FAQs

Please post questions/answers under the following categories. If you don't see an appropriate category for your addition, please add one.

General Application/Framework

Q: Where can I find information about upgrading to a new NiFi version?

- A: The Upgrading NiFi guide provides steps for upgrading NiFi along with suggested NiFi installation configurations that make upgrading even easier. We also offer Migration Guidance. Additionally you can find details on upgrading in the System Properties section of the Administrator Guide.

Q: Where can I find information about the REST API?

- A: The REST API documentation is included in the "help" documentation within the application and also on our web site here. To get to the documentation within the application, click on the "help" link in the upper-right corner of the NiFi user interface. Then, in the pane on the left-hand side, scroll down to the very bottom, where you will see a Developer section, with links to the Developer Guide and the REST API documentation.

Q: What is the base endpoint for the NiFi REST API?

- A: The base endpoint is http://[server]:8080/nifi-api for default settings. If you adjust the conf/nifi.properties, then these values may differ.

Q: How do I select multiple items on the graph at the same time, such as if I want to select and move a group of processors?

- A: You can either select one item, and then hold down the Shift key and select other items, or you can click anywhere on the blank graph, outside what you want to select, then hold down the shift key and drag a selection box around what you want to select.

Q: How do I set up NiFi to run behind a proxy?

- A: There are a couple of key items to know when doing this.
  1) NiFi is comprised of a number of web applications (web UI, web API, documentation, custom UI's, etc). So, you'll need to set up your mapping to the root path. That way, all context paths are passed through accordingly. For instance, if you only mapped the /nifi context path, the custom UI for the UpdateAttribute processor will not work, since it's available at /update-attribute-ui-<version>.
  2) NiFi's REST API will generate URIs for each component on the graph. Since you are coming through a proxy, you'll need to override certain elements of the URIs being generated. You can override the elements of the URI by adding the following HTTP headers when your proxy generat es the HTTP request to the NiFi instance:
  X-ProxyScheme - the scheme to use to connect to your proxy (https in this case)
  X-ProxyHost - the host of your proxy
  X-ProxyPort - the port your proxy is listening on
  X-ProxyContextPath - the path you've configured to map to the NiFi instance

Q: Am I correct in assuming that I can transit large volumes of data through NiFi flows in and out of Hadoop?

- A: Yes, you are correct that large payloads can be moved through NiFi. As data moves through NiFi, a pointer to the data is being passed around, referred to as a FlowFile. The content of the FlowFile is only accessed as needed.
  The key for large payloads would be to operate on the payload in a streaming fashion so that you don't read too many large payloads into memory and exceed your JVM memory. As an example, a typical pattern for bringing data into HDFS from NiFi, is to use a MergeContent processor right before a PutHDFS processor. MergeContent can take many small/medium size files and merge them together to form an appropriate size file for HDFS. It does this by copying all of the input streams from the original files to a new output stream, and can, therefore, merge a large amount of files without exceeding the memory of the JVM.

Q: What happens to my data if there is a power loss and the system goes down?

- A: NiFi stores the data in the repository as it is traversing through the system. There are three key repositories: The FlowFile Repository, the Content Repository, and the Provenance Repository. As a Processor writes data to a flowfile, that is streamed directly to the content repository. When the processor finishes, it commits the session (essentially marks a transaction as complete). This triggers the Provenance Repository to be updated to include the events that occurred for that processor and then the FlowFile repository is then updated to keep track of where in the flow the FlowFile is. Finally, the FlowFile can be moved to the next queue in the flow. This way, if power is lost at any point, NiFi is able to resume where it left off.This, however, glosses over one detail, which is that by default when we update the repositories, we write the information to disk but this is often cached by the operating system. If you truly have a complete loss of power, it is possible to lose those updates to the repository. This can be avoided by configuring the repositories in the nifi.properties file to always sync to disk. This, however, can be a significant hinderance to performance. Simply killing NiFi, though, will not be problematic, as the operating system will still be responsible for flushing that data to the disk.

Q: How do I enable debug logging for a specific processor, rather than system-wide?

- A: The logging is configured in <NIFI_HOME>/conf/logback.xml. In that file, you can specify the fully qualified class name for the processor you want to enable specific logging for. For example:

  <logger name="org.apache.nifi.
  processors.standard.GetFile" level="DEBUG"/>

  This would enable DEBUG-level logging for every GetFile processor in your flow.

Q: I want to know how I can package and deploy the same dataflow from a development environment to a testing environment. Do I need to recreate the entire dataflow again in the different environment?

- A: The primary mechanism is **flow templates**. They do have some important limitations that you'll want to understand (at least as of version 0.4.0). First, some component properties are sensitive, like passwords, and thus are not included in the templates. So you'll have to reenter them

when you apply the template in the new environment. Second, there are at times properties that you'd want to have different values for in different environments. We need to provide an easy property/environment variable mapping mechanism. Although we intend to address this, it is not actively being worked on yet.

- It is also worth noting that the NiFi Expression Language (EL) can use system user environment variables as well as NiFI JVM properties as input. So you could create templates that have components configured with EL statements that reference these environment variables. That would allow each system running the same dataflow to pull in different values. You can add NiFI JVM properties in the bootstrap.conf file in your NiFi installation. System environment variables are set at the OS level. There is an order of preference to be aware of, however. NiFi will first search for FlowFile Attributes matching the defined subject/key name in the EL statement, then system environment variables, and then JVM properties. See the **Expression Language Guide** for more information.

Q: At what point is a piece of data considered under NiFi's control?

- A: NiFi is said to be under control of data once a FlowFile with its content has been generated in a ProcessSession and that session has been committed. See **this section of the Developer Guide** for a more detailed overview of this part of the process.

Q: How do I bend connections so that I can create nicer-looking dataflows?

- A: You can add a bend-point (elbow) at any place on a connection by double-clicking the connection at the desired point. Then, simply use the mouse to grab that point on the connection and drag it so that the connection is bent as you desire. You can remove a bend-point by double-clicking it again. You can also move the label on the connection to any bend-point on the connection.

Q: If no prioritizers are set in a processor, what prioritization scheme is used?

- A: The default prioritization scheme is said to be undefined, and it may change from time to time. If no prioritizers are set, the processor will sort the data based on the FlowFile's Content Claim. This way, it provides the most efficient reading of the data and the highest throughput. We have discussed changing the default setting to First In First Out, but right now it is based on what gives the best performance.

Processors

Q: ListenHTTP -

I have built most of our production flow in NiFi. Part of the data is received from thousands of distributed components via the ListenHTTP processor. I now need to add some automatic registration for these components (this assigns some ID to the component) and enable authentication using the generated ID. Is this possible to implement with the current version? The registration could happen outside (maybe using Play), but the file reception should happen in NiFi in order to initiate the flow.

- A: The ListenHTTP processor isn't so general purpose as to support custom designed authentication and authorization schemes. You could use it to restrict who can provide data through use of HTTPS and specific certificate DN patterns or to simply accept whatever data is brought in and then have a follow-on processor in the flow choose to drop data or not because it knows and trusts the source though. If you want to be able to block data as a client component tries to feed it to NiFi using a custom scheme, then you'll probably want to look at a custom processor or wire together the HandleHttpRequest and HandleHttpResponse processors. With those you can build a rather custom and powerful API/interface and have it support your own chosen authentication and authorization scheme on top of HTTP/S. With those processors if you understand HTTP well you can basically visually construct a web service. It requires the user to have a lot of knowledge to get the most of out it though.

Q: GetSFTP -

I'm setting up a GetSFTP processor to run every morning to download files expected in a regular location that is named based on the previous day's date, e.g. /logs/2015-09-08. I'm trying to set the GetSFTP processor "Remote Path". What NiFi expression language statement should I use?

- A: Try this expression in your Remote Path:
  /logs/${now():toNumber():minus(86400000):format('yyyy-MM-dd')}

Q: GetHTTP & InvokeHTTP -

The GetHTTP processor works fine with static filename when getting files from a website. However, I have a use case where I need to download a file daily and the filename is the date of today, ie: 09222015.zip. Since the URL property of the GetHTTP does not support expression language, I cannot do something like http://example.com/${now():format('MMddyyyy')}.zip. Is there a way I can specify the filename dynamically? Or using other processor to make this work. Please advise.

- A: InvokeHTTP will allow you to use the Expression Language to do an HTTP GET.
  It works a bit differently than a GetHTTP, though, because GetHTTP is a "Source Processor" whereas InvokeHTTP needs to be fed a FlowFile in order to do anything. So you can use GenerateFlowFile as a source and have it generate a 0 byte FlowFile (set the File Size property to "0 B"). Then just connect GenerateFlowFile to InvokeHTTP.
  Having said that, it is a bit awkward to have to use a GenerateFlowFile to trigger InvokeHTTP to run, so there is a ticket (NIFI-993) to change the GetHTTP to evaluate the Expression Language for the URL property. That should be fixed in NiFi version 0.4.0. In the meantime, though, GenerateFlowFile -> InvokeHTTP should provide you with the capability you're looking for.
- In addition, if you want to retain the filename of the file you are ingesting, you could use an UpdateAttribute processor before the InvokeHTTP. For example: GenerateFlowFile -> UpdateAttribute -> InvokeHTTP. In UpdateAttribute create a property called 'filename' and give it a value of: ${now():format('MMddyyyy')}.zip – Then, in the InvokeHTTP, use that for the Expression Language in the URL: http://example.com/${filename} ... Now you'll get your content and will have retained the filename you used to pull it.

Q: GetTwitter -

I am trying to use the GetTwitter processor to pull in tweets with a certain keyword. Do I need to pull in the tweets and then use the RouteOnAttribute processor?

- A: You should not need to use a RouteOnAttribute to pull out only messages that you are interested in. The GetTwitter processor has a property named "Twitter Endpoint." The default value is "Sample Endpoint." This endpoint just returns a sample of approximately 1% of the Twitter data

feed, without applying any sort of filters or anything. If you want to receive only tweets that have the word "foo" (for example), you should set the Twitter Endpoint to "Filter Endpoint." Then, in the "Terms to Filter On" property, set the value to "foo" and you should be good to go. If you want to only accept tweets that are in English, you can also set the "Languages" property to "EN". The key, though, is to ensure that you are using the Filter Endpoint; otherwise, these settings will be ignored. Should you wish to perform additional routing / filtering, you can certainly use additional processors to do so. For example, you could use EvaluateJsonPath to pull out specific fields of interest and then use RouteOnAttribute to route based on those fields.

Q: PutHDFS -

Is there any way to bypass writing FlowFiles on disk and directly pass those files to HDFS as is? Also, if the files are compressed (zip/gzip), can we store the files on HDFS as uncompressed?

- A: There is no way to bypass having NiFi take a copy of the data, and this is by design. NiFi is helping you formulate a graph of dataflow requirements from given source(s) through given processing steps and ultimately driving data into given destination systems. As a result, it takes on the challenge of handling transactionality of each interaction and the buffering and backpressure to deal with the realities of different production /consumption patterns. As for storing files on HDFS as uncompressed - yes: both of those formats (zip/gzip) are supported in NiFi out of the box. You simply run the data through the proper process prior to the PutHDFS process to unpack (zip) --using UnpackContent, or decompress (gzip) --using CompressContent, as needed. (In CompressContent, choose the setting for decompress mode.)

Is there any way to connect to a MapR cluster using the HDFS compatible API?

- A: Yes but it requires compiling NiFI from source. The process is rather straight forward:

1. install mapr-client
2. configure mapt-client as per MapR documentation
3. Confirm everything is working by using `maprlogin kerberos` or `maprlogin password` followed by `hadoop fs -ls -d`
4. compile using profile:
   -Pmapr -Dhadoop.version=<mapr_artifact_version_as_per_mapr_documentation> where artifact version matching your MapR release.
5. create a core-site.xml with the following content:

   ```
   <configuration>
   <property>
   <name>fs.defaultFS</name>
   <value>maprfs:///</value>
   </property>
   </configuration>
   ```
6. create the PutHDFS processor and point to the core-site.xml above

   If the cluster is running in secure mode (Kerberos or MapRSASL)

7. modify bootstrap.conf so that it contains *java.arg.15=-Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf -Dhadoop.login=hybrid*

8. Login as the user that is running NiFi (e.g. nifi) and perform a `maprlogin password`
9. using the same account, try once more to run `hadoop fs -ls`
10. If all works, start the processor

NOTES

1. In real world environments the maprticket assigned to the NiFI user **must** be a long lived ticket, such as a service ticket
2. Presently it doesn't seem to be possible to use pure Kerberos settings. Work is ongoing to try to identify how to solve this issue

Connections

Q: If I have FlowFiles stuck in my queue, how can I clear them out?

- A: Once the Apache NiFi 0.4.0 release is available you will be able to simply 'right-click -> Purge Queue'. For older versions of NiFi If you are testing a flow and do not care about what happens to the test data that is stuck in a connection queue, you can reconfigure the connection and temporarily set the FlowFile Expiration to something like 1 sec so that the FlowFiles currently in the queue will expire. Then, you can reset it to 0 sec (which means never expire). Note that you must stop both processors on either side of a connection before you can edit the settings.

Process Groups

Please add questions about process groups here.

Remote Process Groups/Site-to-Site

Q: How can I send data from one NiFi instance to another?

- A: There are generally two approaches to sending data between NiFi instances...
  - Push: The sending instance brings data to a remote process group which is pointing at the receiving NiFi, the receiving NiFi has an input port to receive the data
  - Pull: The sending instance brings data to an output port, and the receiving instance has a remote process group pointing at the output port of the sending instance
  The advantage of the first approach (push) is that the sending instance is deciding where to send its data, as opposed to data waiting at an output port for anyone to consume.

Q: When setting up site-to-site with a remote process group and networking through firewalls, are the TCP connections one-way, from source node, where the graph is running to the remote process group's node only, or are bidirectional TCP connections required? The reason I ask is I'm encountering problems trying to connect from a data center that has an open outbound firewall, but allows no incoming connections. On the target node, there is no indication in the nifi-app.log of the source node even attempting to connect (not sure if debug logging is required).

- A: On firewall configuration, indeed only TCP traffic on the UI port (8080) plus the site-to-site port (e.g., 8081) need to be open on the target node for unidirectional site-to-site operation (not required to be open on the source node's firewall). No other ports are required across firewall boundaries. The property nifi.remote.input.socket.host must be set to the external (Internet) NAT firewall address in the nifi.properties file. This is the other key configuration item, because when site-to-site connection is established, the source node must connect to the firewall (not directly to the remote target node's local IP, which is the default if this value is not configured. In addition, localhost must be enabled for local operation, as the "service nifi status" (and probably other stuff) makes calls via localhost (in case you're using iptables).

Clustering

Q: I set up a two-node cluster, but after I disconnected one node, did some work in the cluster, and tried to reconnect the second node, I got the error "Failed to connect node to cluster because local flow is different than cluster flow." I deleted the flow.xml.gz file from the node I'm attempting to add and then restarted it, but I still had the same issue. How do I resolve this?

- A: This problem can occur if you have templates on the second node that are not in sync with the cluster. On the second node, delete the flow.xml.gz and also delete the templates (located by default in conf/templates), then try to restart the node and see if it can connect to the cluster.

Q: I am evaluating NiFi for a large project to see if NiFi would fit as the main data collector. The project I am working on would require retrieving several hundreds of millions of files per day (hundreds of TB per day) so my first question is how to achieve distribution/clustering with NiFi, if that's possible.

- A: The NiFi Administrator Guide includes a quick explanation of our clustering capabilities and example configurations. This would be a great place to start and become familiar with NiFi clustering.

Q: I am running a cluster, and after I tried restarting it, I am seeing errors about trying to do something "while in safe mode". What does that mean?

- A: When you restart a cluster, it will stay in safe mode until your Primary Node connects. Here is an excerpt from the Admin Guide that explains how this works and how you can adjust it: "If the cluster restarts, the NCM will "remember" which node was the Primary Node and wait for that node to re-connect before allowing the DFM to make any changes to the dataflow. The ADMIN may adjust how long the NCM waits for the Primary Node to reconnect by adjusting the property *nifi.cluster.manager.safemode.duration* in the *nifi.properties* file." By default, that property is set to wait forever for the Primary Node to connect, but if you do not care which node is the primary node, you can set a time limit here, and the NCM will automatically elect a new Primary Node after that amount of time has passed.

Q: If we have multiple worker nodes in the cluster, do they partition the work if the source allows partitioning - eg: HDFS, or do all the nodes work on the same data? If the nodes partition the work, then how do they coordinate the work distribution and recovery, etc.? From the documentation it appears that the workers are not aware of each other.

- A: Nodes in a cluster work independently from one another and do not know about each other. Each node in a cluster runs the same flow. Typically, if you want to pull from HDFS and partition that data across the cluster, you would run ListHDFS on the Primary Node only, and then use site-to-site (remote process group) to distribute that listing to all nodes in the cluster. Each node would then pull the data that it is responsible to pull and begin working on it. We do realize that this is not ideal to have to setup this way, and it is something that we are working on so that it is much easier to have that listing automatically distributed across the cluster.

Controller Services

Please add questions about controller services here.

Reporting Tasks

Please add questions about reporting tasks here.