configure` option from the context menu. 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 the `Apply` button or cancel all changes by clicking the `Cancel` button.

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.

1.7.3.1. Settings Tab

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

Configure Processor

SETTINGS
SCHEDULING
PROPERTIES
COMMENTS

Name
RouteOnAttribute Enabled

Id
1f422785-0156-1000-0000-00007ec882e9

Type
RouteOnAttribute

Penalty duration ?
30 sec

Bulletin level ?
WARN ▼

Auto terminate relationships ?
 unmatched
FlowFiles that do not match any user-define expression will be routed here

Yield duration ?
1 sec

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. 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 'Auto-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.

1.7.3.2. Scheduling Tab

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

Configure Processor

SETTINGS | **SCHEDULING** | PROPERTIES | COMMENTS

Scheduling strategy ⓘ
Timer driven ▼

Concurrent tasks ⓘ
1

Run duration ⓘ
0ms 25ms 50ms 100ms 250ms 500ms 1s 2s
Lower latency Higher throughput

Run schedule ⓘ
0 sec

CANCEL APPLY

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 [Chron Trigger Tutorial](#) in the Quartz documentation.

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.

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.

When configured for clustering, an Execution setting will be available. This setting is used to determine 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.

The right-hand side of the 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.

1.7.3.3. 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:

Configure Processor

SETTINGS SCHEDULING **PROPERTIES** COMMENTS

Required field +

Property	Value
Routing Strategy ?	Route to Property name ▼ CANCEL OK

CANCEL APPLY

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.

Configure Processor

SETTINGS SCHEDULING **PROPERTIES** COMMENTS

Required field +

Property	Value
Routing Strategy ?	Route to Property name
route 1 ?	Some value 🗑️
route 2 ?	<div style="border: 1px solid #ccc; padding: 5px;"> 1 Some other value </div>

Empty CANCEL OK

CANCEL APPLY

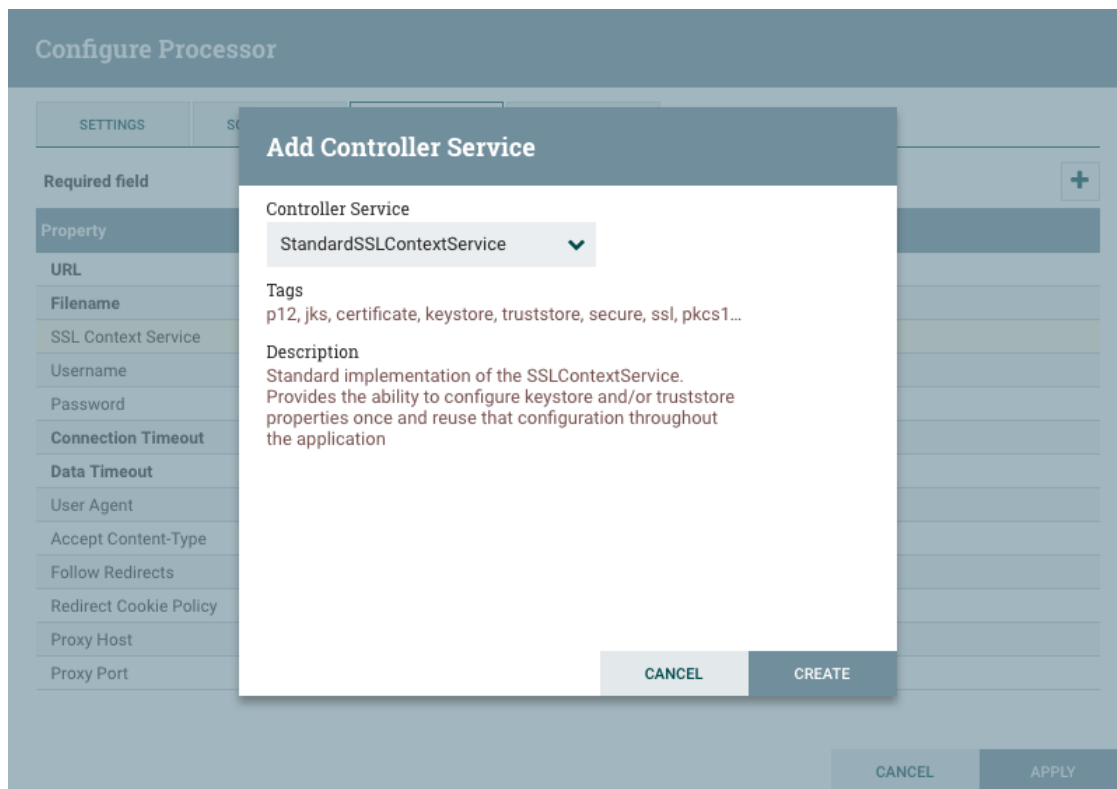
Note that after a User-Defined property has been added, an icon will appear on the right-hand side of that row (



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

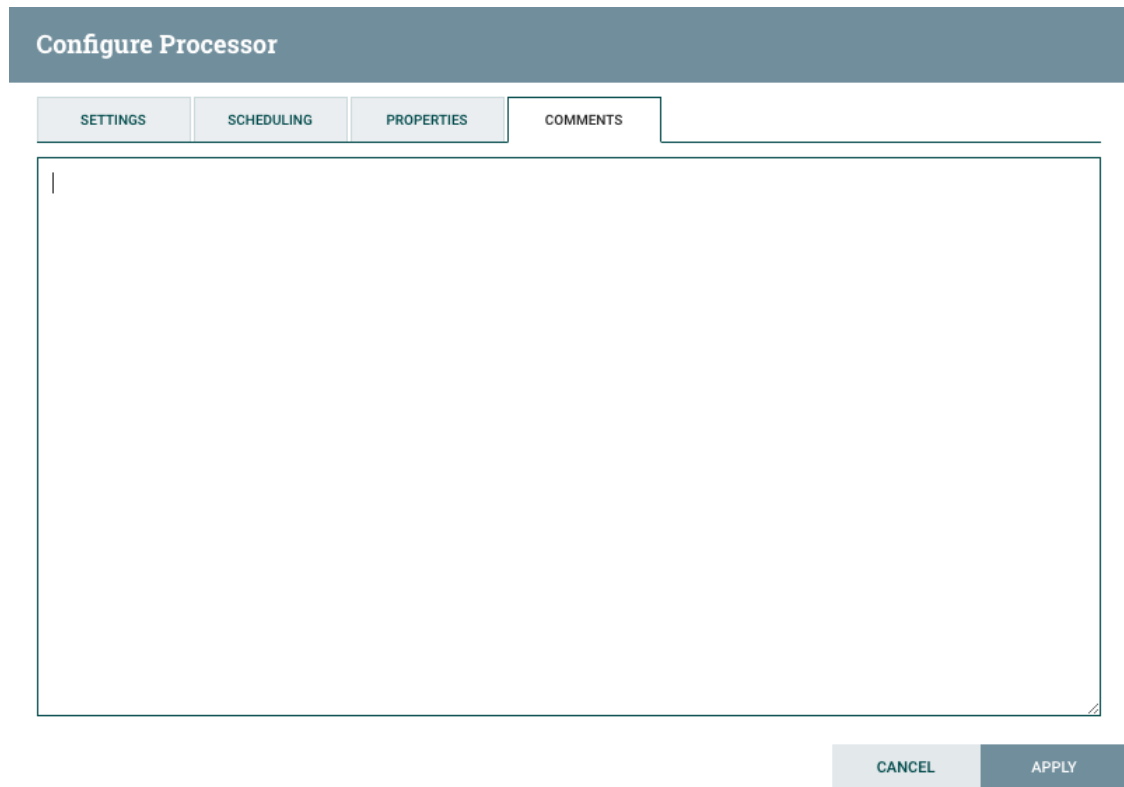
Some processors also have an Advanced User Interface (UI) built into them. For example, the UpdateAttribute processor has an Advanced UI. To access the Advanced UI, click the *Advanced* button that appears at the bottom of the Configure Processor window. Only processors that have an Advanced UI will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



1.7.3.4. 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:



1.7.4. 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.

1.7.5. 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.

In addition to using FlowFile attributes, system properties, and environment properties within Express 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.

To create custom properties for use with Expression Language, identify one or more sets of key/value pairs, and give them to your system administrator.

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

For more information on Expression Language, see the [Expression Language Guide](#). For information on how to define custom properties, see the [System Administrator's Guide](#).

1.7.6. Controller Services

Controller Services are available for reporting tasks, processors, and other services to utilize for configuration or task execution. You can use the NiFi UI to add Controller Services for either reporting tasks or dataflows.

Your ability to view and add Controller Services is dependent on the roles and 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. Roles and privileges can be assigned on a global or Controller Service-specific basis.

Controller Services are not reporting task or dataflow specific. You have access to the full set of available Controller Services whether you are adding it for a reporting task or a dataflow.

1.7.6.1. Adding Controller Settings 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 NiFi Settings window has three tabs across the top: General, Controller Services, and Reporting Tasks. The General tab is for settings that pertain to general information about the NiFi instance. For example, here, the DFM can provide a unique name for the overall dataflow, as well as comments that describe the flow. Be aware that 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).

The General tab also provides settings for the overall maximum thread counts of the instance.

NiFi Settings

GENERAL

CONTROLLER SERVICES

REPORTING TASKS

Maximum timer driven thread count ?

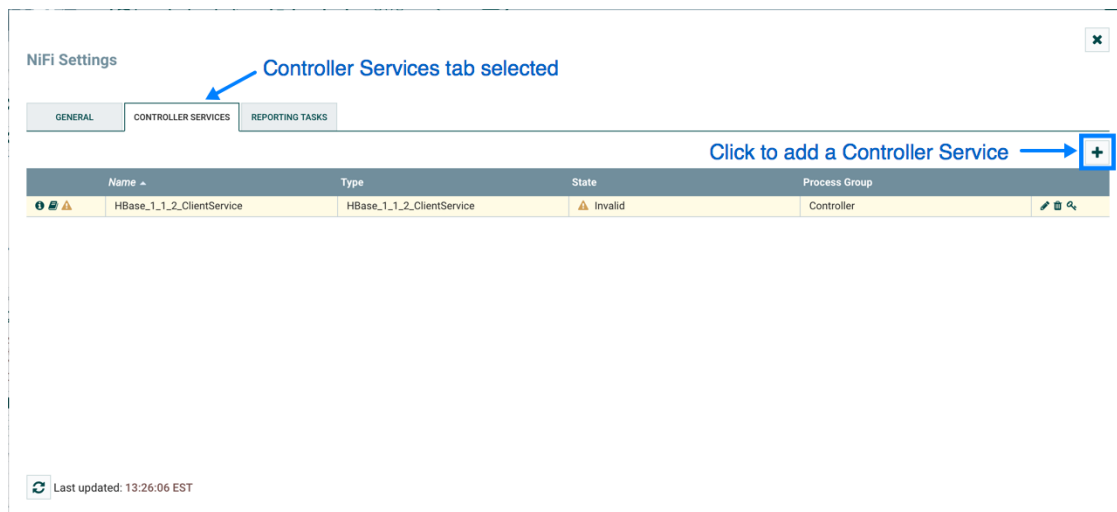
10

Maximum event driven thread count ?

5

APPLY

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



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 of the window to search for the desired Controller Service. Upon selecting a Controller Service from the list, the DFM can see a description of the 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

Tag Cloud: **aws** **cache** **client** **cluster** **connection** **couchbase** **credentials** **database** **dbcp** **distinct** **distributed** **hbase** **hive** **jdbc** **jms** **key/value** **map** **nosql** **pooling** **provider** **server** **set** **state** **store** **subscribe**

Displaying 8 of 12

Type	Tags
AWSCredentialsProviderControllerService	credentials, provider, aws
CouchbaseClusterService	database, couchbase, connection, nosql
DistributedMapCacheClientService	cluster, cache, distributed, state, map
DistributedMapCacheServer	cluster, server, cache, key/value, distributed, map
DistributedSetCacheClientService	cluster, cache, set, distributed, state
DistributedSetCacheServer	server, cache, set, distributed, distinct
HBase_1_1_2_ClientService	client, hbase
StandardSSLContextService	p12, jks, certificate, keystore, truststore, secure, ssl, p...

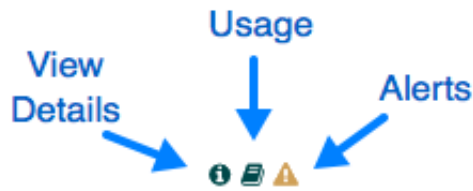
Selected Controller Service:
StandardSSLContextService
Standard implementation of the SSLContextService. Provides the ability to configure keystore and/or truststore properties once and reuse that configuration throughout the application

[CANCEL](#) [ADD](#)

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



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



When the DFM clicks the Edit 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 for the service and provides a list of other components (processors or other controller services) that reference the service.

Configure Controller Service

SETTINGS	PROPERTIES	COMMENTS
<p>Name</p> <div style="border: 1px solid #ccc; padding: 2px;">StandardSSLContextService</div>	<p>Referencing Components ⓘ</p> <p><i>No referencing components.</i></p>	
<p>Id</p> <p>1f7be494-0156-1000-0000-00004ce12c48</p>		
<p>Type</p> <p>StandardSSLContextService</p>		

CANCEL
APPLY

The Properties tab lists the various properties that apply to the particular controller service. As with configuring processors, the DFM may hover the 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 ?	No value set
Keystore Type ?	No value set
Truststore Filename ?	No value set
Truststore Password ?	No value set
Truststore Type ?	No value set
SSL Protocol ?	TLS

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 the Apply button to apply the configuration and close the window, or click the Cancel button to cancel the changes and close the window.

Note that 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 of the Controller Settings window. Then, in order to modify an existing/running controller service, the DFM needs to stop/disable it (as well as all referencing processors, reporting tasks, and controller services). 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. Likewise, when enabling a controller service, the DFM has the option to start/enable all referencing processors, reporting tasks, and controller services.

1.7.6.2. 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, you add a Controller Service only for use with the selected Process Group.

In either case, use the following steps to add a Controller Service:

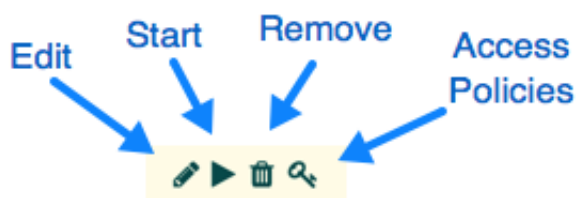
1. Click Configure, either from the Operate Palette, or from the Process Group context menu.
2. From the Process Group Configuration page, select the Controller Services tab.
3. Click the Add button to display the Add Controller Service dialog.
4. Select the Controller Service you want to add, and click Add.
5. Perform any necessary Controller Service configuration tasks by clicking the View Details icon () in the left-hand column.

1.7.7. Reporting Tasks

The Reporting Tasks tab behaves similarly to the Controller Services tab. The DFM has the option to add Reporting Tasks and configure them in the same way as Controller Services.



Once a Reporting Task has been added, the DFM may configure it by clicking the Edit (pencil icon) in the far-right column. Other buttons in this column include the Start button, Remove button, and Usage button, which links to the documentation for the particular Reporting Task.



When the DFM clicks the Edit button, a Configure Reporting Task window opens. It has three tabs: Settings, Properties, and Comments. This window is also 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 a UUID for the Reporting 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: ControllerStatusReportingTask Enabled

Scheduling Strategy ⓘ: Timer driven

Id: bbf6bb2a-0158-1000-637e-e793ceb7cd8b

Run Schedule ⓘ: 5 mins

Type: ControllerStatusReportingTask

CANCEL APPLY

The Properties tab for a Reporting Task lists the 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
Show Deltas	? 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 the Apply button to apply the configuration and close the window, or click Cancel to cancel the changes and close the window.




When you want to run the Reporting Task, click the Start button in the far-right column of the Reporting Tasks tab.

1.7.8. 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 			
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.

1.7.8.1. 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.

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.

1.7.8.2. Settings

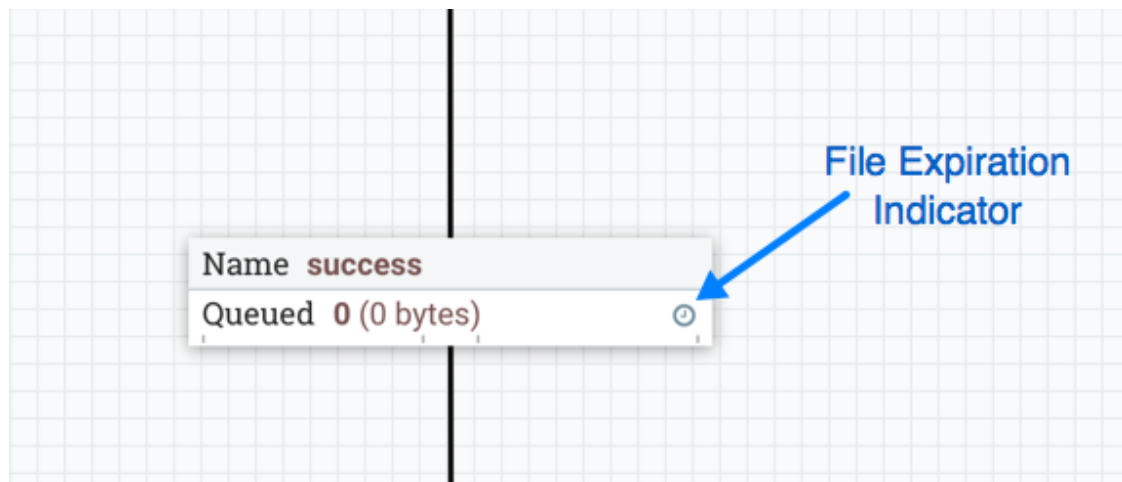
The Settings Tab provides the ability to configure the Connection's name, FlowFile expiration, Back Pressure thresholds, and Prioritization:

The screenshot shows the 'Create Connection' dialog box with the 'SETTINGS' tab selected. The 'Name' field is empty. The 'Id' field shows 'No value set'. The 'FlowFile Expiration' field is set to '0 sec'. The 'Back Pressure Object Threshold' field is set to '10000'. The 'Back Pressure Data Size Threshold' field is set to '1 GB'. The 'Available Prioritizers' list includes: FirstInFirstOutPrioritizer, NewestFlowFileFirstPrioritizer, OldestFlowFileFirstPrioritizer, and PriorityAttributePrioritizer. The 'Selected Prioritizers' field is empty. The 'CANCEL' and 'ADD' buttons are at the bottom right.

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.

1.7.8.2.1. 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.

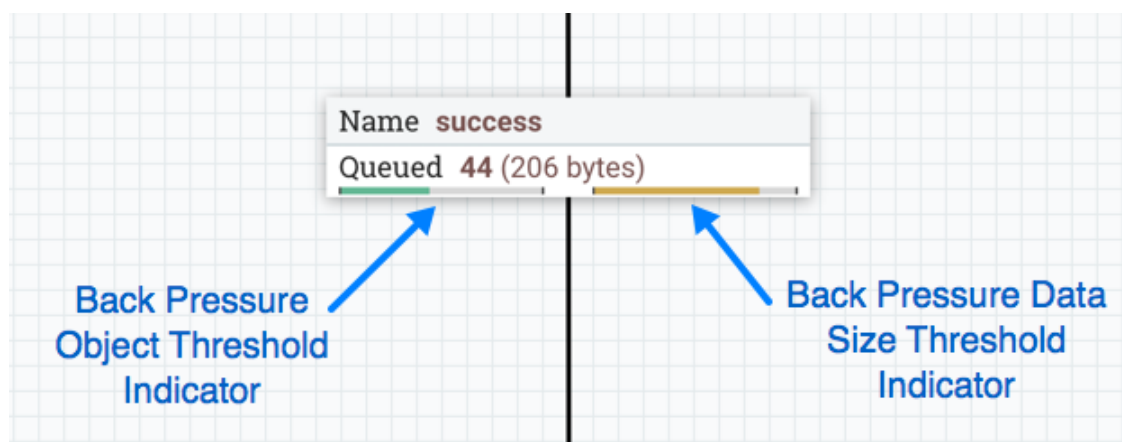


1.7.8.2.2. Back Pressure

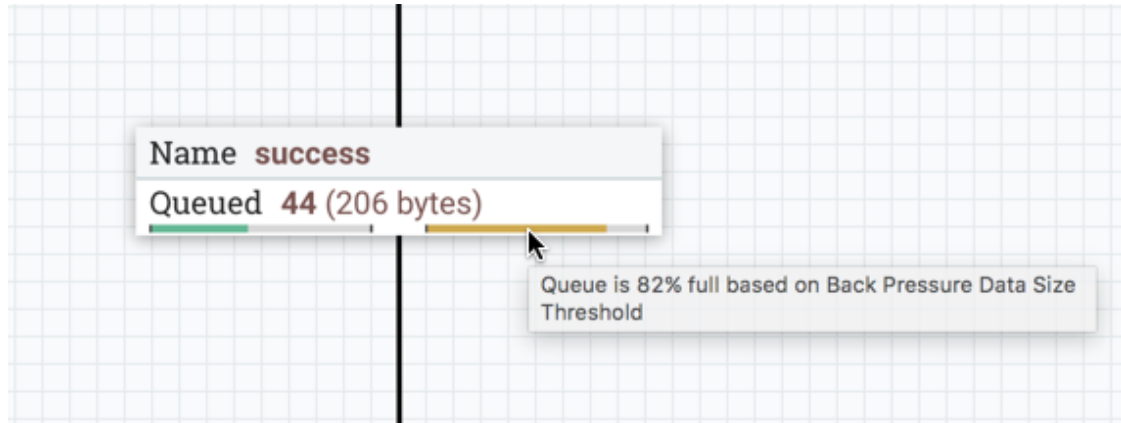
NiFi provides two configuration elements for Back Pressure. These thresholds indicate how much data should be allowed to exist in the queue before the component that is the source of the Connection is no longer scheduled to run. This allows the system to avoid being overrun with data. The first option 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).

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.

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.



1.7.8.2.3. Prioritization

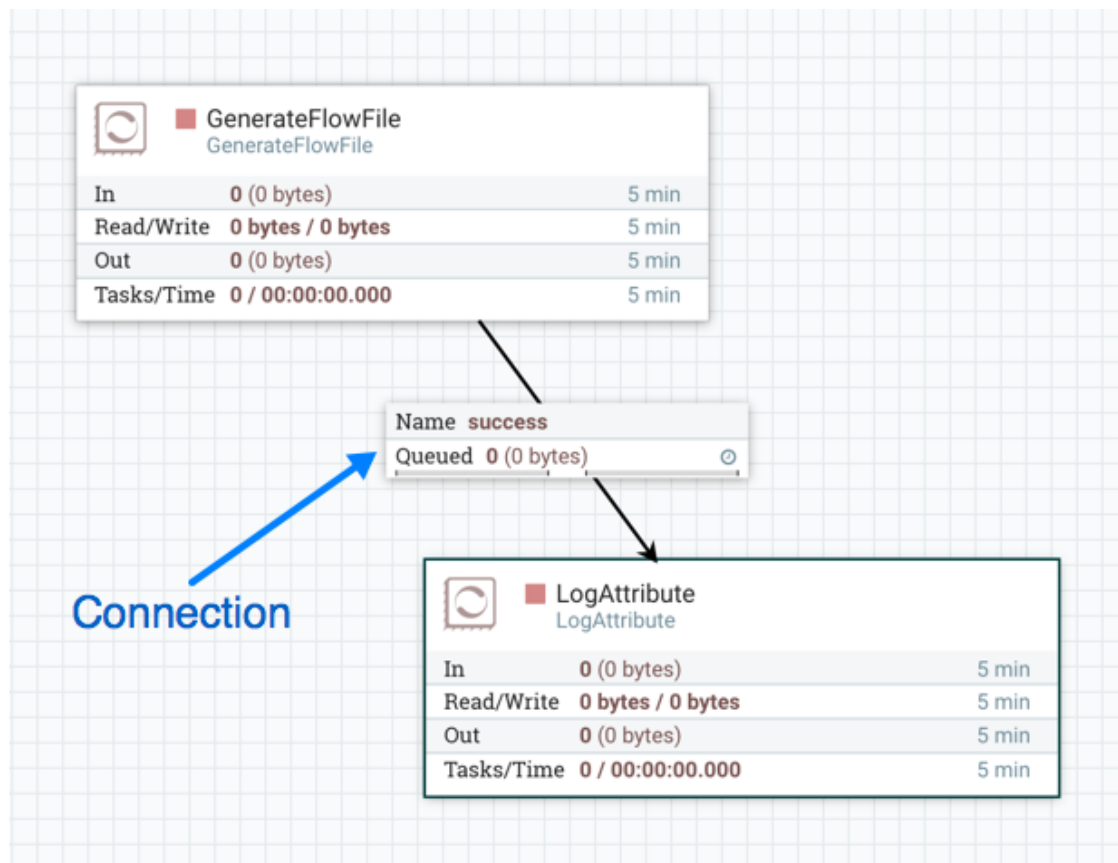
The right-hand side of the tab provides the ability to prioritize the data in the queue so that higher priority data is processed first. Prioritizers can be dragged from the top ('Available prioritizers') to the bottom ('Selected prioritizers'). Multiple prioritizers can be selected. The prioritizer 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:

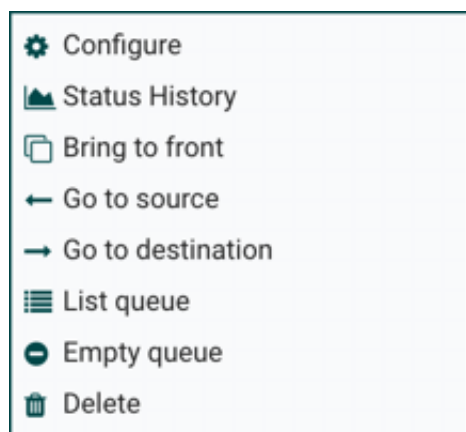
- **FirstInFirstOutPrioritizer:** Given two FlowFiles, the one that reached the connection first will be processed first.
- **NewestFlowFileFirstPrioritizer:** Given two FlowFiles, the one that is newest in the dataflow will be processed first.
- **OldestFlowFileFirstPrioritizer:** Given two FlowFiles, 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 that both have a "priority" attribute, the one that has the highest 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. Values for the "priority" attribute may be alphanumeric, where "a" is a higher priority than "z", and "1" is a higher priority than "9", for example.

1.7.8.2.4. 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.

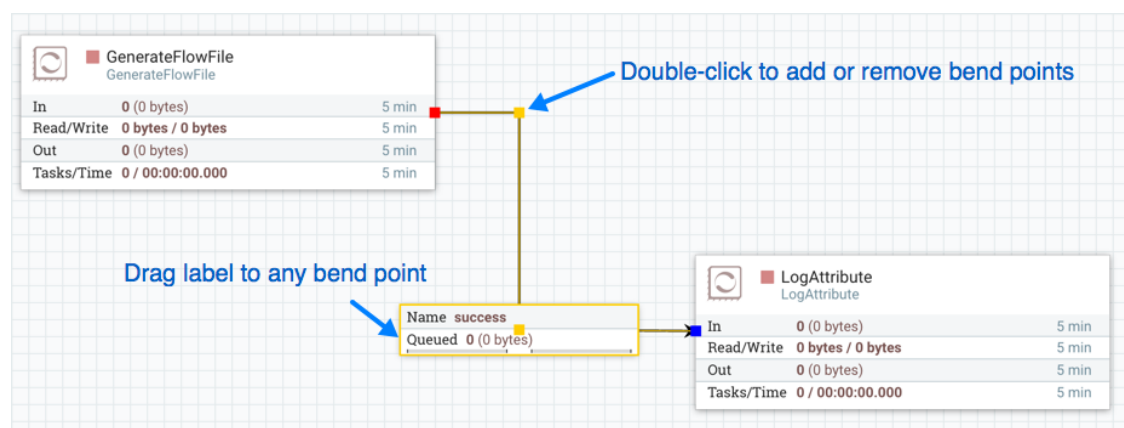


The following options are available:

- **Configure:** This option allows the user to change the configuration of the connection.
- **Status History:** This option opens a graphical representation of the connection's statistical information over time.
- **Bring to front:** This option brings the connection to the front of the canvas if something else (such as another connection) is overlapping it.
- **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.
- **List queue:** This option lists the queue of FlowFiles that may be waiting to be processed.
- **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.

1.7.8.3. 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.




1.7.9. 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:

 GenerateFlowFile GenerateFlowFile		
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

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:

 GenerateFlowFile GenerateFlowFile		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	37,308 / 00:00:01.371	5 min

1.7.10. 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 of the remote NiFi instance, 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 [Security Configuration](#) section of the *System Administrator's Guide*.
- 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

1.7.10.1. 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 of the remote NiFi instance (for more information on the components of a Remote Process Group, see [Remote Process Group Transmission](#) section of this guide.) The URL is the same URL you would use to go to that instance's User Interface. At that 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.

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/

Transport Protocol ? Local Network Interface ?

HTTP Proxy Server Hostname ? HTTP Proxy Server Port ?

HTTP Proxy User ? HTTP Proxy Password ?

Communications Timeout ? Yield Duration ?

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.

1.7.10.2. 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.

If Site-to-Site is configured to run securely, you will need to manage the 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. Configuring the Output Port and managing the port's access policies will again allow the DFM to control how many concurrent tasks are allowed, as well as which users are authorized to pull data from the instance being configured.

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 Root Group Output Ports.

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*.

For information on how to configure access policies, see the [Access Policies](#) section of the *System Administrator's Guide*.

1.7.11. Example Dataflow

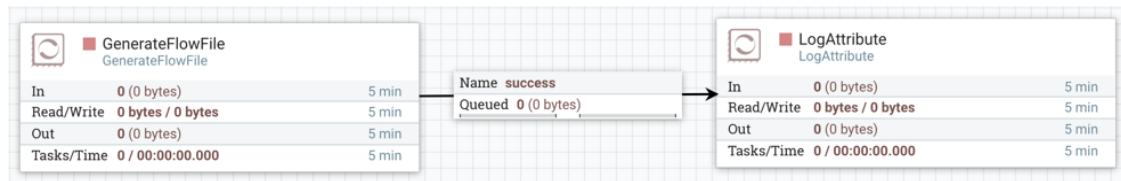
This section has described the steps required to build a dataflow. Now, to put it all together. The following example dataflow consists of just two processors: GenerateFlowFile and LogAttribute. These processors are normally used for testing, but they can also be used to build a quick flow for demonstration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

1.8. Command and Control of the DataFlow

When a component is added to the NiFi canvas, it is in the Stopped state. In order to cause the component to be triggered, the component must be started. Once started, the component can be stopped at any time. From a Stopped state, the component can be configured, started, or disabled.

1.8.1. Starting a Component

In order to start a component, the following conditions must be met:

- The component's configuration must be valid.
- All defined Relationships for the component must be connected to another component or auto-terminated.
- The component must be stopped.
- The component must be enabled.
- The component must have no active tasks. For more information about active tasks, see the "Anatomy of ..." sections under [Monitoring of DataFlow](#) ([Anatomy of a Processor](#), [Anatomy of a Process Group](#), [Anatomy of a Remote Process Group](#)).

Components can be started by selecting all of the components to start and then clicking the Start button (



) in the Operate Palette or by right-clicking a single component and choosing Start from the context menu.

If starting a Process Group, all components within that Process Group (including child Process Groups) will be started, with the exception of those components that are invalid or disabled.

Once started, the status indicator of a Processor will change to a Play symbol (



).

1.8.2. Stopping a Component

A component can be stopped any time that it is running. A component is stopped by right-clicking on the component and clicking Stop from the context menu, or by selecting the component and clicking the Stop button (



) in the Operate Palette.

If a Process Group is stopped, all of the components within the Process Group (including child Process Groups) will be stopped.

Once stopped, the status indicator of a component will change to the Stop symbol (



).

Stopping a component does not interrupt its currently running tasks. Rather, it stops scheduling new tasks to be performed. The number of active tasks is shown in the top-right corner of the Processor (See [Anatomy of a Processor](#) for more information).

1.8.3. Enabling/Disabling a Component

When a component is enabled, it is able to be started. Users may choose to disable components when they are part of a dataflow that is still being assembled, for example. Typically, if a component is not intended to be run, the component is disabled, rather than being left in the Stopped state. This helps to distinguish between components that are intentionally not running and those that may have been stopped temporarily (for instance, to change the component's configuration) and inadvertently were never restarted.

When it is desirable to re-enable a component, it can be enabled by selecting the component and clicking the Enable button (



) in the Operate Palette. This is available only when the selected component or components are disabled. Alternatively, a component can be enabled by checking the checkbox next to the "Enabled" option in the Settings tab of the Processor configuration dialog or the configuration dialog for a Port.

Once enabled, the component's status indicator will change to either Invalid (



) or Stopped (



), depending on whether or not the component is valid.

A component is then disabled by selecting the component and clicking the Disable button (



) in the Operate Palette, or by clearing the checkbox next to the "Enabled" option in the Settings tab of the Processor configuration dialog or the configuration dialog for a Port.

Only Ports and Processors can be enabled and disabled.

1.8.4. Remote Process Group Transmission

Remote Process Groups provide a mechanism for sending data to or retrieving data from a remote instance of NiFi. When a Remote Process Group (RPG) is added to the canvas, it is added with the Transmission Disabled, as indicated by the icon (



) in the top-left corner. When Transmission is Disabled, it can be enabled by right-clicking on the RPG and clicking the "Enable Transmission" menu item. This will cause all ports for which there is a Connection to begin transmitting data. This will cause the status indicator to then change to the Transmission Enabled icon (



).

If there are problems communicating with the Remote Process Group, a Warning indicator (



) may instead be present in the top-left corner. Hovering over this Warning indicator with the mouse will provide more information about the problem.

1.8.4.1. Individual Port Transmission

There are times when the DFM may want to either enable or disable transmission for only a specific Port within the Remote Process Group. This can be accomplished by right-clicking on the Remote Process Group and choosing the "Remote ports" menu item. This provides a configuration dialog from which each Port can be configured:

Remote Process Group Ports

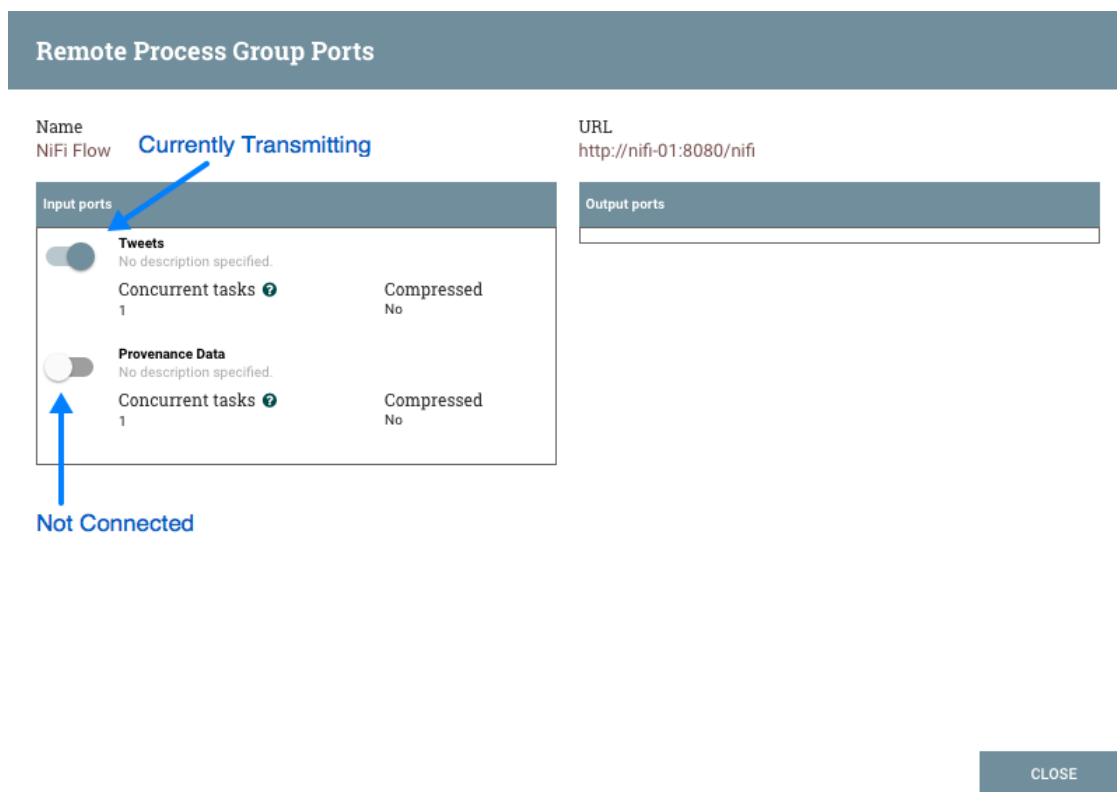
<p>Name NiFi Flow</p>	<p>URL http://nifi-01:8080/nifi</p>
<div style="border: 1px solid #ccc; padding: 5px;"> <p>Input ports</p> <div style="margin-bottom: 10px;"> <p><input checked="" type="checkbox"/> Tweets No description specified.</p> <p>Concurrent tasks ? 1</p> <p style="text-align: right;">Compressed No</p> </div> <div> <p><input type="checkbox"/> Provenance Data No description specified.</p> <p>Concurrent tasks ? 1</p> <p style="text-align: right;">Compressed No</p> </div> </div>	<div style="border: 1px solid #ccc; padding: 5px; height: 30px;"> <p>Output ports</p> </div>

CLOSE

The left-hand side lists all of the Input Ports that the remote instance of NiFi allows data to be sent to. The right-hand side lists all of the Output Ports from which this instance is able to pull data. If the remote instance is using secure communications (the URL of the NiFi instance begins with `https://`, rather than `http://`), any Ports that the remote instance has not made available to this instance will not be shown.

If a Port that is expected to be shown is not shown in this dialog, ensure that the instance has proper permissions and that the Remote Process Group's flow is current. This can be checked by closing the Port Configuration Dialog and looking at the bottom-right corner of the Remote Process Group. The date at which the flow was last refreshed is shown. If the flow appears to be outdated, it can be updated by right-clicking on the Remote Process Group and selecting "Refresh flow." (See [Anatomy of a Remote Process Group](#) for more information).

Each Port is shown with the Port name, followed by its description, currently configured number of Concurrent tasks, and whether or not data sent to this port will be compressed. To the left of this information is a switch to turn the Port on or off. Those Ports that have no Connections attached to them are grayed out:



The on/off switch provides a mechanism to enable and disable transmission for each Port in the Remote Process Group independently. Those Ports that are connected but are not currently transmitting can be configured by clicking the pencil icon (



) below the on/off switch. Clicking this icon will allow the DFM to change the number of

Concurrent tasks and whether or not compression should be used when transmitting data to or from this Port.

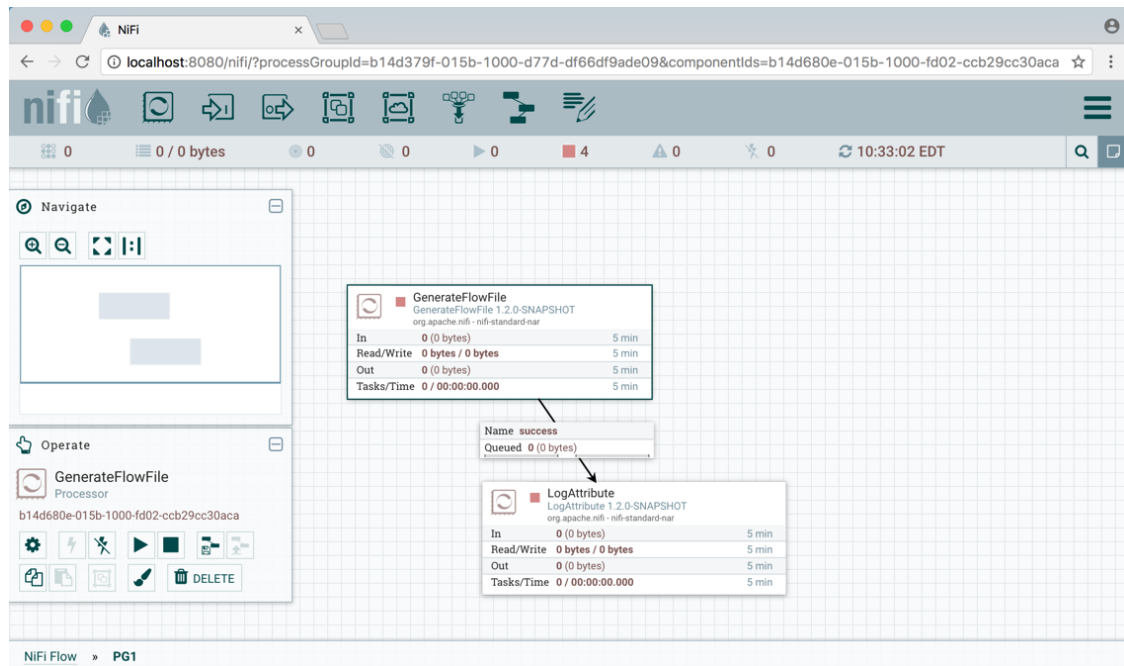
1.9. Navigating within a DataFlow

NiFi provides various mechanisms for getting around a dataflow. The [NiFi User Interface](#) section describes various ways to navigate around the NiFi canvas; however, once a flow exists on the canvas, there are additional ways to get from one component to another. When multiple Process Groups exist in a flow, breadcrumbs appear at the bottom of the screen, providing a way to navigate between them. In addition, to enter a Process Group that is currently visible on the canvas, simply double-click it, thereby "drilling down" into it. Connections also provide a way to jump from one location to another within the flow. Right-click on a connection and select "Go to source" or "Go to destination" in order to jump to one end of the connection or another. This can be very useful in large, complex dataflows, where the connection lines may be long and span large areas of the canvas. Finally, all components provide the ability to jump forward or backward within the flow. Right-click any component (e.g., a processor, process group, port, etc.) and select either "Upstream connections" or "Downstream connections". A dialog window will open, showing the available upstream or downstream connections that the user may jump to. This can be especially useful when trying to follow a dataflow in a backward direction. It is typically easy to follow the path of a dataflow from start to finish, drilling down into nested process groups; however, it can be more difficult to follow the dataflow in the other direction.

1.9.1. Component Linking

A hyperlink can be used to navigate directly to a component on the NiFi canvas. This is especially useful when [Multi-Tenant Authorization](#) is configured. For example, a URL can be given to a user to direct them to the specific process group to which they have privileges.

The default URL for a NiFi instance is `http://<hostname>:8080/nifi`, which points to the root process group. When a component is selected on the canvas, the URL is updated with the component's process group id and component id in the form `http://<hostname>:8080/nifi/?processGroupId=<UUID>&componentIds=<UUIDs>`. In the following screenshot, the `GenerateFlowFile` processor in the process group `PG1` is the selected component:



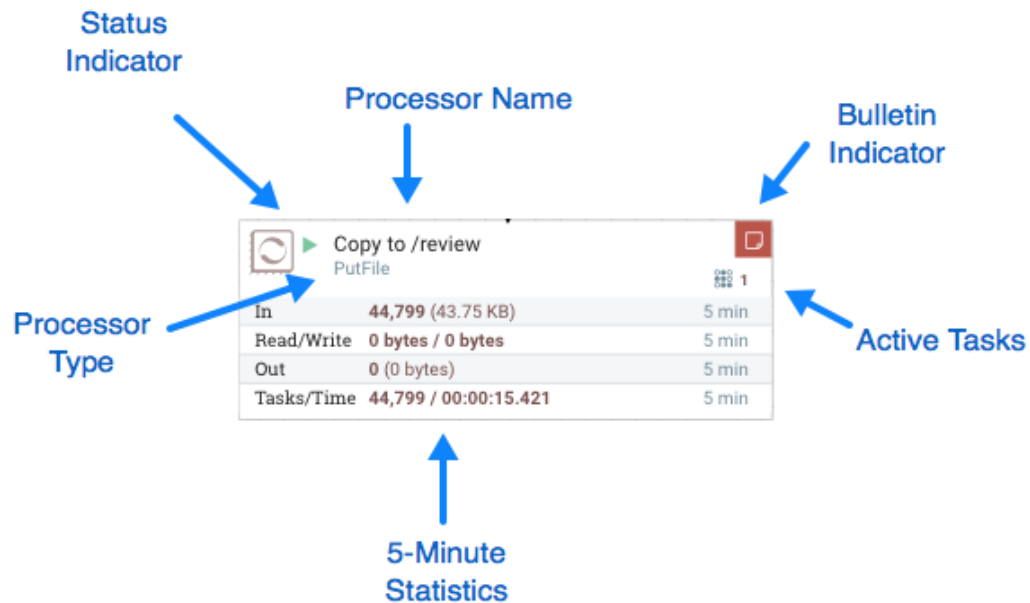
Linking to multiple components on the canvas is supported, with the restriction that the length of the URL cannot exceed a 2000 character limit.

1.10. Monitoring of DataFlow



NiFi provides a great deal of information about the DataFlow in order to monitor its health and status. The Status bar provides information about the overall system health (see [NiFi User Interface](#)). Processors, Process Groups, and Remote Process Groups provide fine-grained details about their operations. Connections and Process Groups provide information about the amount of data in their queues. The Summary Page provides information about all of the components on the canvas in a tabular format and also provides System Diagnostics that include disk usage, CPU utilization, and Java Heap and Garbage Collection information. In a clustered environment, this information is available per-node or as aggregates across the entire cluster. We will explore each of these monitoring artifacts below.

1.10.1. Anatomy of a Processor

NiFi provides a significant amount of information about each Processor on the canvas. The following diagram shows the anatomy of a Processor:



The image outlines the following elements:

- Processor Type:** NiFi provides several different types of Processors in order to allow for a wide range of tasks to be performed. Each type of Processor is designed to perform one specific task. The Processor type (PutFile, in this example) describes the task that this Processor performs. In this case, the Processor writes a FlowFile to disk - or "Puts" a FlowFile to a File.
- Bulletin Indicator:** When a Processor logs that some event has occurred, it generates a Bulletin to notify those who are monitoring NiFi via the User Interface. The DFM is able to configure which bulletins should be displayed in the User Interface by updating the "Bulletin level" field in the "Settings" tab of the Processor configuration dialog. The default value is `WARN`, which means that only warnings and errors will be displayed in the UI. This icon is not present unless a Bulletin exists for this Processor. When it is present, hovering over the icon with the mouse will provide a tooltip explaining the message provided by the Processor as well as the Bulletin level. If the instance of NiFi is clustered, it will also show the Node that emitted the Bulletin. Bulletins automatically expire after five minutes.
- Status Indicator:** Shows the current Status of the Processor. The following indicators are possible:
 -  **Running:**
The Processor is currently running.
 -  **Stopped:**
The Processor is valid and enabled but is not running.



The Processor is enabled but is not currently valid and cannot be started. Hovering over this icon will provide a tooltip indicating why the Processor is not valid.

Invalid:



The Processor is not running and cannot be started until it has been enabled. This status does not indicate whether or not the Processor is valid.

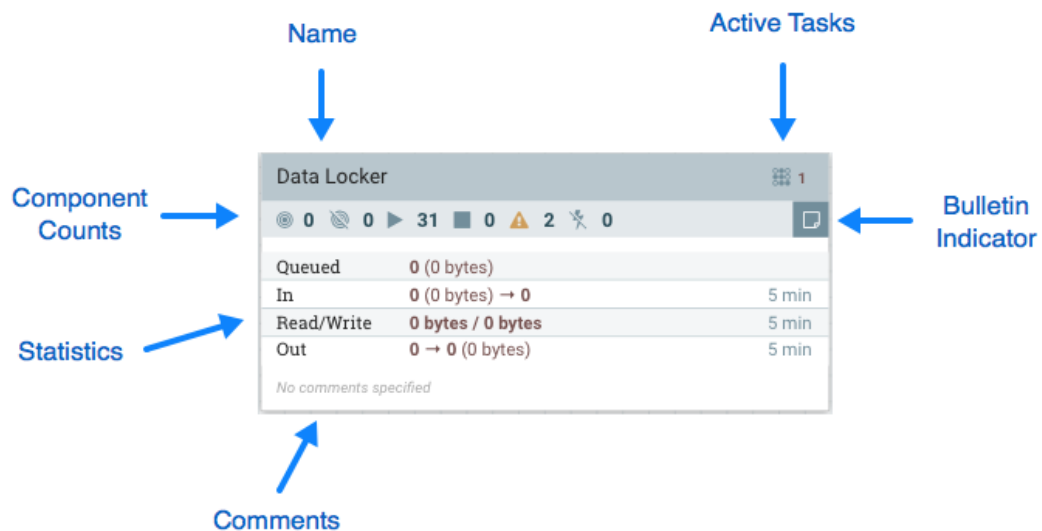
Disabled:

- **Processor Name:** This is the user-defined name of the Processor. By default, the name of the Processor is the same as the Processor Type. In the example, this value is "Copy to / review".
- **Active Tasks:** The number of tasks that this Processor is currently executing. This number is constrained by the "Concurrent tasks" setting in the "Scheduling" tab of the Processor configuration dialog. Here, we can see that the Processor is currently performing two tasks. If the NiFi instance is clustered, this value represents the number of tasks that are currently executing across all nodes in the cluster.
- **5-Minute Statistics:** The Processor shows several different statistics in tabular form. Each of these statistics represents the amount of work that has been performed in the past five minutes. If the NiFi instance is clustered, these values indicate how much work has been done by all of the Nodes combined in the past five minutes. These metrics are:
 - **In:** The amount of data that the Processor has pulled from the queues of its incoming Connections. This value is represented as <count> / <size> where <count> is the number of FlowFiles that have been pulled from the queues and <size> is the total size of those FlowFiles' content. In this example, the Processor has pulled 884 FlowFiles from the input queues, for a total of 8.85 megabytes (MB).
 - **Read/Write:** The total size of the FlowFile content that the Processor has read from disk and written to disk. This provides valuable information about the I/O performance that this Processor requires. Some Processors may only read the data without writing anything while some will not read the data but will only write data. Others will neither read nor write data, and some Processors will both read and write data. In this example, we see that in the past five minutes, this Processor has read 4.7 MB of the FlowFile content and has written 4.7 MB as well. This is what we would expect, since this Processor simply copies the contents of a FlowFile to disk. Note, however, that this is not the same as the amount of data that it pulled from its input queues. This is because some of the files that it pulled from the input queues already exist in the output directory, and the Processor is configured to route FlowFiles to failure when this occurs. Therefore, for those files which already existed in the output directory, data was neither read nor written to disk.
 - **Out:** The amount of data that the Processor has transferred to its outbound Connections. This does not include FlowFiles that the Processor removes itself, or FlowFiles that are routed to connections that are auto-terminated. Like the "In" metric above, this value is represented as <count> / <size> where <count> is the number of FlowFiles that have been transferred to outbound Connections and <size> is the total size of those FlowFiles' content. In this example, all of the Relationships are configured to be auto-terminated, so no FlowFiles are reported as having been transferred Out.

- **Tasks/Time:** The number of times that this Processor has been triggered to run in the past 5 minutes, and the amount of time taken to perform those tasks. The format of the time is <hour>:<minute>:<second>. Note that the amount of time taken can exceed five minutes, because many tasks can be executed in parallel. For instance, if the Processor is scheduled to run with 60 Concurrent tasks, and each of those tasks takes one second to complete, it is possible that all 60 tasks will be completed in a single second. However, in this case we will see the Time metric showing that it took 60 seconds, instead of 1 second. This time can be thought of as "System Time," or said another way, this value is 60 seconds because that's the amount of time it would have taken to perform the action if only a single concurrent task were used.



1.10.2. Anatomy of a Process Group

The Process Group provides a mechanism for grouping components together into a logical construct in order to organize the DataFlow in a way that makes it more understandable from a higher level. The following image highlights the different elements that make up the anatomy of a Process Group:







The Process Group consists of the following elements:

- **Name:** This is the user-defined name of the Process Group. This name is set when the Process Group is added to the canvas. The name can later be changed by right-clicking on the Process Group and clicking the "Configure" menu option. In this example, the name of the Process Group is "Process Group ABC."
- **Bulletin Indicator:** When a child component of a Process Group emits a bulletin, that bulletin is propagated to the component's parent Process Group, as well. When any component has an active Bulletin, this indicator will appear, allowing the user to hover over the icon with the mouse to see the Bulletin.

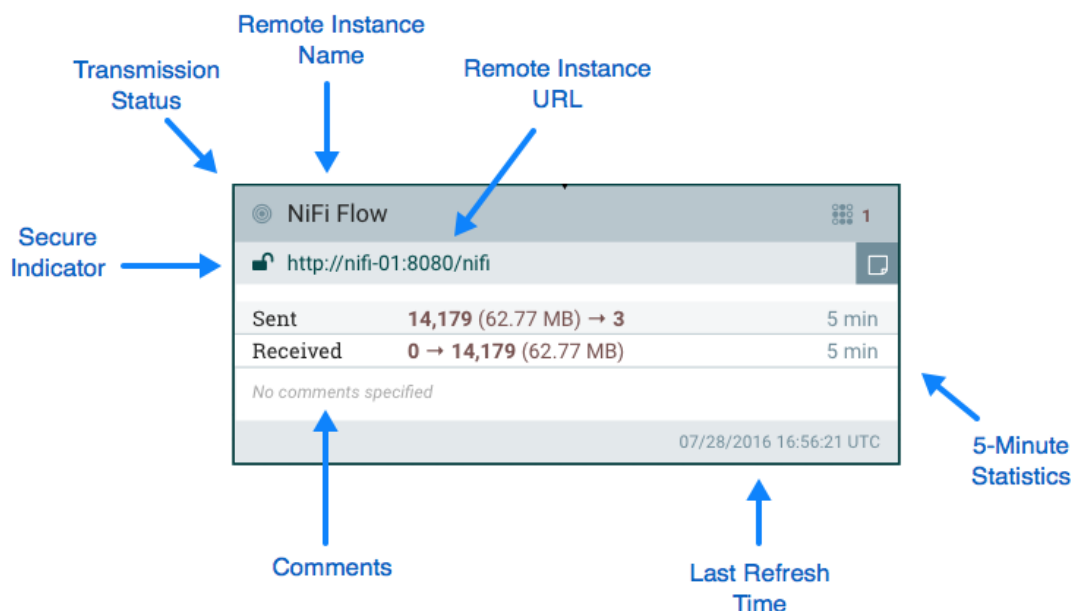
- **Active Tasks:** The number of tasks that are currently executing by the components within this Process Group. Here, we can see that the Process Group is currently performing one task. If the NiFi instance is clustered, this value represents the number of tasks that are currently executing across all nodes in the cluster.
- **Comments:** When the Process Group is added to the canvas, the user is given the option of specifying Comments in order to provide information about the Process Group. The comments can later be changed by right-clicking on the Process Group and clicking the "Configure" menu option. In this example, the Comments are set to "Example Process Group."
- **Statistics:** Process Groups provide statistics about the amount of data that has been processed by the Process Group in the past 5 minutes as well as the amount of data currently enqueued within the Process Group. The following elements comprise the "Statistics" portion of a Process Group:
 - **Queued:** The number of FlowFiles currently enqueued within the Process Group. This field is represented as <count> / <size> where <count> is the number of FlowFiles that are currently enqueued in the Process Group and <size> is the total size of those FlowFiles' content. In this example, the Process Group currently has 1,738 FlowFiles enqueued; those FlowFiles have a total size of 350.03 megabytes (MB).
 - **In:** The number of FlowFiles that have been transferred into the Process Group through all of its Input Ports over the past 5 minutes. This field is represented as <count> / <size> where <count> is the number of FlowFiles that have entered the Process Group in the past 5 minutes and <size> is the total size of those FlowFiles' content. In this example, 686 FlowFiles have entered the Process Group and their total size is 214.01 MB.
 - **Read/Write:** The total size of the FlowFile content that the components within the Process Group have read from disk and written to disk. This provides valuable information about the I/O performance that this Process Group requires. In this example, we see that in the past five minutes, components within this Process Group have read 72.9 MB of the FlowFile content and have written 686.65 MB.
 - **Out:** The number of FlowFiles that have been transferred out of the Process Group through its Output Ports over the past 5 minutes. This field is represented as <count> / <size> where <count> is the number of FlowFiles that have exited the Process Group in the past 5 minutes and <size> is the total size of those FlowFiles' content. In this example, 657 FlowFiles have exited the Process Group and their total size is 477.74 MB.
- **Component Counts:** The Component Counts element provides information about how many components of each type exist within the Process Group. The following provides information about each of these icons and their meanings:
 -  **Transmitting Ports:** The number of Remote Process Group Ports that currently are configured to transmit data to remote instances of NiFi or pull data from remote instances of NiFi. **Transmitting**
 -  **Non-Transmitting Ports:** The number of Remote Process Group Ports that are currently **Non-**

connected to components within this Process Group but currently have their transmission disabled.





-  **Running**
Components: The number of Processors, Input Ports, and Output Ports that are currently running within this Process Group.
-  **Stopped**
Components: The number of Processors, Input Ports, and Output Ports that are currently not running but are valid and enabled. These components are ready to be started.
-  **Invalid**
Components: The number of Processors, Input Ports, and Output Ports that are enabled but are currently not in a valid state. This may be due to misconfigured properties or missing Relationships.
-  **Disabled**
Components: The number of Processors, Input Ports, and Output Ports that are currently disabled. These components may or may not be valid. If the Process Group is started, these components will not cause any errors but will not be started.

1.10.3. Anatomy of a Remote Process Group

When creating a DataFlow, it is often necessary to transfer data from one instance of NiFi to another. In this case, the remote instance of NiFi can be thought of as a Process Group. For this reason, NiFi provides the concept of a Remote Process Group. From the User Interface, the Remote Process Group looks similar to the Process Group. However, rather than showing information about the inner workings and state of a Remote Process Group, such as queue sizes, the information rendered about a Remote Process Group is related to the interaction that occurs between this instance of NiFi and the remote instance.



The image above shows the different elements that make up a Remote Process Group. Here, we provide an explanation of the icons and details about the information provided.

- Transmission Status:** The Transmission Status indicates whether or not data Transmission between this instance of NiFi and the remote instance is currently enabled. The icon shown will be the Transmission Enabled icon () if any of the Input Ports or Output Ports is currently configured to transmit or the Transmission Disabled icon () if all of the Input Ports and Output Ports that are currently connected are stopped.
- Remote Instance Name:** This is the name of the NiFi instance that was reported by the remote instance. When the Remote Process Group is first created, before this information has been obtained, the URL of the remote instance will be shown here instead.
- Remote Instance URL:** This is the URL of the remote instance that the Remote Process Group points to. This URL is entered when the Remote Process Group is added to the canvas and it cannot be changed.
- Secure Indicator:** This icon indicates whether or not communications with the remote NiFi instance are secure. If communications with the remote instance are secure, this will be indicated by the "locked" icon (). If the communications are not secure, this will be indicated by the "unlocked" icon (). If the communications are secure, this instance of NiFi will not be able to communicate

with the remote instance until an administrator for the remote instance grants access. Whenever the Remote Process Group is added to the canvas, this will automatically initiate a request to have a user for this instance of NiFi created on the remote instance. This instance will be unable to communicate with the remote instance until an administrator on the remote instance adds the user to the system and adds the "NiFi" role to the user. In the event that communications are not secure, the Remote Process Group is able to receive data from anyone, and the data is not encrypted while it is transferred between instances of NiFi.

- **5-Minute Statistics:** Two statistics are shown for Remote Process Groups: **Sent** and **Received**. Both of these are in the format <count> / <size> where <count> is the number of FlowFiles that have been sent or received in the previous five minutes and <size> is the total size of those FlowFiles' content.
- **Comments:** The Comments that are provided for a Remote Process Group are not comments added by the users of this NiFi but rather the Comments added by the administrators of the remote instance. These comments indicate the purpose of the NiFi instance as a whole.
- **Last Refreshed Time:** The information that is pulled from a remote instance and rendered on the Remote Process Group in the User Interface is periodically refreshed in the background. This element indicates the time at which that refresh last happened, or if the information has not been refreshed for a significant amount of time, the value will change to indicate *Remote flow not current*. NiFi can be triggered to initiate a refresh of this information by right-clicking on the Remote Process Group and choosing the "Refresh flow" menu item.

1.10.4. Queue Interaction

The FlowFiles enqueued in a Connection can be viewed when necessary. The Queue listing is opened via `List queue` in a Connection's context menu. The listing will return the top 100 FlowFiles in the active queue according to the configured priority. The listing can be performed even if the source and destination are actively running.

Additionally, details for a Flowfile in the listing can be viewed by clicking on the Details icon (



) in the left most column. From here, the FlowFile details and attributes are available as well buttons for downloading or viewing the content. Viewing the content is only available if the `nifi.content.viewer.url` has been configured. If the source or destination of the Connection are actively running, there is a chance that the desired FlowFile will no longer be in the active queue.

The FlowFiles enqueued in a Connection can also be deleted when necessary. The removal of the FlowFiles is initiated via `Empty queue` in the Connection's context menu. This action can also be performed if the source and destination are actively running.

1.10.5. Summary Page

While the NiFi canvas is useful for understanding how the configured DataFlow is laid out, this view is not always optimal when trying to discern the status of the system. In order to

help the user understand how the DataFlow is functioning at a higher level, NiFi provides a Summary page. This page is available in the Global Menu in the top-right corner of the User Interface. See the [NiFi User Interface](#) section for more information about the location of this toolbar.

The Summary Page is opened by selecting Summary from the Global Menu. This opens the Summary table dialog:

NiFi Summary

PROCESSORS | INPUT PORTS | OUTPUT PORTS | REMOTE PROCESS GROUPS | CONNECTIONS | PROCESS GROUPS

Displaying 60 of 60

Filter: by name View: Single node Cluster

Name	Type	Run Status	In / Size 5 min	Read / Write 5 min	Out / Size 5 min	Tasks / Time 5 min
Base64EncodeCo...	Base64EncodeCo...	Invalid	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Capture Network ...	ExecuteProcess	Running (5)	0 (0 bytes)	0 bytes / 245.97 MB	2,883 (245.97 MB)	2,911 / 00:24:59.988
Check if Dataset ...	IdentifyMimeType	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Decompress Data...	CompressContent	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Detect Interesting...	RouteOnAttribute	Running (2)	11,741,700 (13 GB)	0 bytes / 0 bytes	0 (0 bytes)	11,741,700 / 00:09:...
Empty Contents	ReplaceText	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Evaluate.JsonPath	Evaluate.JsonPath	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Evaluate.JsonPath	Evaluate.JsonPath	Running (4)	11,742,510 (13 GB)	13 GB / 0 bytes	11,742,510 (13 GB)	11,742,856 / 00:20:...
Extract File ID	UpdateAttribute	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Extract File ID	UpdateAttribute	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
ExtractText	ExtractText	Running	1,498,914 (244.45 ...)	244.45 MB / 0 bytes	1,496,995 (244.25 ...)	1,500,692 / 00:02:4...

Last updated: 17:45:24 UTC system diagnostics

This dialog provides a great deal of information about each of the components on the canvas. Below, we have annotated the different elements within the dialog in order to make the discussion of the dialog easier.

NiFi Summary

PROCESSORS | INPUT PORTS | OUTPUT PORTS | REMOTE PROCESS GROUPS | CONNECTIONS | PROCESS GROUPS ← Component Tabs

Displaying 60 of 60

Filter: by name View: Single node Cluster

Name	Type	Run Status	In / Size 5 min	Read / Write 5 min	Out / Size 5 min	Tasks / Time 5 min
Base64EncodeCo...	Base64EncodeCo...	Invalid	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Capture Network ...	ExecuteProcess	Running (5)	0 (0 bytes)	0 bytes / 245.97 MB	2,883 (245.97 MB)	2,911 / 00:24:59.988
Check if Dataset ...	IdentifyMimeType	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Decompress Data...	CompressContent	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Detect Interesting...	RouteOnAttribute	Running (2)	11,741,700 (13 GB)	0 bytes / 0 bytes	0 (0 bytes)	11,741,700 / 00:09:...
Empty Contents	ReplaceText	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Evaluate.JsonPath	Evaluate.JsonPath	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Evaluate.JsonPath	Evaluate.JsonPath	Running (4)	11,742,510 (13 GB)	13 GB / 0 bytes	11,742,510 (13 GB)	11,742,856 / 00:20:...
Extract File ID	UpdateAttribute	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
Extract File ID	UpdateAttribute	Running	0 (0 bytes)	0 bytes / 0 bytes	0 (0 bytes)	0 / 00:00:00.000
ExtractText	ExtractText	Running	1,498,914 (244.45 ...)	244.45 MB / 0 bytes	1,496,995 (244.25 ...)	1,500,692 / 00:02:4...

Last updated: 17:45:24 UTC system diagnostics

The Summary page is primarily comprised of a table that provides information about each of the components on the canvas. Above this table is a set of five tabs that can be used to

view the different types of components. The information provided in the table is the same information that is provided for each component on the canvas. Each of the columns in the table may be sorted by clicking on the heading of the column. For more on the types of information displayed, see the sections [Anatomy of a Processor](#), [Anatomy of a Process Group](#), and [Anatomy of a Remote Process Group](#) above.

The Summary page also includes the following elements:

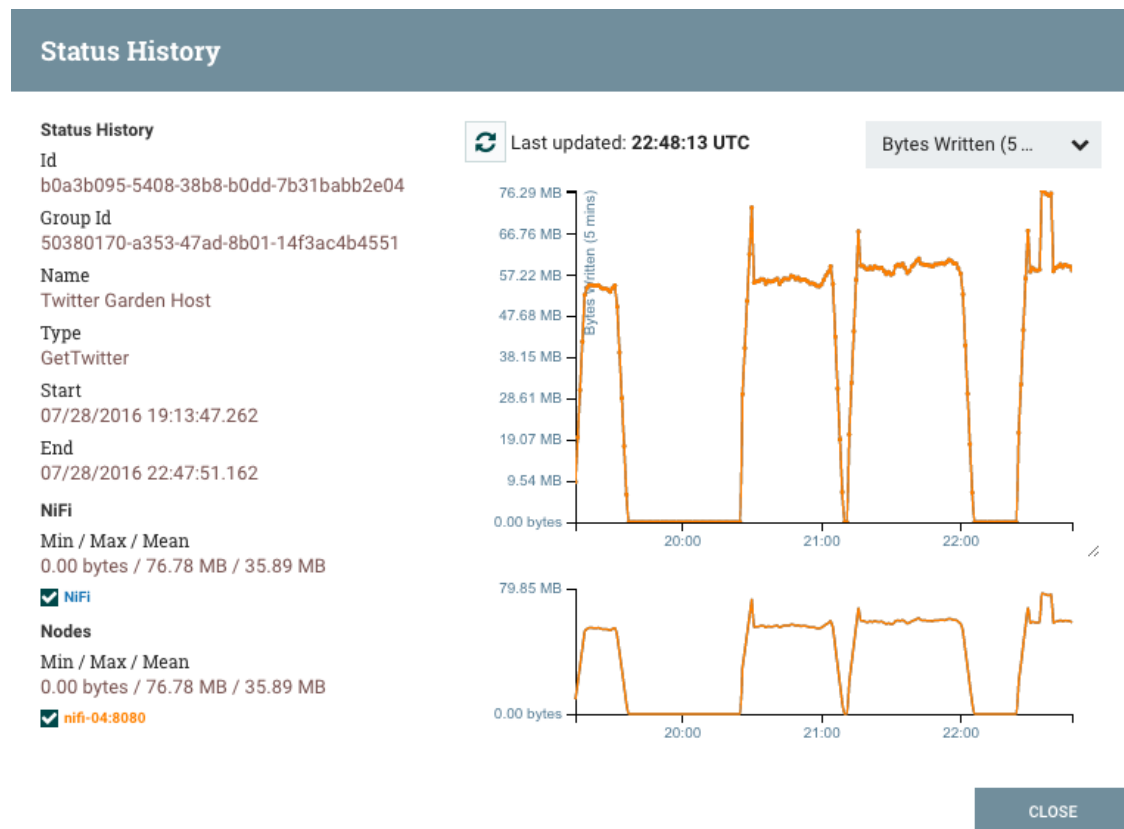
- **Bulletin Indicator:** As in other places throughout the User Interface, when this icon is present, hovering over the icon will provide information about the Bulletin that was generated, including the message, the severity level, the time at which the Bulletin was generated, and (in a clustered environment) the node that generated the Bulletin. Like all the columns in the Summary table, this column where bulletins are shown may be sorted by clicking on the heading so that all the currently existing bulletins are shown at the top of the list.
- **Details:** Clicking the Details icon will provide the user with the details of the component. This dialog is the same as the dialog provided when the user right-clicks on the component and chooses the "View configuration" menu item.
- **Go To:** Clicking this button will close the Summary page and take the user directly to the component on the NiFi canvas. This may change the Process Group that the user is currently in. This icon is not available if the Summary page has been opened in a new browser tab or window (by clicking the "Pop Out" button, as described below).
- **Status History:** Clicking the Status History icon will open a new dialog that shows a historical view of the statistics that are rendered for this component. See the section [Historical Statistics of a Component](#) for more information.
- **Refresh:** The Refresh button allows the user to refresh the information displayed without closing the dialog and opening it again. The time at which the information was last refreshed is shown just to the right of the Refresh button. The information on the page is not automatically refreshed.
- **Filter:** The Filter element allows users to filter the contents of the Summary table by typing in all or part of some criteria, such as a Processor Type or Processor Name. The types of filters available differ according to the selected tab. For instance, if viewing the Processor tab, the user is able to filter by name or by type. When viewing the Connections tab, the user is able to filter by source, by name, or by destination name. The filter is automatically applied when the contents of the text box are changed. Below the text box is an indicator of how many entries in the table match the filter and how many entries exist in the table.
- **Pop-Out:** When monitoring a flow, it is helpful to be able to open the Summary table in a separate browser tab or window. The Pop-Out button, next to the Close button, will cause the entire Summary dialog to be opened in a new browser tab or window (depending on the configuration of the browser). Once the page is "popped out", the dialog is closed in the original browser tab/window. In the new tab/window, the Pop-Out button and the Go-To button will no longer be available.
- **System Diagnostics:** The System Diagnostics window provides information about how the system is performing with respect to system resource utilization. While this is intended mostly for administrators, it is provided in this view because it does provide a

summary of the system. This dialog shows information such as CPU utilization, how full the disks are, and Java-specific metrics, such as memory size and utilization, as well as Garbage Collection information.

1.10.6. Historical Statistics of a Component

While the Summary table and the canvas show numeric statistics pertaining to the performance of a component over the past five minutes, it is often useful to have a view of historical statistics as well. This information is available by right-clicking on a component and choosing the "Status History" menu option or by clicking on the Status History in the Summary page (see [Summary Page](#) for more information).

The amount of historical information that is stored is configurable in the NiFi properties but defaults to 24 hours. For specific configuration information reference the Component Status Repository of the [System Administrator's Guide](#). When the Status History dialog is opened, it provides a graph of historical statistics:



The left-hand side of the dialog provides information about the component that the stats are for, as well as a textual representation of the statistics being graphed. The following information is provided on the left-hand side:

- **Id:** The ID of the component for which the stats are being shown.
- **Group Id:** The ID of the Process Group in which the component resides.
- **Name:** The Name of the Component for which the stats are being shown.

- **Component-Specific Entries:** Information is shown for each different type of component. For example, for a Processor, the type of Processor is displayed. For a Connection, the source and destination names and IDs are shown.
- **Start:** The earliest time shown on the graph.
- **End:** The latest time shown on the graph.
- **Min/Max/Mean:** The minimum, maximum, and mean (arithmetic mean, or average) values are shown. These values are based only on the range of time selected, if any time range is selected. If this instance of NiFi is clustered, these values are shown for the cluster as a whole, as well as each individual node. In a clustered environment, each node is shown in a different color. This also serves as the graph's legend, showing the color of each node that is shown in the graph. Hovering the mouse over the Cluster or one of the nodes in the legend will also make the corresponding node bold in the graph.

The right-hand side of the dialog provides a drop-down list of the different types of metrics to render in the graphs below. The top graph is larger so as to provide an easier-to-read rendering of the information. In the bottom-right corner of this graph is a small handle (



) that can be dragged to resize the graph. The blank areas of the dialog can also be dragged around to move the entire dialog.

The bottom graph is much shorter and provides the ability to select a time range. Selecting a time range here will cause the top graph to show only the time range selected, but in a more detailed manner. Additionally, this will cause the Min/Max/Mean values on the left-hand side to be recalculated. Once a selection has been created by dragging a rectangle over the graph, double-clicking on the selected portion will cause the selection to fully expand in the vertical direction (i.e., it will select all values in this time range). Clicking on the bottom graph without dragging will remove the selection.

1.11. Templates

DFMs have the ability to build very large and complex DataFlows using NiFi. This is achieved by using the basic components: Processor, Funnel, Input/Output Port, Process Group, and Remote Process Group. These can be thought of as the most basic building blocks for constructing a DataFlow. At times, though, using these small building blocks can become tedious if the same logic needs to be repeated several times.

To solve this issue, NiFi provides the concept of a Template. A Template is a way of combining these basic building blocks into larger building blocks. Once a DataFlow has been created, parts of it can be formed into a Template. This Template can then be dragged onto the canvas, or can be exported as an XML file and shared with others. Templates received from others can then be imported into an instance of NiFi and dragged onto the canvas.

1.11.1. Creating a Template

To create a Template, select the components that are to be a part of the template, and then click the "Create Template" (



) button in the Operate Palette (See [NiFi User Interface](#) for more information on the Operate Palette).

Clicking this button without selecting anything will create a Template that contains all of the contents of the current Process Group. This means that creating a Template with nothing selected while on the Root Process Group will create a single Template that contains the entire flow.

After clicking this button, the user is prompted to provide a name and an optional description for the template. Each template must have a unique name. After entering the name and optional description, clicking the Create button will generate the template and notify the user that the template was successfully created, or provide an appropriate error message if unable to create the template for some reason.

It is important to note that if any Processor that is Templated has a sensitive property (such as a password), the value of that sensitive property is not included in the Template. As a result, when dragging the Template onto the canvas, newly created Processors may not be valid if they are missing values for their sensitive properties. Additionally, any Connection that was selected when making the Template is not included in the Template if either the source or the destination of the Connection is not also included in the Template.

1.11.2. Importing a Template

After receiving a Template that has been exported from another NiFi, the first step needed to use the template is to import the template into this instance of NiFi. You may import templates into any Process Group where you have the appropriate authorization.

From the Operate Palette, click the "Upload Template" (



) button (See [NiFi User Interface](#) for more information on the Operate Palette). This will display the Upload Template dialog. Click the find icon and use the File Selection dialog to choose which template file to upload. Select the file and click Open. Clicking the "Upload" button will attempt to import the Template into this instance of NiFi. The Upload Template dialog will update to show "Success" or an error message if there was a problem importing the template.

1.11.3. Instantiating a Template

Once a Template has been created (see [Creating a Template](#)) or imported (see [Importing a Template](#)), it is ready to be instantiated, or added to the canvas. This is accomplished by dragging the Template icon (



) from the Components Toolbar (see [NiFi User Interface](#)) onto the canvas.

This will present a dialog to choose which Template to add to the canvas. After choosing the Template to add, simply click the "Add" button. The Template will be added to the canvas with the upper-left-hand side of the Template being placed wherever the user dropped the Template icon.

This leaves the contents of the newly instantiated Template selected. If there was a mistake, and this Template is no longer wanted, it may be deleted.

1.11.4. Managing Templates

One of the most powerful features of NiFi Templates is the ability to easily export a Template to an XML file and to import a Template that has already been exported. This provides a very simple mechanism for sharing parts of a DataFlow with others. You can select Templates from the Global Menu (see [NiFi User Interface](#)) to open a dialog that displays all of the Templates that are currently available, filter the templates to see only those of interest, export, and delete Templates.

1.11.4.1. Exporting a Template

Once a Template has been created, it can be shared with others in the Template Management page (see [Managing Templates](#)). To export a Template, locate the Template in the table. The Filter in the top-right corner can be used to help find the appropriate Template if several are available. Then click the Export or Download button (



). This will download the template as an XML file to your computer. This XML file can then be sent to others and imported into other instances of NiFi (see [Importing a Template](#)).

1.11.4.2. Removing a Template

Once it is decided that a Template is no longer needed, it can be easily removed from the Template Management page (see [Managing Templates](#)). To delete a Template, locate it in the table (the Filter in the top-right corner may be used to find the appropriate Template if several are available) and click the Delete button (



). This will prompt for confirmation. After confirming the deletion, the Template will be removed from this table and will no longer be available to add to the canvas.

1.12. Data Provenance

While monitoring a dataflow, users often need a way to determine what happened to a particular data object (FlowFile). NiFi's Data Provenance page provides that information. Because NiFi records and indexes data provenance details as objects flow through the system, users may perform searches, conduct troubleshooting and evaluate things like dataflow compliance and optimization in real time. By default, NiFi updates this information every five minutes, but that is configurable.

To access the Data Provenance page, select Data Provenance from the Global Menu. Clicking this button opens a dialog window that allows the user to see the most recent

Data Provenance information available, search the information for specific items, and filter the search results. It is also possible to open additional dialog windows to see event details, replay data at any point within the dataflow, and see a graphical representation of the data's lineage, or path through the flow. (These features are described in depth below.)

NiFi Data Provenance

Displaying 1,000 of 1,000
Oldest event available: 07/14/2016 20:56:52 UTC
Showing the most recent 1,000 of 1,000+ events, please refine the search.

Filter: by component name

Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type	Node
07/29/2016 00:14:...	ATTRIBUTES_MODL...	91ae19fa-5797-45...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	bcc29546-bbd0-43...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	9f4c3b69-6cef-40a...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	38bb2021-1f07-4e...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	f31d3aa0-40b7-46...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	7d12c959-6952-41...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	93f31b5c-be89-49e...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	a1e5a6a4-b44e-4e...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	cf2095c8-052a-47...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	c0db8381-6c13-42...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	9da3e06d-9715-46...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...
07/29/2016 00:14:...	ATTRIBUTES_MODL...	18247e64-41b3-4f...	1.11 KB	EvaluateJsonPath	EvaluateJsonPath	nifi-05.eng.hortonw...

Last updated: 00:14:35 UTC

1.12.1. Provenance Events

Each point in a dataflow where a FlowFile is processed in some way is considered a *provenance event*. Various types of provenance events occur, depending on the dataflow design. For example, when data is brought into the flow, a RECEIVE event occurs, and when data is sent out of the flow, a SEND event occurs. Other types of processing events may occur, such as if the data is cloned (CLONE event), routed (ROUTE event), modified (CONTENT_MODIFIED or ATTRIBUTES_MODIFIED event), split (FORK event), combined with other data objects (JOIN event), and ultimately removed from the flow (DROP event).

The provenance event types are:

Provenance Event	Description
ADDINFO	Indicates a provenance event when additional information such as a new linkage to a new URI or UUID is added
ATTRIBUTES_MODIFIED	Indicates that a FlowFile's attributes were modified in some way
CLONE	Indicates that a FlowFile is an exact duplicate of its parent FlowFile
CONTENT_MODIFIED	Indicates that a FlowFile's content was modified in some way
CREATE	Indicates that a FlowFile was generated from data that was not received from a remote system or external process
DOWNLOAD	Indicates that the contents of a FlowFile were downloaded by a user or external entity
DROP	Indicates a provenance event for the conclusion of an object's life for some reason other than object expiration

Provenance Event	Description
EXPIRE	Indicates a provenance event for the conclusion of an object's life due to the object not being processed in a timely manner
FETCH	Indicates that the contents of a FlowFile were overwritten using the contents of some external resource
FORK	Indicates that one or more FlowFiles were derived from a parent FlowFile
JOIN	Indicates that a single FlowFile is derived from joining together multiple parent FlowFiles
RECEIVE	Indicates a provenance event for receiving data from an external process
REPLAY	Indicates a provenance event for replaying a FlowFile
ROUTE	Indicates that a FlowFile was routed to a specified relationship and provides information about why the FlowFile was routed to this relationship
SEND	Indicates a provenance event for sending data to an external process
UNKNOWN	Indicates that the type of provenance event is unknown because the user who is attempting to access the event is not authorized to know the type


1.12.2. Searching for Events


One of the most common tasks performed in the Data Provenance page is a search for a given FlowFile to determine what happened to it. To do this, click the `Search` button in the upper-right corner of the Data Provenance page. This opens a dialog window with parameters that the user can define for the search. The parameters include the processing event of interest, distinguishing characteristics about the FlowFile or the component that produced the event, the timeframe within which to search, and the size of the FlowFile.


Search Events


Fields


Event Type	<input type="text"/>
FlowFile UUID	<input type="text"/>
Filename	<input type="text"/>
Component ID	<input type="text"/>
Relationship	<input type="text"/>
twitter.msg	<input type="text"/>
language	<input type="text"/>


Start date 


Start time (UTC) 

End date 

End time (UTC) 

Minimum file size 

Maximum file size 

Search location
 

For example, to determine if a particular FlowFile was received, search for an Event Type of "RECEIVE" and include an identifier for the FlowFile, such as its uuid or filename. The asterisk (*) may be used as a wildcard for any number of characters. So, to determine whether a FlowFile with "ABC" anywhere in its filename was received at any time on Jan. 6, 2015, the search shown in the following image could be performed:

Search Events

Fields		
Event Type	RECEIVE	
FlowFile UUID		
Filename	*ABC* <input style="width: 50px;" type="text"/>	
Component ID		
Relationship		
twitter.msg		
language		
Start date	Start time (UTC)	
<input type="text" value="07/28/2016"/>	<input type="text" value="00:00:00"/>	
End date	End time (UTC)	
<input type="text" value="07/28/2016"/>	<input type="text" value="23:59:59"/>	
Minimum file size	Maximum file size	
<input type="text"/>	<input type="text"/>	
Search location	<input style="width: 100%;" type="text" value="cluster"/>	
	<input type="button" value="CANCEL"/> <input style="background-color: #4a7c8c; color: white;" type="button" value="SEARCH"/>	

1.12.3. Details of an Event

In the far-left column of the Data Provenance page, there is a View Details icon for each event

().
 Clicking this button opens a dialog window with three tabs: Details, Attributes, and Content.

Provenance Event

DETAILS	ATTRIBUTES	CONTENT
Time 07/29/2016 00:58:44.829 UTC		Parent FlowFiles (0) No parents
Event Duration No value set		Child FlowFiles (0) No children
Lineage Duration 00:00:00.203		
Type ATTRIBUTES_MODIFIED		
FlowFile Uuid 62d2161f-0b2a-4b2a-a552-ab617bef3811		
File Size 1.1 KB		
Component Id 7bba4f68-2861-3a12-aac6-60f12e11e215		
Component Name EvaluateJsonPath		
Component Type ...		

OK

The Details tab shows various details about the event, such as when it occurred, what type of event it was, and the component that produced the event. The information that is displayed will vary according to the event type. This tab also shows information about the FlowFile that was processed. In addition to the FlowFile's UUID, which is displayed on the left side of the Details tab, the UUIDs of any parent or children FlowFiles that are related to that FlowFile are displayed on the right side of the Details tab.

The Attributes tab shows the attributes that exist on the FlowFile as of that point in the flow. In order to see only the attributes that were modified as a result of the processing event, the user may select the checkbox next to "Only show modified" in the upper-right corner of the Attributes tab.

Provenance Event

DETAILS ATTRIBUTES CONTENT

Attribute Values Show modified attributes only

eventType
ATTRIBUTES_MODIFIED
No value previously set

filename
6320498487869637

newSize
1119
No value previously set

oldSize
1119
No value previously set

path
./

reporting.task.transaction.id
fc9fad99-89f0-4978-a3aa-571bb8b8851b

uuid
62d2161f-0b2a-4b2a-a552-ab617bef3811

OK

1.12.4. Replaying a FlowFile

A DFM may need to inspect a FlowFile's content at some point in the dataflow to ensure that it is being processed as expected. And if it is not being processed properly, the DFM may need to make adjustments to the dataflow and replay the FlowFile again. The Content tab of the View Details dialog window is where the DFM can do these things. The Content tab shows information about the FlowFile's content, such as its location in the Content Repository and its size. In addition, it is here that the user may click the `Download` button to download a copy of the FlowFile's content as it existed at this point in the flow. The user may also click the `Submit` button to replay the FlowFile at this point in the flow. Upon clicking `Submit`, the FlowFile is sent to the connection feeding the component that produced this processing event.

Provenance Event

DETAILS
ATTRIBUTES
CONTENT

Input Claim



Container
default

Section
918

Identifier
1469753924663-275350

Offset
108834

Size
1.1 KB

 DOWNLOAD
 VIEW

Output Claim



Container
default

Section
918

Identifier
1469753924663-275350

Offset
108834

Size
1.1 KB

 DOWNLOAD
 VIEW

Replay

Connection Id
88970033-a406-33a2-b679-711d04de4a35

OK

1.12.5. Viewing FlowFile Lineage

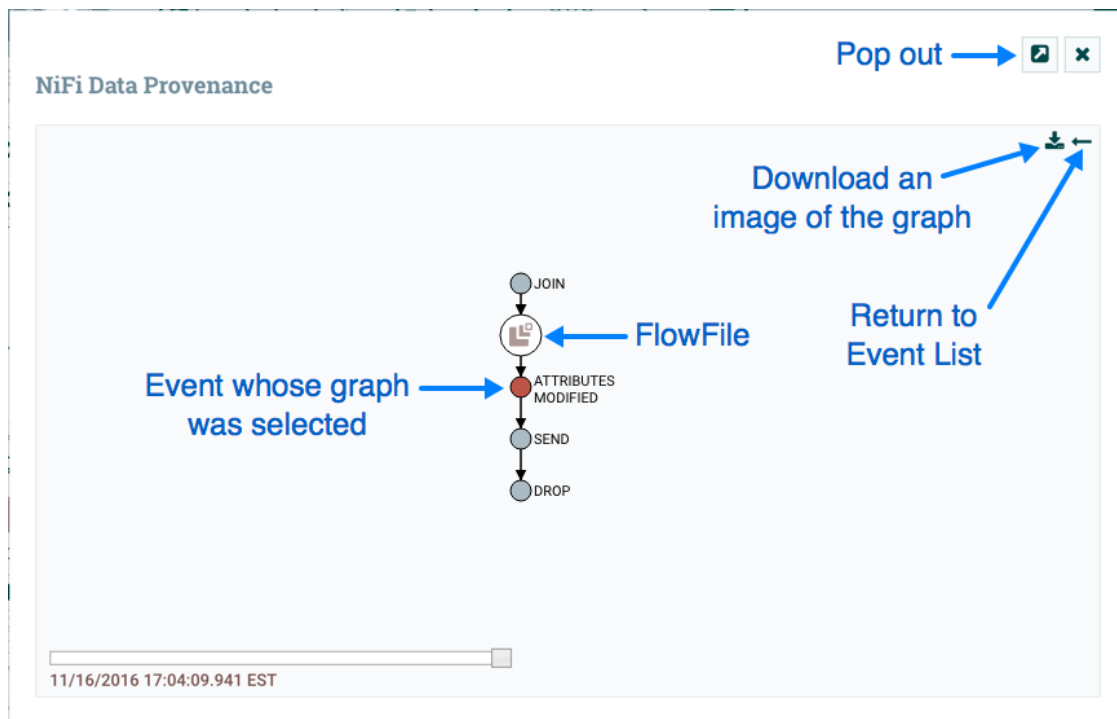
It is often useful to see a graphical representation of the lineage or path a FlowFile took within the dataflow. To see a FlowFile's lineage, click on the "Show Lineage" icon (



) in the far-right column of the Data Provenance table. This opens a graph displaying the FlowFile (

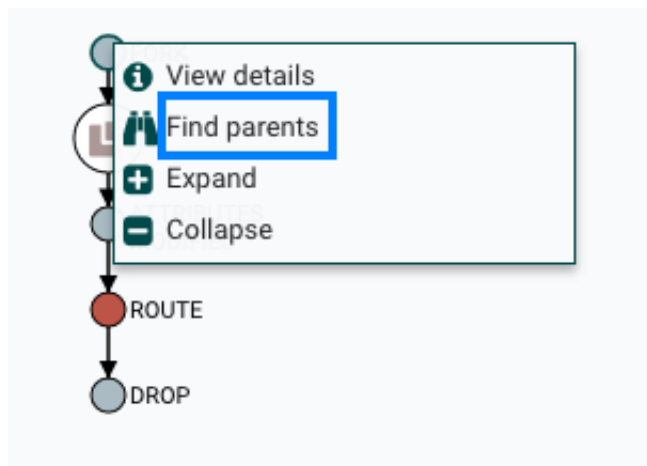


) and the various processing events that have occurred. The selected event will be highlighted in red. It is possible to right-click on any event to see that event's details (see [Details of an Event](#)). To see how the lineage evolved over time, click the slider at the bottom-left of the window and move it to the left to see the state of the lineage at earlier stages in the dataflow.

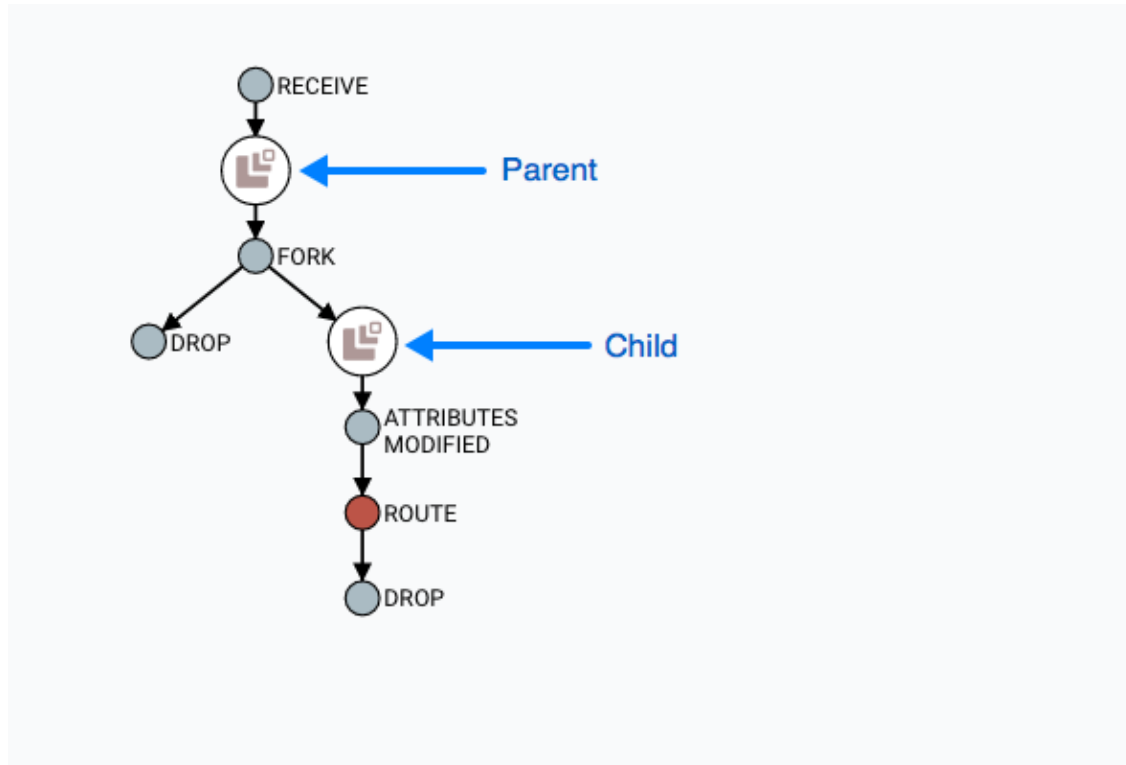


1.12.5.1. Find Parents

Sometimes, a user may need to track down the original FlowFile that another FlowFile was spawned from. For example, when a FORK or CLONE event occurs, NiFi keeps track of the parent FlowFile that produced other FlowFiles, and it is possible to find that parent FlowFile in the Lineage. Right-click on the event in the lineage graph and select "Find parents" from the context menu.

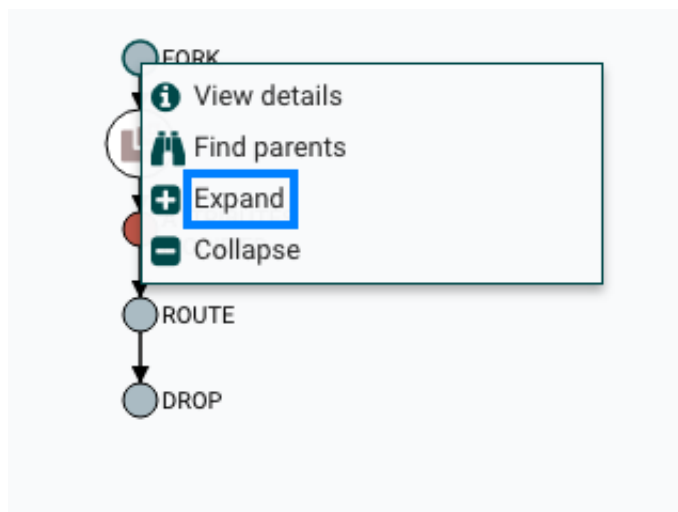


Once "Find parents" is selected, the graph is re-drawn to show the parent FlowFile and its lineage as well as the child and its lineage.

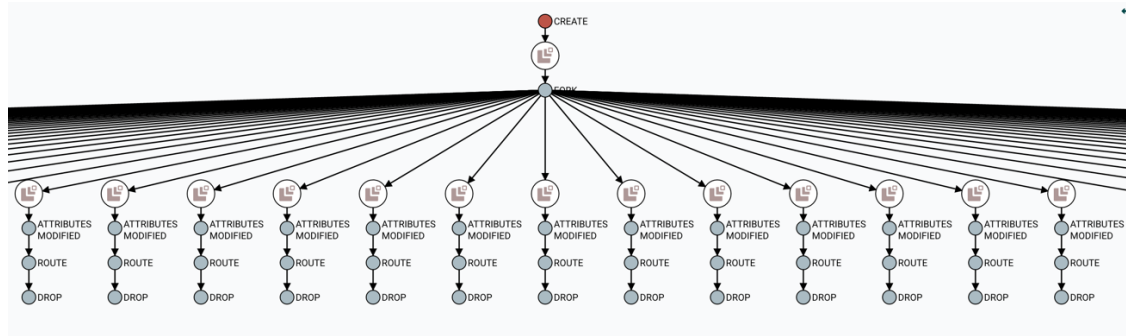


1.12.5.2. Expanding an Event

In the same way that it is useful to find a parent FlowFile, the user may also want to determine what children were spawned from a given FlowFile. To do this, right-click on the event in the lineage graph and select "Expand" from the context menu.



Once "Expand" is selected, the graph is re-drawn to show the children and their lineage.



1.12.6. Encrypted Provenance Repository

While OS-level access control can offer some security over the provenance data written to the disk in a repository, there are scenarios where the data may be sensitive, compliance and regulatory requirements exist, or NiFi is running on hardware not under the direct control of the organization (cloud, etc.). In this case, the provenance repository allows for all data to be encrypted before being persisted to the disk.

Performance The current implementation of the encrypted provenance repository intercepts the record writer and reader of `WriteAheadProvenanceRepository`, which offers significant performance improvements over the legacy `PersistentProvenanceRepository` and uses the AES/GCM algorithm, which is fairly performant on commodity hardware. In most scenarios, the added cost will not be significant (unnoticeable on a flow with hundreds of provenance events per second, moderately noticeable on a flow with thousands - tens of thousands of events per second). However, administrators should perform their own risk assessment and performance analysis and decide how to move forward. Switching back and forth between encrypted/unencrypted implementations is not recommended at this time.

1.12.6.1. What is it?

The `EncryptedWriteAheadProvenanceRepository` is a new implementation of the provenance repository which encrypts all event record information before it is written to the repository. This allows for storage on systems where OS-level access controls are not sufficient to protect the data while still allowing querying and access to the data through the NiFi UI/API.

1.12.6.2. How does it work?

The `WriteAheadProvenanceRepository` was introduced in NiFi 1.2.0 and provided a refactored and much faster provenance repository implementation than the previous `PersistentProvenanceRepository`. The encrypted version wraps that implementation with a record writer and reader which encrypt and decrypt the serialized bytes respectively.

The fully qualified class

`org.apache.nifi.provenance.EncryptedWriteAheadProvenanceRepository` is specified as the provenance repository implementation in `nifi.properties` as

the value of `nifi.provenance.repository.implementation`. In addition, [new properties](#) must be populated to allow successful initialization.

1.12.6.2.1. StaticKeyProvider

The `StaticKeyProvider` implementation defines keys directly in `nifi.properties`. Individual keys are provided in hexadecimal encoding. The keys can also be encrypted like any other sensitive property in `nifi.properties` using the `./encrypt-config.sh` tool in the NiFi Toolkit.

The following configuration section would result in a key provider with two available keys, "Key1" (active) and "AnotherKey".

```
nifi.provenance.repository.encryption.key.provider.implementation=org.apache.nifi.provenance.StaticKeyProvider
nifi.provenance.repository.encryption.key.id=Key1
nifi.provenance.repository.encryption.key=0123456789ABCDEFFEDCBA98765432100123456789ABCDEFFEDCBA9876543210
nifi.provenance.repository.encryption.key.id.AnotherKey=0101010101010101010101010101010101010101010101010101010101010101
```

1.12.6.2.2. FileBasedKeyProvider

The `FileBasedKeyProvider` implementation reads from an encrypted definition file of the format:

```
key1=NGCpDpxBZNN0DBodz0p1SDbTjC2FG5kp1pCmdUKJlxxtcMSo6GC4fMlTyy1mPeKOxzLut3DRX+51j6PCO5SznA==
key2=GYxPbMMDbnraXs09eGJudAM5jTvVYp05XtImkAg4JY4rIbmHOiVUUI6OeOf7ZW+hH42jtPgNW9pSkkQ9HWY/vQ==
key3=SFellxuz7J89Y/IQ7YbJPOL0/YKZRFL/VUxJgEHxxlXpd/8ELA7wwN59K1KTr3BURCcFP5YGmwrSKfr4OE4Vlg==
key4=kZprfcTSTH69UuOU3jMkZfrtiVR/eqWmmbdku3bQcUJ/+UToecNB5lzOVEMBChyEXppyXXC35Wa6GEXFK6PMKw==
key5=c6FzfnKm7UR7xqI2NFpZ+fEKBfSU7+1NvRw+XWQ9U39MONWqk5gvoyOCdFR1kUgeg46jrn5dGXk13sRqE0GETQ==
```

Each line defines a key ID and then the Base64-encoded cipher text of a 16 byte IV and wrapped AES-128, AES-192, or AES-256 key depending on the JCE policies available. The individual keys are wrapped by AES/GCM encryption using the **master key** defined by `nifi.bootstrap.sensitive.key` in `conf/bootstrap.conf`.

1.12.6.2.3. Key Rotation

Simply update `nifi.properties` to reference a new key ID in `nifi.provenance.repository.encryption.key.id`. Previously-encrypted events can still be decrypted as long as that key is still available in the key definition file or `nifi.provenance.repository.encryption.key.id.<OldKeyID>` as the key ID is serialized alongside the encrypted record.

1.12.6.3. Writing and Reading Event Records

Once the repository is initialized, all provenance event record write operations are serialized according to the configured schema

writer (EventIdFirstSchemaRecordWriter by default for WriteAheadProvenanceRepository) to a byte[]. Those bytes are then encrypted using an implementation of ProvenanceEventEncryptor (the only current implementation is AES/GCM/NoPadding) and the encryption metadata (keyId, algorithm, version, IV) is serialized and prepended. The complete byte[] is then written to the repository on disk as normal.

```

107720 F00082AB 00CC5918 EEE8BA0C B8009A83 25E00000 00010000 02C0401AC ED0000573
10752 72002D6F 72672E61 70616368 652E6E69 66692E70 726F7665 6E616E63 652E456E
10784 63727970 74696F6E 4D657461 64617461 F1CD9BC1 C9611FED 02000549 00106369
10816 70686572 42797465 4C656E67 74684C00 09616C67 6F726974 686D7400 124C6A61
10848 76612F6C 616E672F 53747269 6E673B58 00076976 42797465 73740002 5B424C00
10880 056B6579 49647100 7E00014C 00077665 7273696F 6E71007E 00017870 00000177
10912 74001141 45532F47 434D2F4E 6F506164 64696E67 75720002 5B42ACF3 17F80608
10944 54E00200 00787000 00001064 0D1180B4 B6D0C396 6550344A 64B0A374 00044B65
10976 79317400 02763138 B71255E8 107038EF 822BE655 FB187773 ABC47419 A16EC6CD
11008 C6D9E40B 1A493ACB C11DD677 CD8030C4 3EB1E5FF 99A96D7F C4E894E3 51955A6C
11040 38CB18EA 09E928B5 3FEF2343 6F9B1CC5 B86F964C 9ECD947E A0B9FB97 A8075329
11072 74384728 DC2207BA GC38C84C 024F10D0 C7666E9D 6CE3D0DB F448EE4E 4A34A557
11104 665CCA9B 4F7614C3 535D9053 1989EA6D 7936B277 F0515548 363E47DF 7CED90B8
11136 961049E7 DFD554FF 870EA4C0 B41C7A4D CD11CAE7 EEE3D875 ED3849F8 E972CFE2
11168 B392A6B9 1B3221F2 23AE5B89 5459BFC4 D30F9B19 576263BB 2A77EF73 E00F08AE
11200 695F4237 028291DC D2644890 09481B0B 5A07C441 D093B6DC A0DB116B 3CB1FACE
11232 78AD5171 21384968 B17D0C68 32E7F967 AC0E69FE 7C538338 3B97B1DE 58C588FE
11264 A83C6C0F 0356A5DF 03D3DB18 2D3725AD 57C75573 F61384E2 4D1A4569 69B38BA6
11296 27952E97 3FA21FF7 CACCS18C 9F6E7C94 E276DB11 89B771A8 A3870B53 DE618B02
11328 E2FDD786 926CEBB2 E1011759 3D580AAC 751CD631 85C79451 C0F6B01E 418A0000
11360 00020000 028B01AC ED000573 72002D6F 72672E61 70616368 652E6E69 66692E70
11392 726F7665 6E616E63 652E456E 63727970 74696F6E 4D657461 64617461 F1CD9BC1
11424 C9611FED 02000549 00106369 70686572 42797465 4C656E67 74684C00 09616C67
11456 6F726974 686D7400 124C6A61 76612F6C 616E672F 53747269 6E673B58 00076976
11488 42797465 73740002 5B424C00 056B6579 49647100 7E00014C 00077665 7273696F
11520 6E71007E 00017870 0000019E 74001141 45532F47 434D2F4E 6F506164 64696E67
11552 75720002 5R42ACF3 17F80608 54F00700 00787000 0000108C 317CFCD0 9115E0DF
  
```

On record read, the process is reversed. The encryption metadata is parsed and used to decrypt the serialized bytes, which are then deserialized into a ProvenanceEventRecord object. The delegation to the normal schema record writer/reader allows for "random-access" (i.e. immediate seek without decryption of unnecessary records).

Within the NiFi UI/API, there is no detectable difference between an encrypted and unencrypted provenance repository. The Provenance Query operations work as expected with no change to the process.

1.12.6.4. Potential Issues

Switching Implementations When switching between implementation "families" (i.e. VolatileProvenanceRepository or PersistentProvenanceRepository to EncryptedWriteAheadProvenanceRepository), the existing repository must be cleared from the file system before starting NiFi. A terminal command like `localhost:$NIFI_HOME $ rm -rf provenance_repository/` is sufficient.

- Switching between unencrypted and encrypted repositories
- If a user has an existing repository (WriteAheadProvenanceRepository only - **not** PersistentProvenanceRepository) that is not encrypted and switches

their configuration to use an encrypted repository, the application writes an error to the log but starts up. However, previous events are not accessible through the provenance query interface and new events will overwrite the existing events. The same behavior occurs if a user switches from an encrypted repository to an unencrypted repository. Automatic roll-over is a future effort ([NIFI-3722](#)) but NiFi is not intended for long-term storage of provenance events so the impact should be minimal. There are two scenarios for roll-over:

- Encrypted # unencrypted - if the previous repository implementation was encrypted, these events should be handled seamlessly as long as the key provider available still has the keys used to encrypt the events (see [Key Rotation](#))
- Unencrypted # encrypted - if the previous repository implementation was unencrypted, these events should be handled seamlessly as the previously recorded events simply need to be read with a plaintext schema record reader and then written back with the encrypted record writer
- There is also a future effort to provide a standalone tool in NiFi Toolkit to encrypt/decrypt an existing provenance repository to make the transition easier. The translation process could take a long time depending on the size of the existing repository, and being able to perform this task outside of application startup would be valuable ([NIFI-3723](#)).
- Multiple repositories - No additional effort or testing has been applied to multiple repositories at this time. It is possible/likely issues will occur with repositories on different physical devices. There is no option to provide a heterogeneous environment (i.e. one encrypted, one plaintext repository).
- Corruption - when a disk is filled or corrupted, there have been reported issues with the repository becoming corrupted and recovery steps are necessary. This is likely to continue to be an issue with the encrypted repository, although still limited in scope to individual records (i.e. an entire repository file won't be irrecoverable due to the encryption).

1.13. Other Management Features

In addition to the Summary Page, Data Provenance Page, Template Management Page, and Bulletin Board Page, there are other tools in the Global Menu (see [NiFi User Interface](#)) that are useful to the DFM. Select Flow Configuration History to view all the changes that have been made to the dataflow. The history can aid in troubleshooting, such as if a recent change to the dataflow has caused a problem and needs to be fixed. The DFM can see what changes have been made and adjust the flow as needed to fix the problem. While NiFi does not have an "undo" feature, the DFM can make new changes to the dataflow that will fix the problem.

Two other tools in the Global Menu are Controller Settings and Users. The Controller Settings page provides the ability to change the name of the NiFi instance, add comments describing the NiFi instance, and set the maximum number of threads that are available to the application. It also provides tabs where DFMs may add and configure [Controller Services](#) and [Reporting Tasks](#). The Users page is used to manage user access, which is described in the [System Administrator's Guide](#).