Hortonworks DataFlow

MiNiFi Java Agent Administration

(Feb 24, 2017)

docs.hortonworks.com

Hortonworks DataFlow: MiNiFi Java Agent Administration

Hortonworks DataFlow (HDF) is powered by Apache NiFi. A version of this documentation originally appeared on the Apache NiFi website.

HDF is the first integrated platform that solves the real time challenges of collecting and transporting data from a multitude of sources and provides interactive command and control of live flows with full and automated data provenance. HDF is a single combined platform that provides the data acquisition, simple event processing, transport and delivery mechanism designed to accommodate the diverse dataflows generated by a world of connected people, systems and things.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. Hortonworks DataFlow is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

1. MiNiFi System Administrator's Guide

1.1. Automatic Warm-Redeploy

When many MiNiFi agents running on the edge, it may not be possible to manually stop, edit the config.yml and then restart every one every time their configuration needs to change. The Config Change Coordinator and its Ingestors were designed to automatically redeploy in response to a configuration update.

The Config Change Ingestors are the means by which the agent is notified of a potential new configuration. Currently there are three:

- FileChangeIngestor
- RestChangeIngestor
- PullHttpChangeIngestor

After a new configuration has been pulled/received the Ingestors use a Differentiator in order to determine if the currently running config is different than the new config. Which Differentiator is used, is configurable for each Ingestor. Currently there is only one Differentiator:

- WholeConfigDifferentiator: Compares the entire new config with the currently running one, byte for byte.

After a new config is determined to be new, the MiNiFi agent will attempt to restart. The bootstrap first saves the old config into a swap file. The bootstrap monitors the agent as it restarts and if it fails it will roll back to the old config. If it succeeds then the swap file will be deleted and the agent will start processing using the new config.

Note: Data left in connections when the agent attempts to restart will either be mapped to a connection with the same ID in the new config, or orphaned and deleted.

The configuration for Warm-Redeploy is done in the bootstrap.conf and primarily revolve around the Config Change Ingestors. The configuration in the bootstrap.conf is done using the "nifi.minifi.notifier.ingestors" key followed by the full path name of the desired Ingestor implementation to run. Use a comma separated list to define more than one Ingestor implementation. For example:

```
nifi.minifi.notifier.ingestors=org.apache.nifi.minifi.bootstrap.configuration.
ingestors.PullHttpChangeIngestor
```

Ingestor specific configuration is also necessary and done in the bootstrap.conf as well. Specifics for each are detailed below.

1.1.1. FileChangeIngestor

class name: org.apache.nifi.minifi.bootstrap.configuration.ingestors.FileChangeIngestor

This Config Change Ingestor watches a file and when the file is updated, the file is ingested as a new config.

Note: The config file path configured here and in "nifi.minifi.config" cannot be the same. This is due to the swapping mechanism and other implementation limitations.

Below are the configuration options. The file config path is the only required property.

Option	Description
nifi.minifi.notifier.ingestors.file.config.path	Path of the file to monitor for changes. When these occur, the FileChangeNotifier, if configured, will begin the configuration reloading process
nifi.minifi.notifier.ingestors.file.polling.period.seconds	How frequently the file specified by 'nifi.minifi.notifier.file.config.path' should be evaluated for changes. If not set then a default polling period of 15 seconds will be used.
nifi.minifi.notifier.ingestors.file.differentiator	Which differentiator to use. If not set then it uses the WholeConfigDifferentiator as a default.

1.1.2. RestChangeIngestor

class name: org.apache.nifi.minifi.bootstrap.configuration.ingestors.RestChangeIngestor

This Config Change Ingestor sets up a light-weight Jetty HTTP(S) REST service in order to listen to HTTP(S) requests. A potential new configuration is sent via a POST request with the BODY being the potential new config.

NOTE: The encoding is expected to be Unicode and the exact version specified by the BOM mark ('UTF-8','UTF-16BE' or 'UTF-16LE'). If there is no BOM mark, then UTF-8 is used.

Here is an example post request using 'curl' hitting the local machine on pot 8338 and it is executed with the config file "config.yml" in the directory the command is run from:

```
curl --request POST --data-binary "@config.yml" http://localhost:8338/
```

Below are the configuration options. There are no required options. If no properties are set then the server will bind to hostname "localhost" on a random open port, will only connect via HTTP and will use the WholeConfigDifferentiator.

Option	Description
nifi.minifi.notifier.ingestors.receive.http.host	Hostname on which the Jetty server will bind to. If not specified then it will bind to localhost.
nifi.minifi.notifier.ingestors.receive.http.port	Port on which the Jetty server will bind to. If not specified then it will bind to a random open port.
nifi.minifi.notifier.ingestors.receive.http.truststore.location	If using HTTPS, this specifies the location of the truststore.
nifi.minifi.notifier.ingestors.receive.http.truststore.password	If using HTTPS, this specifies the password of the truststore.
nifi.minifi.notifier.ingestors.receive.http.truststore.type	If using HTTPS, this specifies the type of the truststore.
nifi.minifi.notifier.ingestors.receive.http.keystore.location	If using HTTPS, this specifies the location of the keystore.
nifi.minifi.notifier.ingestors.receive.http.keystore.password	If using HTTPS, this specifies the password of the keystore.
nifi.minifi.notifier.ingestors.receive.http.keystore.type	If using HTTPS, this specifies the type of the keystore.
nifi.minifi.notifier.ingestors.receive.http.need.client.auth	If using HTTPS, this specifies whether or not to require client authentication.
nifi.minifi.notifier.ingestors.receive.http.differentiator	Which differentiator to use. If not set then it uses the WholeConfigDifferentiator as a default.

1.1.3. PullHttpChangeIngestor

class name:
org.apache.nifi.minifi.bootstrap.configuration.ingestors.PullHttpChangeIngestor

This Config Change Ingestor periodically sends a GET request to a REST endpoint using HTTP(S) to order to pull the potential new config.

Below are the configuration options. The hostname and port are the only required properties.

Option	Description
nifi.minifi.notifier.ingestors.pull.http.hostname	Hostname on which to pull configurations from
nifi.minifi.notifier.ingestors.pull.http.port	Port on which to pull configurations from
nifi.minifi.notifier.ingestors.pull.http.path	Path on which to pull configurations from
nifi.minifi.notifier.ingestors.pull.http.period.ms	Period on which to pull configurations from, defaults to 5 minutes if not set.
nifi.minifi.notifier.ingestors.pull.http.use.etag	If the destination server is set up with cache control ability and utilizes an "ETag" header, then this should be set to true to utilize it. Very simply, the Ingestor remembers the "ETag" of the last successful pull (returned 200) then uses that "ETag" in a "If-None-Match" header on the next request.
nifi.minifi.notifier.ingestors.pull.http.connect.timeout.ms	Sets the connect timeout for new connections. A value of 0 means no timeout, otherwise values must be a positive whole number in milliseconds.
nifi.minifi.notifier.ingestors.pull.http.read.timeout.ms	Sets the read timeout for new connections. A value of 0 means no timeout, otherwise values must be a positive whole number in milliseconds.
nifi.minifi.notifier.ingestors.pull.http.truststore.location	If using HTTPS, this specifies the location of the truststore.
nifi.minifi.notifier.ingestors.pull.http.truststore.password	If using HTTPS, this specifies the password of the truststore.
nifi.minifi.notifier.ingestors.pull.http.truststore.type	If using HTTPS, this specifies the type of the truststore.
nifi.minifi.notifier.ingestors.pull.http.keystore.location	If using HTTPS, this specifies the location of the keystore.
nifi.minifi.notifier.ingestors.pull.http.keystore.password	If using HTTPS, this specifies the password of the keystore.
nifi.minifi.notifier.ingestors.pull.http.keystore.type	If using HTTPS, this specifies the type of the keystore.
nifi.minifi.notifier.ingestors.pull.http.differentiator	Which differentiator to use. If not set then it uses the WholeConfigDifferentiator as a default.

1.2. Status Reporting and Querying

In NiFi there is a lot of information, such as stats and bulletins, that is only available to view through the UI. MiNiFi provides access to this information through a query mechanism. You can query FlowStatus either using the MiNiFi.sh script or by configuring one of the Periodic Status Reporters. The API for the query is the same for the reporters and the "flowStatus" script option. The API is outlined in the "FlowStatus Query Options" section below.

1.2.1. FlowStatus Script Query

From the minifi.sh script, you can manually query to get the current status of your dataflow. The following is an example of a minifi.sh query you might run to view health,

stats, and bulletins for the TailFile processor. This query returns information to your command-line.

```
minifi.sh flowStatus processor:TailFile:health,stats,bulletins
```

Currently the script only accepts one high level option at a time. Also any names of connections, remote process groups, or processors that contain " " (a space), ":", ";" or "," cause parsing errors when querying.

1.2.2. Periodic Status Reporters

You can set up Periodic Status Reporters to periodically report the status of your dataflow. The query executes at configurable intervals and the results are reported using the configured implementation. Configure the Reporters in the bootstrap.conf file, using the "nifi.minifi.status.reporter.components" key followed by the full path name of the desired Reporter implementation to run. Use a comma separated list to define more than one Reporter implementation. For example:

```
nifi.minifi.status.reporter.components=org.apache.nifi.minifi.bootstrap.
status.reporters.StatusLogger
```

1.2.2.1. StatusLogger

class name: org.apache.nifi.minifi.bootstrap.status.reporters.StatusLogger

The Periodic Status Reporter logs the results of the query to the logs. By default it is logged to the minifi-bootstrap.log but you can modify logback.xml to log to an alternate file and location.

Option	Description
nifi.minifi.status.reporter.log.query	The FlowStatus query to run.
nifi.minifi.status.reporter.log.level	The log level at which to log the status. Available options are "TRACE", "DEBUG", "INFO", "WARN" and "ERROR".
nifi.minifi.status.reporter.log.period	The delay (in milliseconds) between each query.

Example `bootstrap.conf` configuration:

```
# The FlowStatus query to submit to the MiNiFi
instance nifi.minifi.status.reporter.log.query=instance:health,bulletins
# The log level at which the status will be logged
nifi.minifi.status.reporter.log.level=INFO
# The period (in milliseconds) at which to log the status
nifi.minifi.status.reporter.log.period=60000
```

Example `logback.xml` configuration to output the status to its own rolling log file:

```
<appender name="STATUS_LOG_FILE" class="ch.qos.logback.core.rolling.
RollingFileAppender">
    <file>logs/minifi-status.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.
TimeBasedRollingPolicy">
      <!-- For daily rollover, use 'user_%d.log'.
           For hourly rollover, use 'user_%d{yyyy-MM-dd_HH}.log'.
           To GZIP rolled files, replace '.log' with '.log.gz'.
           To ZIP rolled files, replace '.log' with '.log.zip'.
      -->
      <fileNamePattern>./logs/minifi-status_%d.log</fileNamePattern>
```

```
      <!-- keep 5 log files worth of history -->
      <maxHistory>5</maxHistory>
    </rollingPolicy>
    <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
      <pattern>%date %level [%thread] %logger{40} %msg%n</pattern>
    </encoder>
</appender>

<logger name="org.apache.nifi.minifi.bootstrap.status.reporters.StatusLogger"
 level="INFO" additivity="false">
    <appender-ref ref="STATUS_LOG_FILE" />
</logger>
```

1.2.3. FlowStatus Query Options

This section outlines each option to query the MiNiFi instance for the FlowStatus.

1.2.3.1. Processors

To query the processors use the "processor" flag followed by the processor ID to get (or "all") followed by one of the processor options. The processor options are below.

Option	Description
health	The processor's run status, whether or not it has bulletins and the validation errors (if there are any).
bulletins	A list of all the current bulletins (if there are any).
stats	The current stats of the processor. This includes but is not limited to active threads and FlowFiles sent/received.

An example query to get the health, bulletins and stats of the "TailFile" processor is below.

```
minifi.sh flowStatus processor:TailFile:health,stats,bulletins
```

1.2.3.2. Connections

To query the connections use the "connection" flag followed by the connection ID to get (or "all") followed by one of the connection options. The connection options are below.

Option	Description
health	The connections's queued bytes and queued FlowFile count.
stats	The current stats of the connection. This includes input/output count and input/output bytes.

An example query to get the health and stats of the "TailToS2S" connection is below.

```
minifi.sh flowStatus connection:TailToS2S:health,stats
```

1.2.3.3. Remote Process Groups

To query the remote process groups (RPG) use the "remoteProcessGroup" flag followed by the RPG ID to get (or "all") followed by one of the remote process group options. The remote process group options are below.

Option	Description
health	The connections's queued bytes and queued FlowFile count.

Option	Description
bulletins	A list of all the current bulletins (if there are any).
inputPorts	A list of every input port for this RPG and their status. Their status includes it's name, whether the target exit and whether it's currently running.
stats	The current stats of the RPG. This includes the active threads, sent content size and count.

An example query to get the health, bulletins, input ports and stats of all the RPGS is below.

```
minifi.sh flowStatus remoteprocessgroup:all:health,bulletins,inputports,stats
```

1.2.3.4. Controller Services

To query the controller services use the "controllerServices" flag followed by one of the controller service options. The controller service options are below.

Option	Description
health	The controller service's state, whether or not it has bulletins and any validation errors.
bulletins	A list of all the current bulletins (if there are any).

An example query to get the health and bulletins of all the controller services is below.

```
minifi.sh flowStatus controllerservices:health,bulletins
```

1.2.3.5. Provenance Reporting

To query the status of the provenance reporting use the "provenancereporting" flag followed by one of the provenance reporting options. The provenance reporting options are below.

Option	Description
health	The provenance reporting state, active threads, whether or not it has bulletins and any validation errors.
bulletins	A list of all the current bulletins (if there are any).

An example query to get the health and bulletins of the provenance reporting is below.

```
minifi.sh flowStatus provenancereporting:health,bulletins
```

1.2.3.6. Instance

To query the status of the MiNiFi instance use the "instance" flag followed by one of the instance options. The instance options are below.

Option	Description
health	The provenance reporting state, active threads, whether or not it has bulletins and any validation errors.
bulletins	A list of all the current bulletins (if there are any).
stats	The current stats of the instance. This including but not limited to bytes read/written and FlowFiles sent/ transferred.

An example query to get the health, stats and bulletins of the instance is below.

```
minifi.sh flowStatus instance:health,stats,bulletins
```

1.2.3.7. System Diagnostics

To query the system diagnostics use the "systemdiagnostics" flag followed by one of the system diagnostics options. The system diagnostics options are below.

Option	Description
heap	Information detailing the state of the JVM heap.
processorstats	The system processor stats. This includes the available processors and load average.
contentrepositoryusage	A list of each content repository and stats detailing its usage.
flowfilerepositoryusage	Stats about the current usage of the FlowFile repository.
garbagecollection	A list of the garbage collection events, detailing their name, collection count and time.

An example query to get the heap, processor stats, content repository usage, FlowFile repository usage and garbage collection from the system diagnostics is below.

```
minifi.sh flowStatus systemdiagnostics:heap,processorstats,
contentrepositoryusage,flowfilerepositoryusage,garbagecollection
```

1.2.3.8. Example

This is an example of a simple query to get the health of all the processors and its results from a simple flow:

```
User:minifi-0.0.1-SNAPSHOT user ./bin/minifi.sh flowStatus
 processor:all:health

Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_74.jdk/Contents/Home
MiNiFi home: /Users/user/projects/nifi-minifi/minifi-assembly/target/minifi-0.
0.1-SNAPSHOT-bin/minifi-0.0.1-SNAPSHOT

Bootstrap Config File: /Users/user/projects/nifi-minifi/minifi-assembly/
target/minifi-0.0.1-SNAPSHOT-bin/minifi-0.0.1-SNAPSHOT/conf/bootstrap.conf

FlowStatusReport{controllerServiceStatusList=null, processorStatusList=
[{name='TailFile', processorHealth={runStatus='Running', hasBulletins=false,
 validationErrorList=[]}, processorStats=null,
bulletinList=null}], connectionStatusList=null, remoteProcessGroupStatusList=
null, instanceStatus=null, systemDiagnosticsStatus=null,
 reportingTaskStatusList=null, errorsGeneratingReport=[]}
```

1.3. Config File

The config.yml in the *conf* directory is the main configuration file for controlling how MiNiFi runs. This section provides an overview of the properties in this file. The file is a YAML and follows the YAML format laid out here.

Alternatively, the MiNiFi Toolkit Converter can aid in creating a config.yml from a generated template exported from a NiFi instance. This tool can be downloaded from http://nifi.apache.org/minifi/download.html under the `MiNiFi Toolkit Binaries` section. Information on the toolkit's usage is available at https://nifi.apache.org/minifi/minifi-toolkit.html.

Note

Values for periods of time and data sizes must include the unit of measure, for example "10 sec" or "10 MB", not simply "10".

1.3.1. Versioning

The "MiNiFi Config Version" property is used to indicate to the configuration parser which version of the config file it is looking at. If the property is empty or missing, version 1 is assumed.

The MiNiFi Toolkit Converter is capable of parsing previous versions (possibly subject to a future deprecation policy) and writing out the current version. It can also validate that a given config file parses and upconverts to the current version without issue.

1.3.1.1. Version 1 -> Version 2 changes

1. Use ids instead of names for processors, connections.
2. Allow multiple source relationships for connections.
3. Support process groups, input ports, output ports
4. Change Id Key for RPGs from "Remote Processing Groups" to the proper "Remote Process Groups" (not "ing")

1.3.2. Flow Controller

The first section of config.yml is for naming and commenting on the file.

Property	Description
MiNiFi Config Version	The version of the configuration file. The default value if this property is missing or empty is 1, the current value is 2.
name	The name of the file.
comment	A comment describing the usage of this config file.

1.3.3. Core Properties

The Core Properties section applies to the core framework as a whole.

Property	*Description*
flow controller graceful shutdown period	Indicates the shutdown period. The default value is 10 sec.
flow service write delay interval	When many changes are made to the flow.xml, this property specifies how long to wait before writing out the changes, so as to batch the changes into a single write. The default value is 500 ms.
administrative yield duration	If a component allows an unexpected exception to escape, it is considered a bug. As a result, the framework will pause (or administratively yield) the component for this amount of time. This is done so that the component does not use up massive amounts of system resources, since it is known to have problems in the existing state. The default value is 30 sec.
bored yield duration	When a component has no work to do (i.e., is "bored"), this is the amount of time it will wait before checking to

Property	*Description*
	see if it has new data to work on. This way, it does not use up CPU resources by checking for new work too often. When setting this property, be aware that it could add extra latency for components that do not constantly have work to do, as once they go into this "bored" state, they will wait this amount of time before checking for more work. The default value is 10 millis.
max concurrent threads	The maximum number of threads any processor can have running at one time.

1.3.4. FlowFile Repository

The FlowFile repository keeps track of the attributes and current state of each FlowFile in the system. By default, this repository is installed in the same root installation directory as all the other repositories; however, it is advisable to configure it on a separate drive if available.

Property	*Description*
partitions	The number of partitions. The default value is 256.
checkpoint interval	The FlowFile Repository checkpoint interval. The default value is 2 mins.
always sync	If set to *true*, any change to the repository will be synchronized to the disk, meaning that NiFi will ask the operating system not to cache the information. This is very expensive and can significantly reduce NiFi performance. However, if it is *false*, there could be the potential for data loss if either there is a sudden power loss or the operating system crashes. The default value is*false*.

1.3.4.1. Swap Subsection

A part of the FlowFile Repository section there is a Swap subsection.

NiFi keeps FlowFile information in memory (the JVM) but during surges of incoming data, the FlowFile information can start to take up so much of the JVM that system performance suffers. To counteract this effect, NiFi "swaps" the FlowFile information to disk temporarily until more JVM space becomes available again. The "Swap" subsection of properties govern how that process occurs.

Property	*Description*
threshold	The queue threshold at which NiFi starts to swap FlowFile information to disk. The default value is 20000.
in period	The swap in period. The default value is 5 sec.
in threads	The number of threads to use for swapping in. The default value is 1.
out period	The swap out period. The default value is 5 sec.
out threads	The number of threads to use for swapping out. The default value is 4.

1.3.5. Content Repository

The Content Repository holds the content for all the FlowFiles in the system. By default, it is installed in the same root installation directory as all the other repositories; however, administrators will likely want to configure it on a separate drive if available. If nothing

else, it is best if the Content Repository is not on the same drive as the FlowFile Repository. In dataflows that handle a large amount of data, the Content Repository could fill up a disk and the FlowFile Repository, if also on that disk, could become corrupt. To avoid this situation, configure these repositories on different drives.

Property	***Description***
content claim max appendable size	The maximum size for a content claim. The default value is 10 MB.
content claim max flow files	The maximum number of FlowFiles to assign to one content claim. The default value is 100.
always sync	If set to *true*, any change to the repository will be synchronized to the disk, meaning that NiFi will ask the operating system not to cache the information. This is very expensive and can significantly reduce NiFi performance. However, if it is *false*, there could be the potential for data loss if either there is a sudden power loss or the operating system crashes. The default value is*false*.

1.3.6. Provenance Repository

Property	***Description***
provenance rollover time	The amount of time to wait before rolling over the latest data provenance information so that it is available to be accessed by components. The default value is 1 min.

1.3.7. Component Status Repository

The Component Status Repository contains the information for the Component Status History tool in the User Interface. These properties govern how that tool works.

The buffer.size and snapshot.frequency work together to determine the amount of historical data to retain. As an example to configure two days worth of historical data with a data point snapshot occurring every 5 minutes you would configure snapshot.frequency to be "5 mins" and the buffer.size to be "576". To further explain this example for every 60 minutes there are 12 (60 / 5) snapshot windows for that time period. To keep that data for 48 hours (12 * 48) you end up with a buffer size of 576.

Property	***Description***
buffer size	Specifies the buffer size for the Component Status Repository. The default value is 1440.
snapshot frequency	This value indicates how often to present a snapshot of the components' status history. The default value is 1 min.

1.3.8. Security Properties

These properties pertain to various security features in NiFi. Many of these properties are covered in more detail in the Security Configuration section of this Administrator's Guide.

Property	***Description***
keystore	The full path and name of the keystore. It is blank by default.
keystore type	The keystore type. It is blank by default.
keystore password	The keystore password. It is blank by default.
key password	The key password. It is blank by default.

Property	*Description*
truststore	The full path and name of the truststore. It is blank by default.
truststore type	The truststore type. It is blank by default.
truststore password	The truststore password. It is blank by default.
ssl protocol	The protocol to use when communicating via https. Necessary to transfer provenance securely.

1.3.8.1. Sensitive Properties Subsection

Some properties for processors are marked as *sensitive* and should be encrypted. These following properties will be used to encrypt the properties while in use by MiNiFi. This will currently *not* be used to encrypt properties in the config file.

Property	*Description*
key	This is the password used to encrypt any sensitive property values that are configured in processors. By default, it is blank, but the system administrator should provide a value for it. It can be a string of any length, although the recommended minimum length is 10 characters. Be aware that once this password is set and one or more sensitive processor properties have been configured, this password should not be changed.
algorithm	The algorithm used to encrypt sensitive properties. The default value is `PBEWITHMD5AND256BITAES-CBC-OPENSSL`.
provider	The sensitive property provider. The default value is BC.

1.3.9. Processors

The current implementation of MiNiFi supports multiple processors. the "Processors" subsection is a list of these processors. Each processor must specify these properties. They are the basic configuration general to all processor implementations. Make sure that all relationships for a processor are accounted for in the auto-terminated relationship list or are used in a connection.

Property	*Description*
name	The name of what this processor will do. This is not used for any underlying implementation but solely for the users of this configuration and MiNiFi agent.
id	The id of this processor. This can be omitted but in processors without this field, there should not be any duplicate names and connections will need to specify source and destination name instead of id. If set it should be a filesystem-friendly value (regex: [A-Za-z0-9_-]+)
class	The fully qualified java class name of the processor to run. For example for the standard TailFile processor it would be: org.apache.nifi.processors.standard.TailFile
max concurrent tasks	The maximum number of tasks that the processor will use.
scheduling strategy	The strategy for executing the processor. Valid options are `CRON_DRIVEN` or `TIMER_DRIVEN`
scheduling period	This property expects different input depending on the scheduling strategy selected. For the`TIMER_DRIVEN` scheduling strategy, this value is a time duration specified by a number followed by a time unit. For example, 1

Property	*Description*
	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 in the scheduling tab section of the NiFi User documentation.
penalization period	Specifies how long FlowFiles will be penalized.
yield period	In the event the processor cannot make progress it should `yield` which will prevent the processor from being scheduled to run for some period of time. That period of time is specific using this property.
run duration nanos	If the processor supports batching this property can be used to control how long the Processor should be scheduled to run each time that it is triggered. Smaller values will have lower latency but larger values will have higher throughput. This period should typically only be set between 0 and 2000000000 (2 seconds).
auto-terminated relationships list	A YAML list of the relationships to auto-terminate for the processor.
annotation data	Some processors make use of "Annotation Data" in order to do more complex configuration, such as the Advanced portion of UpdateAttribute. This data will be unique to each implementing processor and more than likely will not be written out manually.

1.3.9.1. Processor Properties

Within the Processor Configuration section, there is the `Properties` subsection. The keys and values in this section are the property names and values for the processor. For example the TailFile processor would have a section like this:

```
Properties:
    File to Tail: logs/nifi-app.log
    Rolling Filename Pattern: nifi-app*
    State File: ./conf/state/tail-file
    Initial Start Position: Beginning of File
```

1.3.10. Process Groups

Process groups can be nested from the top level. They can contain other process groups as well and can be used to logically group related operations.

Property	*Description*
name	The name of what this process group will do.
id	The id of this process group. This needs to be set to a unique filesystem-friendly value (regex: [A-Za-z0-9_-]+)
Processors	The processors contained in this Process Group. (Defined above)
Remote Process Groups	The remote process groups contained in this Process Group. (Defined below)
Connections	The connections contained in this Process Group. (Defined below)
Input Ports	The input ports contained in this Process Group. (Defined below)

Property	*Description*
Output Ports	The output ports contained in this Process Group. (Defined below)
Funnels	The funnels contained in this Process Group. (Defined below)
Process Groups	The child Process Groups contained in this Process Group.

1.3.11. Input Ports

These ports provide input to the Process Group they reside on. (Currently only for internal Input ports.)

Property	*Description*
name	The name of what this input port will do.
id	The id of this input port. This needs to be set to a unique filesystem-friendly value (regex: [A-Za-z0-9_-]+)

1.3.12. Output Ports

These ports provide output from the Process Group they reside on. (Currently only for internal Output ports.)

Property	*Description*
name	The name of what this output port will do.
id	The id of this output port. This needs to be set to a unique filesystem-friendly value (regex: [A-Za-z0-9_-]+)

1.3.13. Funnels

Funnels can be used to combine outputs from multiple processors into a single connection for ease of design.

Property	*Description*
id	The id of this funnel. This needs to be set to a unique filesystem-friendly value (regex: [A-Za-z0-9_-]+)

1.3.14. Connections

There can be multiple connections in this version of MiNiFi. The "Connections" subsection is a list of connections. Each connection must specify these properties.

Property	*Description*
name	The name of what this connection will do. This is used for the id of the connection so it must be unique.
id	The id of this connection. This needs to be left empty or set to a filesystem-friendly value (regex: [A-Za-z0-9_-]+)
source name	The name of what of the processor that is the source for this connection.
source relationship name	The name of the processors relationship to route to this connection
destination name	The name of the component to receive this connection.

Property	*Description*
max work queue size	This property is the max number of FlowFiles that can be in the queue before back pressure is applied. When back pressure is applied the source processor will no longer be scheduled to run.
max work queue data size	This property specifies the maximum amount of data (in size) that should be queued up before applying back pressure. When back pressure is applied the source processor will no longer be scheduled to run.
flowfile expiration	Indicates how long FlowFiles are allowed to exist in the connection before be expired (automatically removed from the flow).
queue prioritizer class	This configuration option specifies the fully qualified java class path of a queue prioritizer to use. If no special prioritizer is desired then it should be left blank. An example value of this property is: org.apache.nifi.prioritizer.NewestFlowFileFirstPrioritizer

1.3.15. Remote Process Groups

MiNiFi can be used to send data using the Site to Site protocol (via a Remote Process Group) or a Processor. These properties configure the Remote Process Groups that use Site-To-Site to send data to a core instance.

Property	*Description*
name	The name of what this Remote Process Group points to. This is not used for any underlying implementation but solely for the users of this configuration and MiNiFi agent.
comment	A comment about the Remote Process Group. This is not used for any underlying implementation but solely for the users of this configuration and MiNiFi agent.
url	The URL of the core NiFi instance.
timeout	How long MiNiFi should wait before timing out the connection.
yield period	When communication with this Remote Process Group fails, it will not be scheduled again for this amount of time.
transport protocol	The transport protocol to use for this Remote Process Group. Can be either "RAW" or "HTTP"

1.3.15.1. Input Ports Subsection

When connecting via Site to Site, MiNiFi needs to know which input port to communicate to of the core NiFi instance. These properties designate and configure communication with that port.

Property	*Description*
id	The id of the input port as it exists on the core NiFi instance. To get this information access the UI of the core instance, right the input port that is desired to be connect to and select "configure". The id of the port should under the "Id" section.
name	The name of the input port as it exists on the core NiFi instance. To get this information access the UI of the core instance, right the input port that is desired to be connect to and select "configure". The id of the port should under the "Port name" section.

Property	*Description*
comments:	A comment about the Input Port. This is not used for any underlying implementation but solely for the users of this configuration and MiNiFi agent.
max concurrent tasks	The number of tasks that this port should be scheduled for at maximum.
use compression	Whether or not compression should be used when communicating with the port. This is a boolean value of either "true" or "false"

1.3.16. Provenance Reporting

MiNiFi is currently designed only to report provenance data using the Site to Site protocol. These properties configure the underlying reporting task that sends the provenance events.

Property	*Description*
comment	A comment about the Provenance reporting. This is not used for any underlying implementation but solely for the users of this configuration and MiNiFi agent.
scheduling strategy	The strategy for executing the Reporting Task. Valid options are `CRON_DRIVEN` or `TIMER_DRIVEN`
scheduling period	This property expects different input depending on the scheduling strategy selected. For 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 in the scheduling tab section of the NiFi User documentation.
destination url	The URL to post the Provenance Events to.
port name	The name of the input port as it exists on the receiving NiFi instance. To get this information access the UI of the core instance, right the input port that is desired to be connect to and select "configure". The id of the port should under the "Port name" section.
originating url	The URL of this MiNiFi instance. This is used to include the Content URI to send to the destination.
use compression	Indicates whether or not to compress the events when being sent.
timeout	How long MiNiFi should wait before timing out the connection.
batch size	Specifies how many records to send in a single batch, at most. This should be significantly above the expected amount of records generated between scheduling. If it is not, then there is the potential for the Provenance reporting to lag behind event generation and never catch up.

1.4. Example Config File

Below are two example config YAML files. The first tails the minifi-app.log, send the tailed log and provenance data back to a secure instance of NiFi. The second

uses a series of processors to tail the app log, routes off only lines that contain "WriteAheadFlowFileRepository" and puts it as a file in the "./" directory.

```
MiNiFi Config Version: 1
Flow Controller:
    name: MiNiFi Flow
    comment:

Core Properties:
    flow controller graceful shutdown period: 10 sec
    flow service write delay interval: 500 ms
    administrative yield duration: 30 sec
    bored yield duration: 10 millis

FlowFile Repository:
    partitions: 256
    checkpoint interval: 2 mins
    always sync: false
    Swap:
        threshold: 20000
        in period: 5 sec
        in threads: 1
        out period: 5 sec
        out threads: 4

Provenance Repository:
    provenance rollover time: 1 min

Content Repository:
    content claim max appendable size: 10 MB
    content claim max flow files: 100
    always sync: false

Component Status Repository:
    buffer size: 1440
    snapshot frequency: 1 min

Security Properties:
    keystore: /tmp/ssl/localhost-ks.jks
    keystore type: JKS
    keystore password: localtest
    key password: localtest
    truststore: /tmp/ssl/localhost-ts.jks
    truststore type: JKS
    truststore password: localtest
    ssl protocol: TLS
    Sensitive Props:
        key:
        algorithm: PBEWITHMD5AND256BITAES-CBC-OPENSSL
        provider: BC

Processors:
    - name: TailFile
      class: org.apache.nifi.processors.standard.TailFile
      max concurrent tasks: 1
      scheduling strategy: TIMER_DRIVEN
      scheduling period: 1 sec
      penalization period: 30 sec
      yield period: 1 sec
      run duration nanos: 0
```

```
      auto-terminated relationships list:
      Properties:
          File to Tail: logs/minifi-app.log
          Rolling Filename Pattern: minifi-app*
          Initial Start Position: Beginning of File

Connections:
    - name: TailToS2S
      source name: TailFile
      source relationship name: success
      destination name: 8644cbcc-a45c-40e0-964d-5e536e2ada61
      max work queue size: 0
      max work queue data size: 1 MB
      flowfile expiration: 60 sec
      queue prioritizer class: org.apache.nifi.prioritizer.
NewestFlowFileFirstPrioritizer

Remote Processing Groups:
    - name: NiFi Flow
      comment:
      url: https://localhost:8090/nifi
      timeout: 30 secs
      yield period: 10 sec
      Input Ports:
          - id: 8644cbcc-a45c-40e0-964d-5e536e2ada61
            name: tailed log
            comments:
            max concurrent tasks: 1
            use compression: false

Provenance Reporting:
    comment:
    scheduling strategy: TIMER_DRIVEN
    scheduling period: 30 sec
    destination url: https://localhost:8090/
    port name: provenance
    originating url: http://${hostname(true)}:8081/nifi
    use compression: true
    timeout: 30 secs
    batch size: 1000
```

```
``` yaml
Flow Controller:
    name: MiNiFi Flow
    comment:

Core Properties:
    flow controller graceful shutdown period: 10 sec
    flow service write delay interval: 500 ms
    administrative yield duration: 30 sec
    bored yield duration: 10 millis
    max concurrent threads: 1

FlowFile Repository:
    partitions: 256
    checkpoint interval: 2 mins
    always sync: false
    Swap:
```
```

```
        threshold: 20000
        in period: 5 sec
        in threads: 1
        out period: 5 sec
        out threads: 4

Content Repository:
    content claim max appendable size: 10 MB
    content claim max flow files: 100
    always sync: false

Component Status Repository:
    buffer size: 1440
    snapshot frequency: 1 min

Security Properties:
    keystore: /tmp/ssl/localhost-ks.jks
    keystore type: JKS
    keystore password: localtest
    key password: localtest
    truststore: /tmp/ssl/localhost-ts.jks
    truststore type: JKS
    truststore password: localtest
    ssl protocol: TLS
    Sensitive Props:
        key:
        algorithm: PBEWITHMD5AND256BITAES-CBC-OPENSSL
        provider: BC

Processors:
    - name: TailAppLog
      class: org.apache.nifi.processors.standard.TailFile
      max concurrent tasks: 1
      scheduling strategy: TIMER_DRIVEN
      scheduling period: 10 sec
      penalization period: 30 sec
      yield period: 1 sec
      run duration nanos: 0
      auto-terminated relationships list:
      Properties:
          File to Tail: logs/minifi-app.log
          Rolling Filename Pattern: minifi-app*
          Initial Start Position: Beginning of File
    - name: SplitIntoSingleLines
      class: org.apache.nifi.processors.standard.SplitText
      max concurrent tasks: 1
      scheduling strategy: TIMER_DRIVEN
      scheduling period: 0 sec
      penalization period: 30 sec
      yield period: 1 sec
      run duration nanos: 0
      auto-terminated relationships list:
          - failure
          - original
      Properties:
          Line Split Count: 1
          Header Line Count: 0
          Remove Trailing Newlines: true
    - name: RouteErrors
      class: org.apache.nifi.processors.standard.RouteText
```

```
      max concurrent tasks: 1
      scheduling strategy: TIMER_DRIVEN
      scheduling period: 0 sec
      penalization period: 30 sec
      yield period: 1 sec
      run duration nanos: 0
      auto-terminated relationships list:
          - unmatched
          - original
      Properties:
          Routing Strategy: Route to 'matched' if line matches all conditions
          Matching Strategy: Contains
          Character Set: UTF-8
          Ignore Leading/Trailing Whitespace: true
          Ignore Case: true
          Grouping Regular Expression:
          WALFFR: WriteAheadFlowFileRepository
    - name: PutFile
      class: org.apache.nifi.processors.standard.PutFile
      max concurrent tasks: 1
      scheduling strategy: TIMER_DRIVEN
      scheduling period: 0 sec
      penalization period: 30 sec
      yield period: 1 sec
      run duration nanos: 0
      auto-terminated relationships list:
          - failure
          - success
      Properties:
          Directory: ./
          Conflict Resolution Strategy: replace
          Create Missing Directories: true
          Maximum File Count:
          Last Modified Time:
          Permissions:
          Owner:
          Group:

Connections:
    - name: TailToSplit
      source name: TailAppLog
      source relationship name: success
      destination name: SplitIntoSingleLines
      max work queue size: 0
      max work queue data size: 1 MB
      flowfile expiration: 60 sec
      queue prioritizer class: org.apache.nifi.prioritizer.
NewestFlowFileFirstPrioritizer
    - name: SplitToRoute
      source name: SplitIntoSingleLines
      source relationship name: splits
      destination name: RouteErrors
      max work queue size: 0
      max work queue data size: 1 MB
      flowfile expiration: 60 sec
      queue prioritizer class: org.apache.nifi.prioritizer.
NewestFlowFileFirstPrioritizer
    - name: RouteToS2S
      source name: RouteErrors
      source relationship name: matched
```

```
      destination name: PutFile
      max work queue size: 0
      max work queue data size: 1 MB
      flowfile expiration: 60 sec
      queue prioritizer class: org.apache.nifi.prioritizer.
NewestFlowFileFirstPrioritizer

Provenance Reporting:
    comment:
    scheduling strategy: TIMER_DRIVEN
    scheduling period: 30 sec
    destination url: https://localhost:8080/
    port name: provenance
    originating url: http://${hostname(true)}:8081/nifi
    use compression: true
    timeout: 30 secs
    batch size: 1000
```