

Hortonworks DataFlow

Upgrade

(Feb 24, 2017)

Hortonworks DataFlow: Upgrade

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

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.



Except where otherwise noted, this document is licensed under
Creative Commons Attribution ShareAlike 3.0 License.
<http://creativecommons.org/licenses/by-sa/3.0/legalcode>

Table of Contents

1. NiFi Upgrade Guide	1
1.1. Upgrade Overview	1
1.1.1. Minimum JRE/JDK Support	1
1.1.2. Kerberos System Properties	1
1.1.3. DBCPConnectionPool Service	2
1.1.4. MonitorDiskUsage Reporting Task	2
1.1.5. Connection/Relationship Default Back Pressure Settings	2
1.1.6. Multi-Tenant Authorization Model	2
1.1.7. HTTP(S) Site-to-Site	3
1.1.8. Zero-Master Clustering	3
1.1.9. Secured Zookeeper	4
1.1.10. QueryDatabaseTable Processor	5
1.1.11. TailFile Processor	5
1.1.12. LDAP referral strategy 'IGNORE' bug fix	5
1.2. Before you Begin	5
1.2.1. Optimizing your NiFi Directory Structure for Upgrade	5
1.2.2. Preserving your custom processors	7
1.2.3. Preserving customized NAR files	7
1.3. Upgrading a single instance of NiFi	7
1.4. Upgrading a NiFi cluster	9

1. NiFi Upgrade Guide

You can use this document upgrading from NiFi 0.x to NiFi 1.0.0, for both standalone and clustered NiFi instances. You can perform the NiFi upgrade process with minimal interruption to production dataflows.

- [Upgrade Overview](#)
- [Before you Begin](#)
- [Upgrading a Single NiFi Instance](#)
- [Upgrading a NiFi Cluster](#)

1.1. Upgrade Overview

This section describes the changes made to NiFi 1.0.0. Because NiFi 1.0.0 is a major release, you should review this section carefully, and ensure that you understand the impact that these changes may have on your existing dataflows.

- [Minimum JRE/JDK Support](#)
- [Kerberos System Properties](#)
- [DBCPConnectionPool Service](#)
- [MonitorDiskUsage Reporting Task](#)
- [Connection/Relationship Default Back Pressure Settings](#)
- [Multi-tenant Authorization Model](#)
- [HTTP\(S\) Site-to-Site](#)
- [Zero-Master Clustering](#)
- [Secured Zookeeper](#)
- [QueryDatabaseTable Processor](#)
- [TailFile Processor](#)
- [LDAP referral strategy 'IGNORE' bug fix](#)

1.1.1. Minimum JRE/JDK Support

Java 8 is now the minimum JRE/JDK supported.

1.1.2. Kerberos System Properties

SPNEGO and service principals for Kerberos are now established via separate system properties.

New SPNEGO properties:

- nifi.kerberos.spnego.principal
- nifi.kerberos.spnego.keytab.location
- nifi.kerberos.spnego.authentication.expiration

New service properties:

- nifi.kerberos.service.principal
- nifi.kerberos.service.keytab.location

Removed properties:

- nifi.kerberos.keytab.location
- nifi.kerberos.authentication.expiration

1.1.3. DBCPConnectionPool Service

The Database Driver Jar Url property has been replaced by the Database Driver Location(s) property which accepts a comma-separated list of URLs or local files and folders containing the driver JAR.

Existing processors that reference this service will be invalid until you have configured the new Database Driver Location(s) property.

1.1.4. MonitorDiskUsage Reporting Task

This standard reporting task has been simplified to let you specify a logical name, a directory, and a threshold to monitor. Previously it was tightly coupled to the internal flow file and content repositories in a manner that did not align to the pluggable nature of those repositories.

The new approach gives the user total control over what they want MonitorDiskUsage to monitor.

1.1.5. Connection/Relationship Default Back Pressure Settings

In previous versions, no backpressure settings were supplied by default. In NiFi 1.0.0, new connections made have a default value set of 10,000 flow files and 1GB worth of data size.

1.1.6. Multi-Tenant Authorization Model

- Authority Provider model has been replaced by a Multi-tenant Authorization model. Access privileges are now defined by policies that can be applied system-wide or to individual components. Details can be found in [Multi-tenant Authorization](#) in *Hortonworks DataFlow Administration*.

- The system properties `nifi.authority.provider.configuration.file` and `nifi.security.user.authority.provider` have been replaced by `nifi.authorizer.configuration.file` and `nifi.security.user.authorizer`, respectively. Details on configuration can be found in [Authorizer Configuration](#) in *Hortonworks DataFlow Administration*.
- You can convert NiFi 0.7.0 authorized users and roles to the new authorization model. An existing `authorized-users.xml` file can be referenced in the `authorizers.xml` "Legacy Authorized Users File" property to automatically generate users and authorizations. Details on configuration can be found in [Authorizers.xml Setup](#) in *Hortonworks DataFlow Administration*.

1.1.7. HTTP(S) Site-to-Site

- HTTP(S) protocol is now supported in Site-to-Site as an underlying transport protocol.
- HTTP(S) protocol is enabled by default (`nifi.remote.input.http.enabled=true`). Configuration details can be found in [Site-to-Site Properties](#) in *Hortonworks DataFlow Administration*.

Of note:

- With both socket and HTTP protocols supported, `nifi.remote.input.socket.host` has been renamed to `nifi.remote.input.host`.
- `nifi.remote.input.secure` is now set to false by default.

1.1.8. Zero-Master Clustering

Master/slave clustering model has been replaced by a Zero-Master Clustering paradigm. Each node in a NiFi cluster performs the same tasks on the data, but each operates on a different set of data. A DataFlow manager can now interact with the NiFi cluster through the UI of any node.

ZooKeeper elects a single node as the Cluster Coordinator and also handles failover. All cluster nodes report heartbeat and status information to the Cluster Coordinator, which is responsible for disconnecting and connecting nodes. Additionally, every cluster has one Primary Node, also elected by ZooKeeper.

Configuration details can be found in the [Clustering Configuration](#), the [Cluster Common Properties](#), the [Cluster Node Properties](#), and the [ZooKeeper Properties](#) sections of *Hortonworks DataFlow Administration*.

Of note for your upgrade:

- NiFi Cluster Manager (NCM) configuration and properties are no longer relevant and have been removed.
- The following properties should be set on each node:
 - `nifi.web.http.port=<node port>`
 - `nifi.cluster.is.node=true`

- `nifi.cluster.node.address=<fully qualified hostname of the node>`
- `nifi.cluster.node.protocol.port=<node protocol port>`
- `nifi.state.management.embedded.zookeeper.start=true`
- `nifi.state.management.provider.cluster=zk-provider`
- `nifi.state.management.embedded.zookeeper.properties=./conf/zookeeper.properties`
- `nifi.zookeeper.connect.string=<A comma-separated list of host:port pairs to connect to ZooKeeper. For example, my-zk-server1:2181,my-zk-server2:2181,my-zk-server3:2183>`
- Coordinated dataflow selection across cluster nodes. During startup, a cluster coordinator is selected at random, and manages the distribution of the dataflow across all nodes. You should set the following to properties, to ensure that the cluster coordinator and other nodes have time to select the correct dataflow:
 - `nifi.cluster.flow.election.max.wait.time=5 mins`
 - `nifi.cluster.flow.election.max.candidates=<number of NiFi nodes in the cluster>`
- Embedded ZooKeeper setup
 - The `zookeeper.properties` file needs to be populated with a list of each node's embedded ZooKeeper server. The servers are specified in the form of `server.1`, `server.2`, to `server.n`. Each of these servers is configured as `<hostname>:<quorum port>[:<leader election port>]`. For example, `server.1=nifi-node1-hostname:2888:3888`.
 - The `zookeeper.properties` file has a property named `dataDir` which is set to `./state/zookeeper` by default. For each node, create a file named `myid` and place it in this directory. The contents of this file should be the index of the server as specified by the server.`<number>`. Configuration details can be found in [Embedded ZooKeeper Server](#) in *Hortonworks DataFlow Administration*.
- State Management – In the `state-management.xml` file, set the “Connect String” property to the same list of ZooKeeper host:port pairs used for the `nifi.zookeeper.connect.string` property value.
- Secure Clustered Environment – The identities for each node must be specified in the `authorizers.xml` file. The authorization policies required for the nodes to communicate will then be created during startup. Details on configuration can be found in [Authorizers.xml Setup](#) in *Hortonworks DataFlow Administration*.

1.1.9. Secured Zookeeper

- The username and password mechanism to provide ZooKeeper authentication is no longer supported. As a result, the “Username” and “Password” properties in the `state-management.xml` file have been removed.
- The “Access Control” property in the `state-management.xml` file is now set to “Open” by default. It should be changed to “CreatorOnly” when Zookeeper is secured via Kerberos.

1.1.10. QueryDatabaseTable Processor

The 'SQL Pre-processing Strategy' property has been replaced by the 'Database Type' property. This property sets the type of database for generating database-specific code. Property values include 'Generic' (default) and 'Oracle' (for custom SQL clauses).

1.1.11. TailFile Processor

TailFile originally stored state in a local file, then state management was added in 0.5.0 to support reading in the local state and moving it into the state manager. In 1.0.0, the auto-migration from the old state mechanism has been removed.

If upgrading from a pre-0.5.0 version of NiFi, it is suggested to upgrade to a version greater than or equal to 0.5.0 first, then go to 1.0.0 to not lose state on existing TailFile processors.

1.1.12. LDAP referral strategy 'IGNORE' bug fix

Errors occurred if the Referral Strategy was set to 'IGNORE' due to a typo in the code to accept 'INGORE' instead. The `login-identity-providers.xml` file should now be configured with the intended value of 'IGNORE' for the Referral Strategy property.

1.2. Before you Begin

Before you begin, you can perform some tasks to ensure the efficiency and success of your upgrade process.

1.2.1. Optimizing your NiFi Directory Structure for Upgrade

When you install NiFi and perform the minimum configuration requirements by editing the `nifi.properties` file, your installation directory contains several files and directories. Assuming you have installed into an `opt/` directory, the NiFi installation directory structure is similar to:

```
/
--> opt/
    --> nifi-current_version
        |--> LICENSE
        |--> NOTICE
        |--> README
        |--> bin/
        |--> conf/
        |--> docs/
        |--> lib/
        |--> work/
        |--> content_repository/
        |--> flowfile_repository/
        |--> logs/
        |--> provenance_repository/
```

To optimize the NiFi directory structure for upgrade:

1. Move the following directories out of the NiFi installation directory:

- Database repository
- All content repositories
- Flowfile repository
- NiFi log repository
- All provenance repositories

See the [Hortonworks DataFlow Administration](#) for information on options for creating these repositories to optimize performance and stability.

2. Move static items that do not change from version to version out of the conf directory. These items include:

- flow.xml.gz
- authorized-users.xml
- The templates directory



Note

NiFi 1.0.0 no longer contains `authorized-users.xml`. Instead, it contains `authorizations.xml` and `users.xml`. During upgrade, preserve `authorized-users.xml` so that NiFi 1.0.0 can use it to populate the values of the `authorizations.xml` and `users.xml` files.

A good directory structure that supports easy upgrading looks similar to:

```
/
|--> opt/
|   |--> nifi-current_version
|       |--> LICENSE
|       |--> NOTICE
|       |--> README
|       |--> bin/
|       |--> conf/
|       |--> docs/
|       |--> lib/
|       |--> work/
|   |--> Configuration_resources/
|       |--> authorized-users.xml
|       |--> flow.xml.gz
|       |--> templates/
|       |--> custom_lib/
|--> database_repository/
|--> content_repository_1/
|--> content_repository_N
|--> flowfile_repository/
|--> NiFi-logs/
```

```
|--> provenance_repository_1/  
|--> provenance_repository_N/
```



Important

Upgrading from NiFi 0.x to 1.x converts your database, content, flowfile, log, and provenance repositories automatically. Once upgraded, you can not revert the repositories back to 0.x format.

1.2.2. Preserving your custom processors

If you have written customer processors, you can preserve them during the upgrade if you store them in one centralized location.

1. Create a second library directory, called `custom_lib`.
2. Move your custom processors to this new lib directory.
3. Add a new line to the `nifi.properties` file to specify this new lib directory:

```
nifi.nar.library.directory=./lib  
nifi.nar.library.directory.custom=/opt/configuration_resources/custom_lib
```

1.2.3. Preserving customized NAR files

If you have customized any of the default NAR files, the upgrade to NiFi 1.0.0 overwrites these changes. To preserve the customization:

1. Identify and save the changes you have made to the default NAR files.
2. Perform your upgrade to NiFi 1.0.0.
3. Redo the changes you have made to the NAR files, in NiFi 1.0.0.
4. Start a dataflow.

1.3. Upgrading a single instance of NiFi

To upgrade to the NiFi 1.0.0, perform the following steps:

1. Download the [latest NiFi release](#).
2. Install the new release in `/opt`. Your directory structure should look similar to:

```
/
|--> opt/
|   |--> nifi-current_version
|   |   |--> LICENSE
|   |   |--> NOTICE
|   |   |--> README
|   |   |--> bin/
|   |   |--> conf/
|   |   |--> docs/
```

```
| | | --> lib/
| | | --> work/
|
| --> nifi-1.0.0
| | --> LICENSE
| | --> NOTICE
| | --> README
| | --> bin/
| | --> conf/
| | --> docs/
| | --> lib/
| | --> work/
|
| --> Configuration_resources/
| | --> authorized-users.xml
| | --> flow.xml.gz
| | --> templates/
| | --> custom_lib/
|
|--> database_repository/
|--> content_repository_1/
|--> content_repository_N/
|--> flowfile_repository/
|--> NiFi-logs/
|--> provenance_repository_1/
|--> provenance_repository_N/
```

3. Save the `authorized-users.xml` from your older NiFi version, so that NiFi 1.0.0 can use it to populate the values of the `authorizations.xml` and `users.xml` files.
4. Using the values already configured in the following files inside the previous NiFi conf directory, update the corresponding lines in the same files under target NiFi directory (nifi-1.0.0):
 - `bootstrap.conf`
 - `logback.xml`
 - `nifi.properties`



Note

- Ensure that you make no typos when you are configuring the various paths to your existing repos and the path to the `flow.xml.gz` file in the `nifi.properties` file in the newer release.
 - Ensure that you have the same Run As user configured in the `bootstrap.conf` file.
 - If no new lines were added to these files, you can copy them directly from an older version to a newer. If you do copy the `nifi.properties` file, update the `nifi.version` number.
5. Verify that all file and directory ownerships for your target NiFi directory match what you set on the previous version.
 6. Stop the older NiFi version.



Note

You can stop NiFi while files still exist in your dataflows.

- Before you start the upgraded NiFi nodes, add the location of your `authorized-users.xml` file to `authorizations.xml`, if you want to map your previous authorized users to the new NiFi authentication model.

```
<authorizer>
<identifier>file-provider</identifier>
<class>org.apache.nifi.authorization.FileAuthorizer</class>
<property name="Authorizations File">./conf/authorizations.xml</property>
<property name="Users File">./conf/users.xml</property>
<property name="Initial Admin Identity"></property>
<property name="Legacy Authorized Users File"></property>
```

- Start your upgraded NiFi version and immediately tail the app log.
 - Use the app log to verify that the new NiFi has fully started and begins processing data.
 - If it fails to start, you can restart the previous NiFi version while you investigate the cause.
 - NiFi is commonly configured to run as a service. Make sure that any path or links for that service are updated to point at the newly installed version's executables now.
 - Once you have verified that the new version has started, is processing data, and can be accessed via the UI, you can delete the older version.

1.4. Upgrading a NiFi cluster

The NiFi software configuration described in this procedure applies to both standalone and cluster node instances of NiFi software.

- Download the [latest NiFi release](#).
- Install the new release in `/opt`. Your directory structure should look similar to:

```
/
|--> opt/
|   |--> nifi-current_version
|   |   |--> LICENSE
|   |   |--> NOTICE
|   |   |--> README
|   |   |--> bin/
|   |   |--> conf/
|   |   |--> docs/
|   |   |--> lib/
|   |   |--> work/
|   |--> nifi-1.0.0
|   |   |--> LICENSE
|   |   |--> NOTICE
|   |   |--> README
```

```
| | | --> bin/
| | | --> conf/
| | | --> docs/
| | | --> lib/
| | | --> work/
|
| | | --> Configuration_resources/
| | | | --> authorized-users.xml
| | | | --> flow.xml.gz
| | | | --> templates/
| | | | --> custom_lib/
|
| --> database_repository/
| --> content_repository_1/
| --> content_repository_N/
| --> flowfile_repository/
| --> NiFi-logs/
| --> provenance_repository_1/
| --> provenance_repository_N/
```

3. Save the `authorized-users.xml` from your older NiFi version, so that NiFi 1.0.0 can use it to populate the values of the `authorizations.xml` and `users.xml` files.
4. Using the values already configured in the following files inside the previous NiFi conf directory, update the corresponding lines in the same files under target NiFi directory (nifi-1.0.0):
 - `bootstrap.conf`
 - `logback.xml`
 - `nifi.properties`



Note

- Ensure that you make no typos when you are configuring the various paths to your existing repos and the path to the `flow.xml.gz` file in the `nifi.properties` file in the newer release.
- Ensure that you have the same Run As user configured in the `bootstrap.conf` file.
- If no new lines were added to these files, you can copy them directly from an older version to a newer. If you do copy the `nifi.properties` file, update the `nifi.version` number.

5. Verify that all file and directory ownerships for your target NiFi directory match what you set on the previous version.
6. Stop the NCM and all nodes running the previous NiFi version.



Note

- NiFi 1.0.0 implements zero-master clustering and there is no NCM in NiFi 1.0.0.

- It is important to stop everything. You cannot upgrade and restart one node at a time. You could end up with mismatched versions of NiFi connected to your cluster NCM.
- It is OK to stop your NiFi cluster while files still exist in your dataflows.

7. Before you start the upgraded NiFi nodes, add the location of your `authorized-users.xml` file to `authorizations.xml`, if you want to map your previous authorized users to the new NiFi authentication model.

```
<authorizer>
<identifier>file-provider</identifier>
<class>org.apache.nifi.authorization.FileAuthorizer</class>
<property name="Authorizations File">./conf/authorizations.xml</property>
<property name="Users File">./conf/users.xml</property>
<property name="Initial Admin Identity"></property>
<property name="Legacy Authorized Users File"></property>
```

8. To support NiFi's new zero-master clustering model, ensure that the following properties are set on each node in your cluster:

- `nifi.web.http.port=<node port>`
- `nifi.cluster.is.node=true`
- `nifi.cluster.node.address=<fully qualified hostname of the node>`
- `nifi.cluster.node.protocol.port=<node protocol port>`
- `nifi.state.management.embedded.zookeeper.start=true`
- `nifi.state.management.provider.cluster=zk-provider`
- `nifi.state.management.embedded.zookeeper.properties=./conf/zookeeper.properties`
- `nifi.zookeeper.connect.string=<A comma-separated list of host:port pairs to connect to ZooKeeper. For example, my-zk-server1:2181,my-zk-server2:2181,my-zk-server3:2183>`

Configuration details can be found in the [Clustering Configuration](#), the [Cluster Common Properties](#), the [Cluster Node Properties](#), and the [ZooKeeper Properties](#) sections of *Hortonworks DataFlow Administration*.

9. Set the following NiFi properties to ensure that the cluster coordinator and other nodes have time to select the correct dataflow:

- `nifi.cluster.flow.election.max.wait.time=5 mins`
- `nifi.cluster.flow.election.max.candidates=<number of NiFi nodes in the cluster>`

10. Start the upgraded NiFi nodes.



Note

Check the `nifi-app.log` for errors and for any nodes that fail to join the upgraded cluster.

11.If you configured NiFi to run as a service, ensure that any path or links for that service are updated to point at the newly installed version executables.

12.Configure the `state-management.xml` and `zookeeper.properties` file on every node with the following information:

- Specify whether you want to use an embedded or external Zookeeper server.
- If you are using an embedded server, identify the nodes on which you want to run the Zookper server. For these nodes, be sure to also edit the state management porition of the `nifi.properties` file.

See the [Hortonworks DataFlow Administration](#) for additional information on state management and using the Zookeeper server.

13.If you want to start using LDAP user authentication, configure the `login-identity-providers.xml` file, and update the `nifi.properties` file.

14.If you want to setup NiFi JVM lifecycle event notifications, configure the `bootstrap-notification.services.xml` file and update the relevant settings in the `bootstrap.conf` file.

15.Once you have verified that the new version has started, is processing data, and can be accessed via the UI, you can delete the older version.