Apache NiFi 3

Apache NiFi Toolkit

Date of Publish: 2018-08-13

http://docs.hortonworks.com

Contents

TLS Generation Toolkit

In order to facilitate the secure setup of NiFi, you can use the tls-toolkit command line utility to automatically generate the required keystores, truststore, and relevant configuration files. This is especially useful for securing multiple NiFi nodes, which can be a tedious and error-prone process.

> **Note:** JKS keystores and truststores are recommended for NiFi. This tool allows the specification of other keystore types on the command line but will ignore a type of PKCS12 for use as the truststore because that format has some compatibility issues between BouncyCastle and Oracle implementations.

The tls-toolkit command line tool has two primary modes of operation:

1. Standalone - generates the certificate authority, keystores, truststores, and nifi.properties files in one command.
2. Client/Server mode - uses a Certificate Authority Server that accepts Certificate Signing Requests from clients, signs them, and sends the resulting certificates back. Both client and server validate the other's identity through a shared secret.

Standalone

Standalone mode is invoked by running ./bin/tls-toolkit.sh standalone -h which prints the usage information along with descriptions of options that can be specified.

You can use the following command line options with the tls-toolkit in standalone mode:

- -a,--keyAlgorithm <arg> Algorithm to use for generated keys (default: RSA)
- -B,--clientCertPassword <arg> Password for client certificate. Must either be one value or one for each client DN (auto-generate if not specified)
- -c,--certificateAuthorityHostname <arg> Hostname of NiFi Certificate Authority (default: localhost)
- -C,--clientCertDn <arg> Generate client certificate suitable for use in browser with specified DN (Can be specified multiple times)
- -d,--days <arg> Number of days issued certificate should be valid for (default: 1095)
- -f,--nifiPropertiesFile <arg> Base nifi.properties file to update (Embedded file identical to the one in a default NiFi install will be used if not specified)
- -g,--differentKeyAndKeystorePasswords Use different generated password for the key and the keystore
- -G,--globalPortSequence <arg> Use sequential ports that are calculated for all hosts according to the provided hostname expressions (Can be specified multiple times, MUST BE SAME FROM RUN TO RUN)
- -h,--help Print help and exit
- -k,--keySize <arg> Number of bits for generated keys (default: 2048)
- -K,--keyPassword <arg> Key password to use. Must either be one value or one for each host (auto-generate if not specified)
- -n,--hostnames <arg> Comma separated list of hostnames
- --nifiDnPrefix <arg> String to prepend to hostname(s) when determining DN (default: CN=)
- --nifiDnSuffix <arg> String to append to hostname(s) when determining DN (default: , OU=NIFI)
- -o,--outputDirectory <arg> The directory to output keystores, truststore, config files (default: ../bin)
- -O,--isOverwrite Overwrite existing host output
- -P,--trustStorePassword <arg> Keystore password to use. Must either be one value or one for each host (auto-generate if not specified)
- -s,--signingAlgorithm <arg> Algorithm to use for signing certificates (default: SHA256WITHRSA)
- -S,--keyStorePassword <arg> Keystore password to use. Must either be one value or one for each host (auto-generate if not specified)
- --subjectAlternativeNames <arg> Comma-separated list of domains to use as Subject Alternative Names in the certificate

- -T,--keyStoreType <arg> The type of keystores to generate (default: jks)

Hostname Patterns:

- Square brackets can be used in order to easily specify a range of hostnames. Example: [01-20]
- Parentheses can be used in order to specify that more than one NiFi instance will run on the given host(s). Example: (5)

Examples:

Create 4 sets of keystore, truststore, nifi.properties for localhost along with a client certificate with the given DN:

```
bin/tls-toolkit.sh standalone -n 'localhost(4)' -C 'CN=username,OU=NIFI'
```

Create keystore, truststore, nifi.properties for 10 NiFi hostnames in each of 4 subdomains:

```
bin/tls-toolkit.sh standalone -n 'nifi[01-10].subdomain[1-4].domain'
```

Create 2 sets of keystore, truststore, nifi.properties for 10 NiFi hostnames in each of 4 subdomains along with a client certificate with the given DN:

```
bin/tls-toolkit.sh standalone -n 'nifi[01-10].subdomain[1-4].domain(2)' -C
 'CN=username,OU=NIFI'
```

Client/Server

Client/Server mode relies on a long-running Certificate Authority (CA) to issue certificates. The CA can be stopped when you're not bringing nodes online.

Server

The CA server is invoked by running ./bin/tls-toolkit.sh server -h which prints the usage information along with descriptions of options that can be specified.

You can use the following command line options with the tls-toolkit in server mode:

- -a,--keyAlgorithm <arg> Algorithm to use for generated keys (default: RSA)
- --configJsonIn <arg> The place to read configuration info from (defaults to the value of configJson), implies useConfigJson if set (default: configJson value)
- -d,--days <arg> Number of days issued certificate should be valid for (default: 1095)
- -D,--dn <arg> The dn to use for the CA certificate (default: CN=YOUR_CA_HOSTNAME,OU=NIFI)
- -f,--configJson <arg> The place to write configuration info (default: config.json)
- -F,--useConfigJson Flag specifying that all configuration is read from configJson to facilitate automated use (otherwise configJson will only be written to)
- -g,--differentKeyAndKeystorePasswords Use different generated password for the key and the keystore
- -h,--help Print help and exit
- -k,--keySize <arg> Number of bits for generated keys (default: 2048)
- -p,--PORT <arg> The port for the Certificate Authority to listen on (default: 8443)
- -s,--signingAlgorithm <arg> Algorithm to use for signing certificates (default: SHA256WITHRSA)
- -T,--keyStoreType <arg> The type of keystores to generate (default: jks)
- -t,--token <arg> The token to use to prevent MITM (required and must be same as one used by clients)

Client

The client can be used to request new Certificates from the CA. The client utility generates a keypair and Certificate Signing Request (CSR) and sends the CSR to the Certificate Authority. The client is invoked by running ./bin/tls-toolkit.sh client -h which prints the usage information along with descriptions of options that can be specified.

You can use the following command line options with the tls-toolkit in client mode:

- -a,--keyAlgorithm <arg> Algorithm to use for generated keys (default: RSA)
- -c,--certificateAuthorityHostname <arg> Hostname of NiFi Certificate Authority (default: localhost)
- -C,--certificateDirectory <arg> The directory to write the CA certificate (default: .)
- --configJsonIn <arg> The place to read configuration info from, implies useConfigJson if set (default: configJson value)
- -D,--dn <arg> The DN to use for the client certificate (default: CN=<localhost name>,OU=NIFI) (this is auto-populated by the tool)
- -f,--configJson <arg> The place to write configuration info (default: config.json)
- -F,--useConfigJson Flag specifying that all configuration is read from configJson to facilitate automated use (otherwise configJson will only be written to)
- -g,--differentKeyAndKeystorePasswords Use different generated password for the key and the keystore
- -h,--help Print help and exit
- -k,--keySize <arg> Number of bits for generated keys (default: 2048)
- -p,--PORT <arg> The port to use to communicate with the Certificate Authority (default: 8443)
- --subjectAlternativeNames <arg> Comma-separated list of domains to use as Subject Alternative Names in the certificate
- -T,--keyStoreType <arg> The type of keystores to generate (default: jks)
- -t,--token <arg> The token to use to prevent MITM (required and must be same as one used by CA)

After running the client you will have the CA's certificate, a keystore, a truststore, and a config.json with information about them as well as their passwords.

For a client certificate that can be easily imported into the browser, specify: -T PKCS12

Encrypt-Config Tool

The encrypt-config command line tool (invoked as ./bin/encrypt-config.sh or bin\encrypt-config.bat) reads from a 'nifi.properties' file with plaintext sensitive configuration values, prompts for a master password or raw hexadecimal key, and encrypts each value. It replaces the plain values with the protected value in the same file, or writes to a new 'nifi.properties' file if specified.

The default encryption algorithm utilized is AES/GCM 128/256-bit. 128-bit is used if the JCE Unlimited Strength Cryptographic Jurisdiction Policy files are not installed, and 256-bit is used if they are installed.

You can use the following command line options with the encrypt-config tool:

- -h,--help Prints this usage message
- -v,--verbose Sets verbose mode (default false)
- -n,--niFiProperties <arg> The nifi.properties file containing unprotected config values (will be overwritten)
- -l,--loginIdentityProviders <arg> The login-identity-providers.xml file containing unprotected config values (will be overwritten)
- -a,--authorizers <arg> The authorizers.xml file containing unprotected config values (will be overwritten)
- -f,--flowXml <arg> The flow.xml.gz file currently protected with old password (will be overwritten)
- -b,--bootstrapConf <arg> The bootstrap.conf file to persist master key
- -o,--outputNiFiProperties <arg> The destination nifi.properties file containing protected config values (will not modify input nifi.properties)
- -i,--outputLoginIdentityProviders <arg> The destination login-identity-providers.xml file containing protected config values (will not modify input login-identity-providers.xml)

- -u,--outputAuthorizers <arg> The destination authorizers.xml file containing protected config values (will not modify input authorizers.xml)
- -g,--outputFlowXml <arg> The destination flow.xml.gz file containing protected config values (will not modify input flow.xml.gz)
- -k,--key <arg> The raw hexadecimal key to use to encrypt the sensitive properties
- -e,--oldKey <arg> The old raw hexadecimal key to use during key migration
- -p,--password <arg> The password from which to derive the key to use to encrypt the sensitive properties
- -w,--oldPassword <arg> The old password from which to derive the key during migration
- -r,--useRawKey If provided, the secure console will prompt for the raw key value in hexadecimal form
- -m,--migrate If provided, the nifi.properties and/or login-identity-providers.xml sensitive properties will be re-encrypted with a new key
- -x,--encryptFlowXmlOnly If provided, the properties in flow.xml.gz will be re-encrypted with a new key but the nifi.properties and/or login-identity-providers.xml files will not be modified
- -s,--propsKey <arg> The password or key to use to encrypt the sensitive processor properties in flow.xml.gz
- -A,--newFlowAlgorithm <arg> The algorithm to use to encrypt the sensitive processor properties in flow.xml.gz
- -P,--newFlowProvider <arg> The security provider to use to encrypt the sensitive processor properties in flow.xml.gz

As an example of how the tool works, assume that you have installed the tool on a machine supporting 256-bit encryption and with the following existing values in the 'nifi.properties' file:

```
# security properties #
nifi.sensitive.props.key=thisIsABadSensitiveKeyPassword
nifi.sensitive.props.algorithm=PBEWITHMD5AND256BITAES-CBC-OPENSSL
nifi.sensitive.props.provider=BC
nifi.sensitive.props.additional.keys=

nifi.security.keystore=/path/to/keystore.jks
nifi.security.keystoreType=JKS
nifi.security.keystorePasswd=thisIsABadKeystorePassword
nifi.security.keyPasswd=thisIsABadKeyPassword
nifi.security.truststore=
nifi.security.truststoreType=
nifi.security.truststorePasswd=
```

Enter the following arguments when using the tool:

```
encrypt-config.sh
-b bootstrap.conf
-k 0123456789ABCDEFFEDCBA98765432100123456789ABCDEFFEDCBA9876543210
-n nifi.properties
```

As a result, the 'nifi.properties' file is overwritten with protected properties and sibling encryption identifiers (aes/gcm/256, the currently supported algorithm):

```
# security properties #
nifi.sensitive.props.key=n2z+tTTbHuZ4V4V2||uWhdasyDXD4ZG2lMAes/
vqh6u4vaz4xgL4aEbF4Y/dXevqk3ulRcOwf1vc4RDQ==
nifi.sensitive.props.key.protected=aes/gcm/256
nifi.sensitive.props.algorithm=PBEWITHMD5AND256BITAES-CBC-OPENSSL
nifi.sensitive.props.provider=BC
nifi.sensitive.props.additional.keys=

nifi.security.keystore=/path/to/keystore.jks
nifi.security.keystoreType=JKS
nifi.security.keystorePasswd=oBjT92hIGRElIGOh||MZ6uYuWNBrOA6usq/
Jt3DaD2e4otNirZDytac/w/KFe0HOkrJR03vcbo
nifi.security.keystorePasswd.protected=aes/gcm/256
```

```
nifi.security.keyPasswd=ac/BaE35SL/esLiJ||
+ULRvRLYdIDA2VqpE0eQXDEMjaLBMG2kbKOdOwBk/hGebDKlVg==
nifi.security.keyPasswd.protected=aes/gcm/256
nifi.security.truststore=
nifi.security.truststoreType=
nifi.security.truststorePasswd=
```

Additionally, the 'bootstrap.conf' file is updated with the encryption key as follows:

```
# Master key in hexadecimal format for encrypted sensitive configuration
 values
nifi.bootstrap.sensitive.key=0123456789ABCDEFFEDCBA98765432100123456789ABCDEFFEDCBA98765
```

Sensitive configuration values are encrypted by the tool by default, however you can encrypt any additional properties, if desired. To encrypt additional properties, specify them as comma-separated values in the nifi.sensitive.props.additional.keys property.

If the 'nifi.properties' file already has valid protected values, those property values are not modified by the tool.

When applied to 'login-identity-providers.xml' and 'authorizers.xml', the property elements are updated with an encryption attribute:

Example of protected login-identity-providers.xml:

```
   <!-- LDAP Provider -->
   <provider>
       <identifier>ldap-provider</identifier>
       <class>org.apache.nifi.ldap.LdapProvider</class>
       <property name="Authentication Strategy">START_TLS</property>
       <property name="Manager DN">someuser</property>
       <property name="Manager Password" encryption="aes/
gcm/128">q4r7WIgN0MaxdAKM||SGgdCTPGSFEcuH4RraMYEdeyVbOx93abdWTVSWvh1w+klA</
property>
       <property name="TLS - Keystore"></property>
       <property name="TLS - Keystore Password" encryption="aes/
gcm/128">Uah59TWX+Ru5GY5p||B44RT/LJtC08QWA5ehQf01JxIpf0qSJUzug25UwkF5a50g</
property>
       <property name="TLS - Keystore Type"></property>
       ...
   </provider>
```

Example of protected authorizers.xml:

```
   <!-- LDAP User Group Provider -->
   <userGroupProvider>
       <identifier>ldap-user-group-provider</identifier>
       <class>org.apache.nifi.ldap.tenants.LdapUserGroupProvider</class>
       <property name="Authentication Strategy">START_TLS</property>
       <property name="Manager DN">someuser</property>
       <property name="Manager Password" encryption="aes/
gcm/128">q4r7WIgN0MaxdAKM||SGgdCTPGSFEcuH4RraMYEdeyVbOx93abdWTVSWvh1w+klA</
property>
       <property name="TLS - Keystore"></property>
       <property name="TLS - Keystore Password" encryption="aes/
gcm/128">Uah59TWX+Ru5GY5p||B44RT/LJtC08QWA5ehQf01JxIpf0qSJUzug25UwkF5a50g</
property>
       <property name="TLS - Keystore Type"></property>
       ...
   </userGroupProvider>
---
```

Administrative Tools

The admin toolkit contains command line utilities for administrators to support NiFi maintenance in standalone and clustered environments. These utilities include:

- Notify - The notification tool allows administrators to send bulletins to the NiFi UI using the command line.
- Node Manager - The node manager tool allows administrators to perform a status check on a node as well as to connect, disconnect, or remove nodes that are part of a cluster.
- File Manager - The file manager tool allows administrators to backup, install or restore a NiFi installation from backup.

The admin toolkit is bundled with the nifi-toolkit and can be executed with scripts found in the bin folder.

Prerequisites for Running Admin Toolkit in a Secure Environment

For secured nodes and clusters, two policies should be configured in advance:

- Access the controller - A user that will have access to these utilities should be authorized in NiFi by creating an "access the controller" policy (/controller) with both view and modify rights.
- Proxy user request - If not previously set node's identity (the DN value of the node's certificate) should be authorized to proxy requests on behalf of a user

When executing either the notify or node manager tools in a secured environment the proxyDN flag option should be used in order to properly identify the user that was authorized to execute these commands. In non-secure environments, or if running the status operation on the Node Manager tool, the flag is ignored.

Notify

Notify allows administrators to send messages as bulletins to NiFi. Notify is supported on NiFi version 1.2.0 and higher. The notification tool is also available in a notify.bat file for use on Windows machines.

To send notifications:

```
notify.sh -d {$NIFI_HOME} -b {nifi bootstrap file path} -m {message} [-l
 {level}] [-v]
```

To show help:

```
notify.sh -h
```

The following are available options:

- -b,--bootstrapConf <arg> Existing Bootstrap Configuration file (required)
- -d,--nifiInstallDir <arg> NiFi Root Folder (required)
- -h,--help Help Text (optional)
- -l,--level <arg> Status level of bulletin - INFO, WARN, ERROR
- -m,--message <arg> Bulletin message (required)
- -p,--proxyDN <arg> Proxy or User DN (required for secured nodes)
- -v,--verbose Verbose messaging (optional)

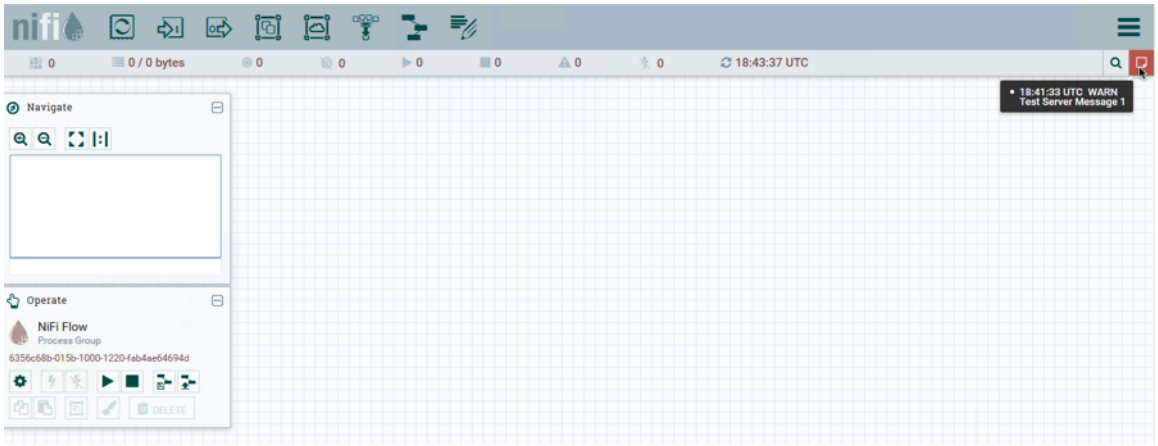
Example usage on Linux:

```
./notify.sh -d /usr/nifi/nifi_current -b /usr/nifi/nifi_current/conf/
bootstrap.conf -m "Test Message Server 1" -l "WARN" -p "ydavis@nifi" -v
```

Example usage on Windows:

```
notify.bat -v -d "C:\\Program Files\\nifi\\nifi-1.2.0-SNAPSHOT" -b "C:\
\Program Files\\nifi\\nifi-1.2.0-SNAPSHOT\\conf\\bootstrap.conf" -m "Test
 Message Server 1" -v
```

Executing the above command line should result in a bulletin appearing in NiFi:



Node Manager

Node manager supports connecting, disconnecting and removing a node when in a cluster (an error message displays if the node is not part of a cluster) as well as obtaining the status of a node. When nodes are disconnected from a cluster and need to be connected or removed, a list of urls of connected nodes should be provided to send the required command to the active cluster. Node Manager supports NiFi version 1.0.0 and higher. Node Manager is also available in 'node-manager.bat' file for use on Windows machines.

To connect, disconnect, or remove a node from a cluster:

```
node-manager.sh -d {$NIFI_HOME} -b { nifi bootstrap file path}
-o {remove|disconnect|connect|status} [-u {url list}] [-p {proxy name}] [-v]
```

To show help:

```
node-manager.sh -h
```

The following are available options:

- -b,--bootstrapConf <arg> Existing Bootstrap Configuration file (required)
- -d,--nifiInstallDir <arg> NiFi Root Folder (required)
- -h,--help Help Text (optional)
- -o, --operation <arg> Operations supported: status, connect (cluster), disconnect (cluster), remove (cluster)
- -p,--proxyDN <arg> Proxy or User DN (required for secured nodes doing connect, disconnect and remove operations)
- -u,--clusterUrls <arg> Comma delimited list of active urls for cluster (optional). Not required for disconnecting a node yet will be needed when connecting or removing from a cluster
- -v,--verbose Verbose messaging (optional)

Example usage on Linux:

```
# disconnect without cluster url list
./node-manager.sh
-d /usr/nifi/nifi_current
-b /usr/nifi/nifi_current/conf/bootstrap.conf
-o disconnect
-p ydavis@nifi
-v
```

```
#with url list
./node-manager.sh
-d /usr/nifi/nifi_current
-b /usr/nifi/nifi_current/conf/bootstrap.conf
-o connect
-u 'http://nifi-server-1:8080,http://nifi-server-2:8080'
-v
```

Example usage on Windows:

```
node-manager.bat
-d "C:\\Program Files\\nifi\\nifi-1.2.0-SNAPSHOT"
-b "C:\\Program Files\\nifi\\nifi-1.2.0-SNAPSHOT\\conf\\bootstrap.conf"
-o disconnect
-v
```

Expected behavior

Status:

To obtain information on UI availability of a node, the status operation can be used to determine if the node is running. If the -u (clusterUrls) option is not provided the current node url is checked otherwise the urls provided will be checked.

Disconnect:

When a node is disconnected from the cluster, the node itself should appear as disconnected and the cluster should have a bulletin indicating the disconnect request was received. The cluster should also show n-1/n nodes available in the cluster. For example, if 1 node is disconnected from a 3-node cluster, then 2 of 3 nodes should show on the remaining nodes in the cluster. Changes to the flow should not be allowed on the cluster with a disconnected node.

Connect:

When the connect command is executed to reconnect a node to a cluster, upon completion the node itself should show that it has rejoined the cluster by showing n/n nodes. Previously it would have shown Disconnected. Other nodes in the cluster should receive a bulletin of the connect request and also show n/n nodes allowing for changes to be allowed to the flow.

Remove:

When the remove command is executed the node should show as disconnected from a cluster. The nodes remaining in the cluster should show n-1/n-1 nodes. For example, if 1 node is removed from a 3-node cluster, then the remaining 2 nodes should show 2 of 2 nodes). The cluster should allow a flow to be adjusted. The removed node can rejoin the cluster if restarted and the flow for the cluster has not changed. If the flow was changed, the flow template of the removed node should be deleted before restarting the node to allow it to obtain the cluster flow (otherwise an uninheritable flow file exception may occur).

File Manager

The File Manager utility allows system administrators to take a backup of an existing NiFi installation, install a new version of NiFi in a designated location (while migrating any previous configuration settings) or restore an installation from a previous backup. File Manager supports NiFi version 1.0.0 and higher and is available in 'file-manager.bat' file for use on Windows machines.

To show help:

```
file-manager.sh -h
```

The following are available options:

- -b,--backupDir <arg> Backup NiFi Directory (used with backup or restore operation)
- -c,--nifiCurrentDir <arg> Current NiFi Installation Directory (used optionally with install or restore operation)
- -d,--nifiInstallDir <arg> NiFi Installation Directory (used with install or restore operation)
- -h,--help Print help info (optional)
- -i,--installFile <arg> NiFi Install File (used with install operation)
- -m,--moveRepositories Allow repositories to be moved to new/restored nifi directory from existing installation, if available (used optionally with install or restore operation)
- -o,--operation <arg> File operation (install | backup | restore)
- -r,--nifiRollbackDir <arg> NiFi Installation Directory (used with install or restore operation)
- -t,--bootstrapConf <arg> Current NiFi Bootstrap Configuration File (used optionally)
- -v,--verbose Verbose messaging (optional)
- -x,--overwriteConfigs Overwrite existing configuration directory with upgrade changes (used optionally with install or restore operation)

Example usage on Linux:

```
# backup NiFi installation
# option -t may be provided to ensure backup of external boostrap.conf file
./file-manager.sh
-o backup
-b /tmp/nifi_bak
-c /usr/nifi_old
-v
```

```
# install NiFi using compressed tar file into /usr/nifi directory (should
 install as /usr/nifi/nifi-1.3.0).
# migrate existing configurations with location determined by external
 bootstrap.conf and move over repositories from nifi_old
# options -t and -c should both be provided if migration of configurations,
 state and repositories are required
./file-manager.sh
-o install
-i nifi-1.3.0.tar.gz
-d /usr/nifi
-c /usr/nifi/nifi_old
-t /usr/nifi/old_conf/bootstrap.conf
-v
-m
```

```
# restore NiFi installation from backup directory and move back repositories
# option -t may be provided to ensure bootstrap.conf is restored to the file
 path provided, otherwise it is placed in the
# default directory under the rollback path (e.g. /usr/nifi_old/conf)
./file-manager.sh
-o restore
-b /tmp/nifi_bak
-r /usr/nifi_old
-c /usr/nifi
```

```
-m
-v
```

Expected Behavior

Backup:

During the backup operation a backup directory is created in a designated location for an existing NiFi installation. Backups will capture all critical files (including any internal or external configurations, libraries, scripts and documents) however it excludes backing up repositories and logs due to potential size. If configuration/library files are external from the existing installation folder the backup operation will capture those as well.

Install:

During the install operation File Manager will perform installation using the designated NiFi binary file (either tar.gz or zip file) to create a new installation or migrate an existing nifi installation to a new one. Installation can optionally move repositories (if located within the configuration folder of the current installation) to the new installation as well as migrate configuration files to the newer installation.

Restore:

The restore operation allows an existing installation to revert back to a previous installation. Using an existing backup directory (created from the backup operation) the FileManager utility will restore libraries, scripts and documents as well as revert to previous configurations. NOTE: If repositories were changed due to the installation of a newer version of NiFi these may no longer be compatible during restore. In that scenario exclude the -m option to ensure new repositories will be created or, if repositories live outside of the NiFi directory, remove them so they can be recreated on startup after restore.

ZooKeeper Migrator

You can use the NiFi ZooKeeper Migrator to perform the following tasks:

- Moving ZooKeeper information from one ZooKeeper cluster to another
- Migrating ZooKeeper node ownership

For example, you may want to use the ZooKeeper Migrator when you are:

- Upgrading from NiFi 0.x to NiFi 1.x in which embedded ZooKeepers are used
- Migrating from an embedded ZooKeeper in NiFi 0.x or 1.x to an external ZooKeeper
- Upgrading from NiFi 0.x with an external ZooKeeper to NiFi 1.x with the same external ZooKeeper
- Migrating from an external ZooKeeper to an embedded ZooKeeper in NiFi 1.x

zk-migrator.sh Command Line Parameters

You can use the following command line options with the ZooKeeper Migrator:

- -a,--auth <username:password> Allows the specification of a username and password for authentication with ZooKeeper. This option is mutually exclusive with the -k,--krb-conf option.
- -f,--file <filename> The file used for ZooKeeper data serialized as JSON. When used with the -r,--receive option, data read from ZooKeeper will be stored in the given filename. When used with the -s,--send option, the data in the file will be sent to ZooKeeper.
- -h,--help Prints help, displays available parameters with descriptions
- --ignore-source Allows the ZooKeeper Migrator to write to the ZooKeeper and path from which the data was obtained.

- -k,--krb-conf <jaas-filename> Allows the specification of a JAAS configuration file to allow authentication with a ZooKeeper configured to use Kerberos. This option is mutually exclusive with the -a,--auth option.
- -r,--receive Receives data from ZooKeeper and writes to the given filename (if the -f,--file option is provided) or standard output. The data received will contain the full path to each node read from ZooKeeper. This option is mutually exclusive with the -s,--send option.
- -s,--send Sends data to ZooKeeper that is read from the given filename (if the -f,--file option is provided) or standard input. The paths for each node in the data being sent to ZooKeeper are absolute paths, and will be stored in ZooKeeper under the path portion of the -z,--zookeeper argument. Typically, the path portion of the argument can be omitted, which will store the nodes at their absolute paths. This option is mutually exclusive with the -r,--receive option.
- --use-existing-acl Allows the Zookeeper Migrator to write ACL values retrieved from the source Zookeeper server to destination server. Default action will apply Open rights for unsecured destinations or Creator Only rights for secured destinations.
- -z,--zookeeper <zookeeper-endpoint> The ZooKeeper server(s) to use, specified by a connect string, comprised of one or more comma-separated host:port pairs followed by a path, in the format of host:port[,host2:port… ,hostn:port]/znode/path.

Migrating Between Source and Destination ZooKeepers

Before you begin, confirm that:

- You have installed the destination ZooKeeper cluster.
- You have installed and configured a NiFi cluster to use the destination ZooKeeper cluster.
- If you are migrating ZooKeepers due to upgrading NiFi from 0.x to 1.x,, you have already followed appropriate NiFi upgrade steps.
- You have configured Kerberos as needed.
- You have not started processing any dataflow (to avoid duplicate data processing).
- If one of the ZooKeeper clusters you are using is configured with Kerberos, you are running the ZooKeeper Migrator from a host that has access to NiFi's ZooKeeper client jaas configuration file.

ZooKeeper Migration Steps

1. Collect the following information:

Required Information	Description
Source ZooKeeper hostname (sourceHostname)	The hostname must be one of the hosts running in the ZooKeeper ensemble, which can be found in <NiFi installation dir>/conf/zookeeper.properties. Any of the hostnames declared in the server.N properties can be used.
Destination ZooKeeper hostname (destinationHostname)	The hostname must be one of the hosts running in the ZooKeeper ensemble, which can be found in <NiFi installation dir>/conf/zookeeper.properties. Any of the hostnames declared in the server.N properties can be used.
Source ZooKeeper port (sourceClientPort)	This can be found in zookeeper.properties of the <NiFi installation dir>/conf/zookeeper.properties. The port is specified in the clientPort property.
Destination ZooKeeper port (destinationClientPort)	This can be found in zookeeper.properties of the <NiFi installation dir>/conf/zookeeper.properties. The port is specified in the clientPort property.
Export data path	Determine the path that will store a json file containing the export of data from ZooKeeper. It must be readable and writable by the user running the zk-migrator tool.

Source ZooKeeper Authentication Information	This information is in <NiFi installation dir>/conf/state-management.xml. For NiFi 0.x, if Creator Only is specified in state-management.xml, you need to supply authentication information using the -a,--auth argument with the values from the Username and Password properties in state-management.xml. For NiFi 1.x, supply authentication information using the -k,--krb-conf argument. + If the state-management.xml specifies Open, no authentication is required.
Destination ZooKeeper Authentication Information	This information is in <NiFi installation dir>/conf/state-management.xml. For NiFi 0.x, if Creator Only is specified in state-management.xml, you need to supply authentication information using the -a,--auth argument with the values from the Username and Password properties in state-management.xml. For NiFi 1.x, supply authentication information using the -k,--krb-conf argument. + If the state-management.xml specifies Open, no authentication is required.
Root path to which NiFi writes data in Source ZooKeeper (sourceRootPath)	This information can be found in <NiFi installation dir>/conf/state-management.xml under the Root Node property in the cluster-provider element. (default: /nifi)
Root path to which NiFi writes data in Destination ZooKeeper (destinationRootPath)	This information can be found in <NiFi installation dir>/conf/state-management.xml under the Root Node property in the cluster-provider element.

2. Stop all processors in the NiFi flow. If you are migrating between two NiFi installations, the flows on both must be stopped.
3. Export the NiFi component data from the source ZooKeeper. The following command reads from the specified ZooKeeper running on the given hostname:port, using the provided path to the data, and authenticates with ZooKeeper using the given username and password. The data read from ZooKeeper is written to the file provided.
   - For NiFi 0.x
     - For an open ZooKeeper:
       - zk-migrator.sh -r -z sourceHostname:sourceClientPort/sourceRootPath/components -f /path/to/export/zk-source-data.json
     - For a ZooKeeper using username:password for authentication:
       - zk-migrator.sh -r -z sourceHostname:sourceClientPort/sourceRootPath/components -a <username:password> -f /path/to/export/zk-source-data.json
   - For NiFi 1.x
     - For an open ZooKeeper:
       - zk-migrator.sh -r -z sourceHostname:sourceClientPort/sourceRootPath/components -f /path/to/export/zk-source-data.json
     - For a ZooKeeper using Kerberos for authentication:
       - zk-migrator.sh -r -z sourceHostname:sourceClientPort/sourceRootPath/components -k /path/to/jaasconfig/jaas-config.conf -f /path/to/export/zk-source-data.json
4. (Optional) If you have used the new NiFi installation to do any processing, you can also export its ZooKeeper data as a backup prior to performing the migration.
   - For an open ZooKeeper:
     - zk-migrator.sh -r -z destinationHostname:destinationClientPort/destinationRootPath/components -f /path/to/export/zk-destination-backup-data.json
   - For a ZooKeeper using Kerberos for authentication:
     - zk-migrator.sh -r -z destinationHostname:destinationClientPort/destinationRootPath/components -k /path/to/jaasconfig/jaas-config.conf -f /path/to/export/zk-destination-backup-data.json
5. Migrate the ZooKeeper data to the destination ZooKeeper. If the source and destination ZooKeepers are the same, the --ignore-source option can be added to the following examples.
   - For an open ZooKeeper:

    - zk-migrator.sh -s -z destinationHostname:destinationClientPort/destinationRootPath/components -f /path/to/export/zk-source-data.json
  - For a ZooKeeper using Kerberos for authentication:
    - zk-migrator.sh -s -z destinationHostname:destinationClientPort/destinationRootPath/components -k /path/to/jaasconfig/jaas-config.conf -f /path/to/export/zk-source-data.json
6. Once the migration has completed successfully, start the processors in the NiFi flow. Processing should continue from the point at which it was stopped when the NiFi flow was stopped.