CEM MiNiFi C++ Agent 1.25.05

MiNiFi C++ Agent Installation

Date published: 2020-10-14
Date modified: 2025-06-30

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Installing your MiNiFi C++ agent..... 4
Before you begin installing MiNiFi C++ agent....4
System requirements for MiNiFi C++....4
Before you begin MiNiFi agent installation on Windows....5
Installing MiNiFi C++ agent....5
Installing C++ agent on Windows from command line....5
Installing C++ agent on Windows using MSI....6
MiNiFi C++ agent configurations....11
Configuring your MiNiFi C++ agent....11
Configuring communication with secured EFM....13
Configuring MiNiFi C++ repositories....13
Encrypting sensitive data....16
Integrating with the Windows certificate store....18
Starting MiNiFi C++ Agent....19
Troubleshooting MiNiFi C++ agent....20

Installing your MiNiFi C++ agent

Learn how to install the MiNiFi C++ agent. You need to install and configure the MiNiFi C++ agent and then start it. You can also do the same on Windows.

Before you begin installing MiNiFi C++ agent

To start using the MiNiFi C++ agent, you need to install it. Learn how to obtain the MiNiFi C++ software bit, install the agent, and configure it. Also learn the system requirements to do so.

System requirements for MiNiFi C++

Carefully review the system requirements for MiNiFi C++ before you begin installing it.

Operating system support in CEM MiNiFi C++ Agent 1.24.05 release

Operating System	Version
RHEL and compatible	8.x, 9.x
Debian	11, 12
Ubuntu	20.04, 22.04, 24.04
Windows	8, 10, Server 2012, Server 2012 R2, Server 2016, Server 2019

Operating system support in CEM MiNiFi C++ Agent 1.21.10 - 1.24.04 releases

Operating System	Version
RHEL/CentOS	7.x, 8.x
Debian	10, 11
Ubuntu	18.04, 20.04
Windows	8, 10, Server 2012, Server 2012 R2, Server 2016, Server 2019

Operating system support in CEM MiNiFi C++ Agent 1.21.06 and 1.21.08 releases

Operating System	Version
RHEL/CentOS	7.x, 8.x
Debian	10
Ubuntu	18.04, 20.04
Windows	8, 10, Server 2012, Server 2012 R2, Server 2016, Server 2019

Operating system support in releases before CEM MiNiFi C++ Agent 1.21.06

Operating System	Version
RHEL/CentOS	7.x, 8.x
Debian	9
Ubuntu	16.04, 18.04
Windows	8, 10, Server 2012, Server 2012 R2, Server 2016, Server 2019

Before you begin MiNiFi agent installation on Windows

You must go through the prerequisites before you begin installation of MiNiFi agent on Windows.

Requirements

Apache NiFi MiNiFi C++ is built on Window Server 2016, 2019, and Windows 10 operating systems. Since the project is CMake focused, Cloudera recommends building the Microsoft Installer (MSI) or Windows Installer through the ms_build.bat script. In order to build the MSI, please install the WiX Toolset.

The project previously required OpenSSL to be installed. If you follow Cloudera build procedures, you do not need to install that dependency. Further, any MSI distributable requires that systems install the Visual Studio 2010 redistributables.

Building through the build script

The preferred way of building the project is through the win_build_vs.bat script found in the root source folder.

You must supply a single command, the build directory. Typically you create a directory with CMake and build the MSI in that directory referencing your CMake tree. Simply supply the win_build_vs.bat an argument like build as the name of your build directory and it will create this directory building the project within it. Alternatively, you can use the win_build_vs.bat build /K /T /P command to build Kafka, skip tests, and build the CMake at the same time.

If WiX Toolset is installed, cpack creates your MSI in the chosen build directory.

You can also build a 64 bit MSI by adding the /64option. To do so, you require the 64-bit version of the Visual Studio redistributables mentioned in system requirements.

Note: The 64-bit MSI will not run on 32-bit Microsoft Windows platforms. However, the 32-bit MSI will work across distributions.

Installing MiNiFi C++ agent

To install the MiNiFi C++ agent, you need to download the software for the agent and extract it.

Procedure

1. Download the MiNiFi C++ agent software by selecting the appropriate tar.gz or zip file.

```
wget {cpp.tar.gz}
```

2. Extract it to your preferred home directory.
3. Run the installation script.

```
bin/minifi.sh install
```

Installing C++ agent on Windows from command line

Learn how to install the MiNiFi C++ agent from command line.

Procedure

1. To install the MiNiFi C++ agent from the command line with specific extensions, list all desired extensions in the following format:

```
msiexec /i nifi-minifi-cpp.msi AGREETOLICENSE=Yes ADDLOCAL=CM_C_bin,CM_C
_tzdata,InstallService,InstallVSRedistributableFiles,InstallConf,CM_C_ht
tp_curl,CM_C_sql /quiet
```

2. If you want to exclude specific extensions while installing all others, use this format:

```
msiexec /i nifi-minifi-cpp.msi AGREETOLICENSE=Yes ADDLOCAL=ALL REMOVE=CM
_C_sql /quiet
```

Follow the same pattern for all extension identifiers (for example: CM_C_sql. Take the extension name (for example, minifi-sql), replace the minifi- prefix with CM_C_ and replace dashes with underscores. For example, minifi-http-curl becomes CM_C_http_curl.

Note: To enable the agent to connect to the EFM server, ensure that the minifi-http-curl extension is installed.

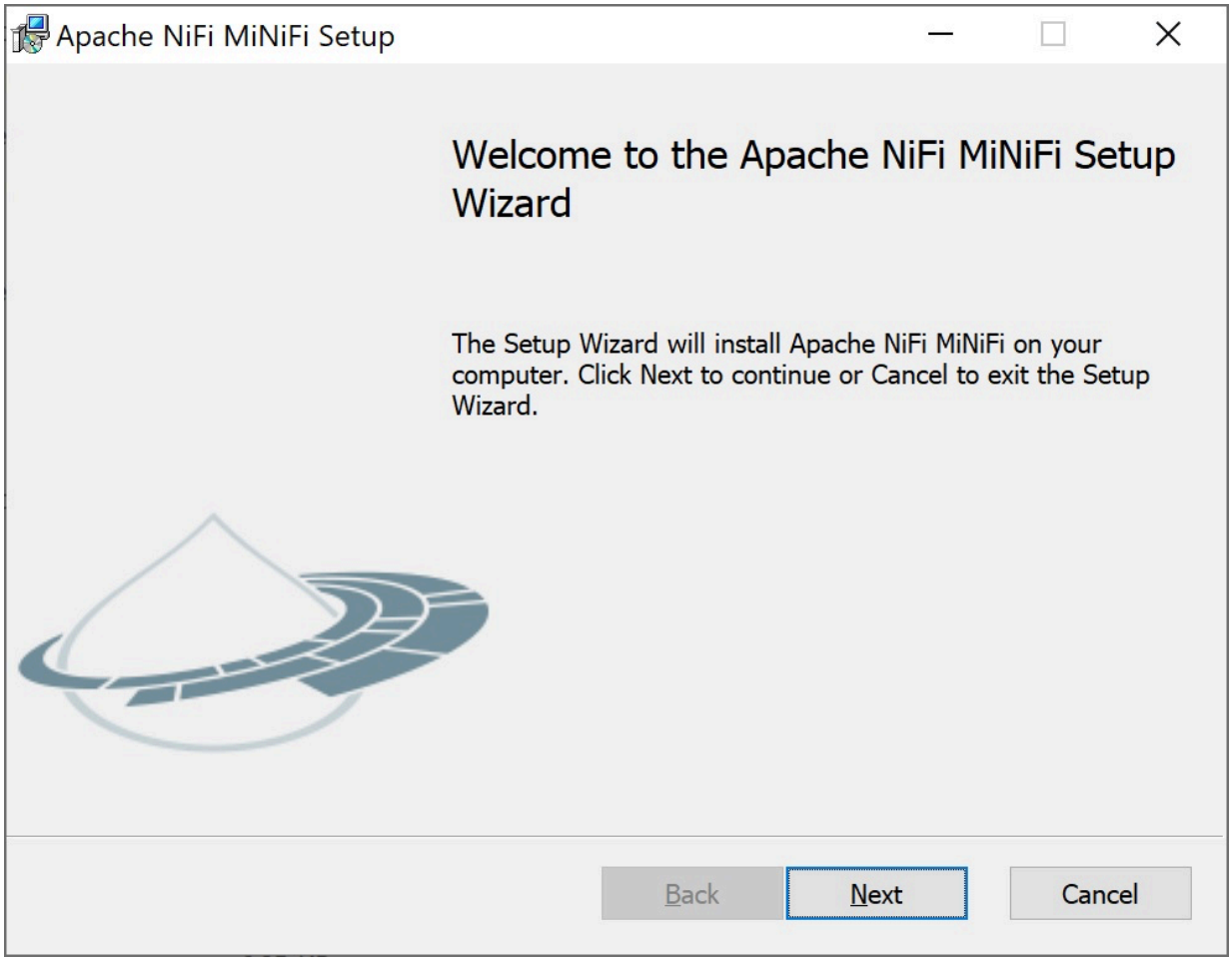
For more information and configuration options for the msiexec command, see the Microsoft documentation on msiexec.

Installing C++ agent on Windows using MSI

Learn how to install and configure the MiNiFi C++ agent on your Windows system by using the provided MSI.

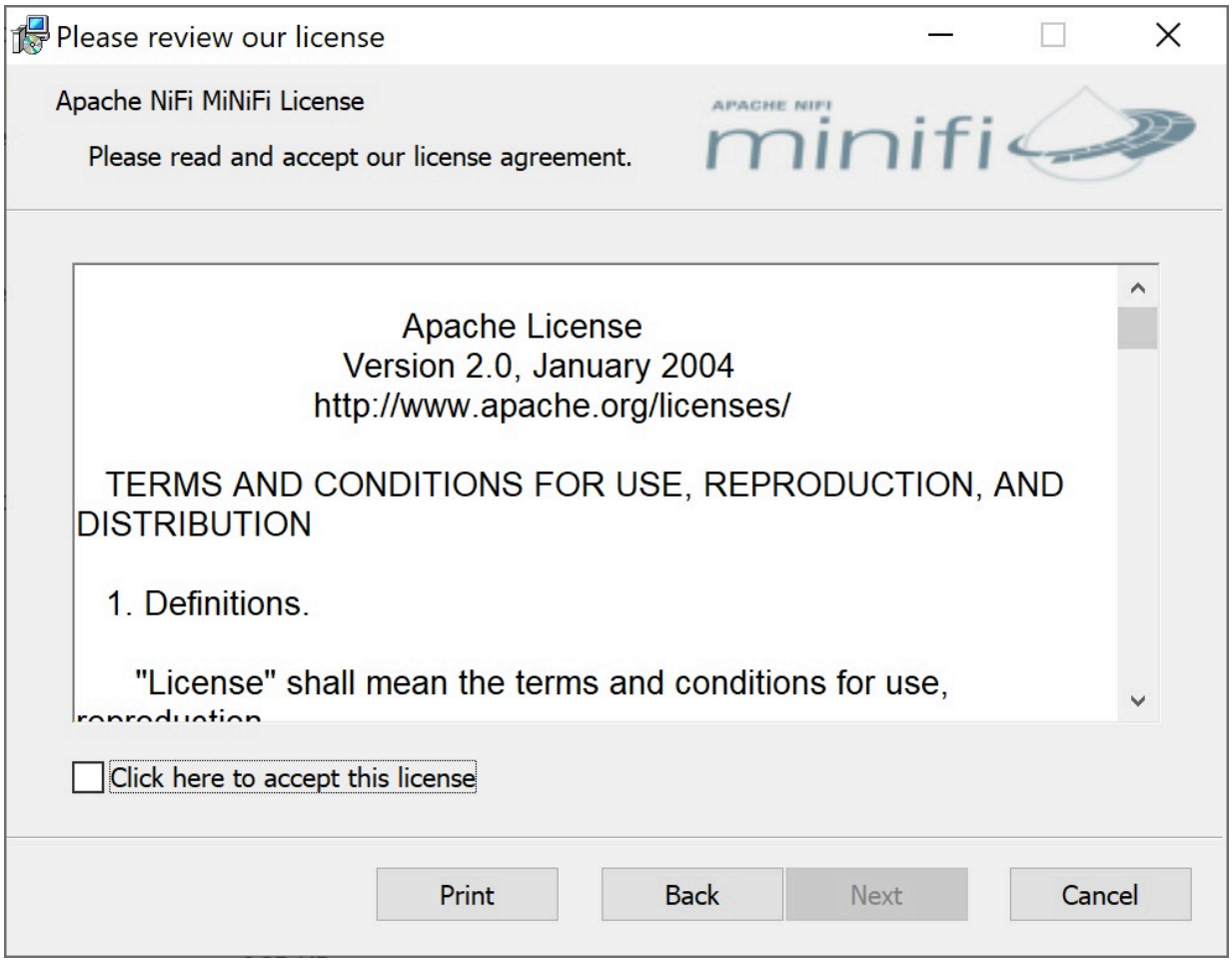
Procedure

1. Launch the Apache NiFi MiNiFi Setup wizard, and click Next.



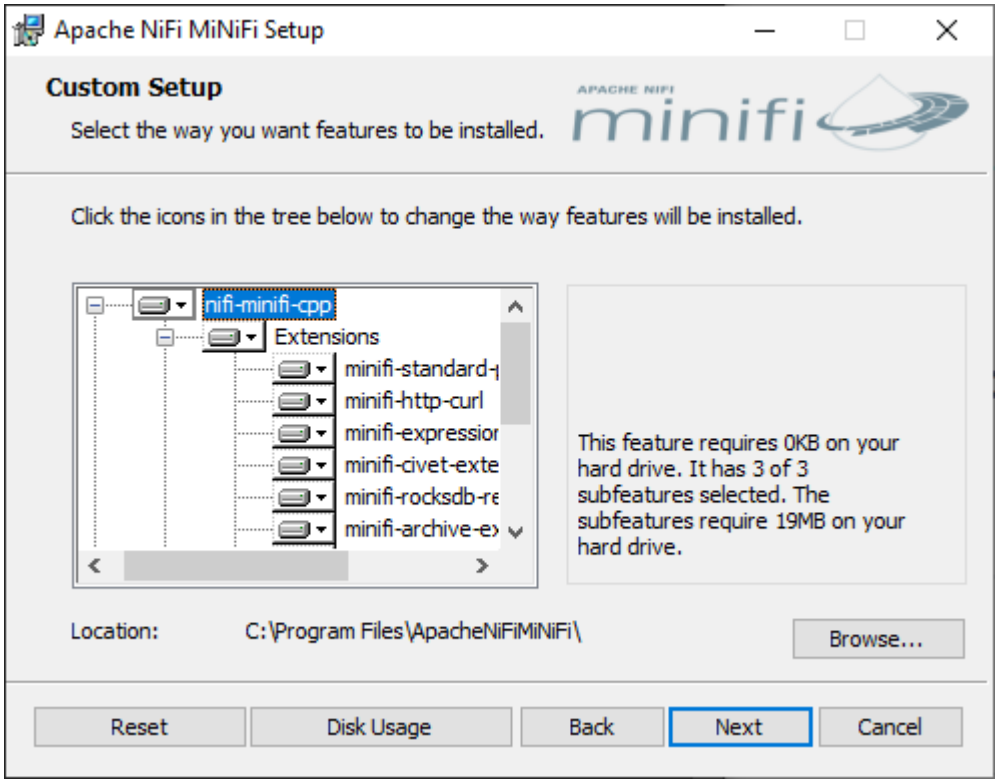
The Apache NiFi MiNiFi License page is displayed.

2. Check the Click here to accept this license option and click Next.



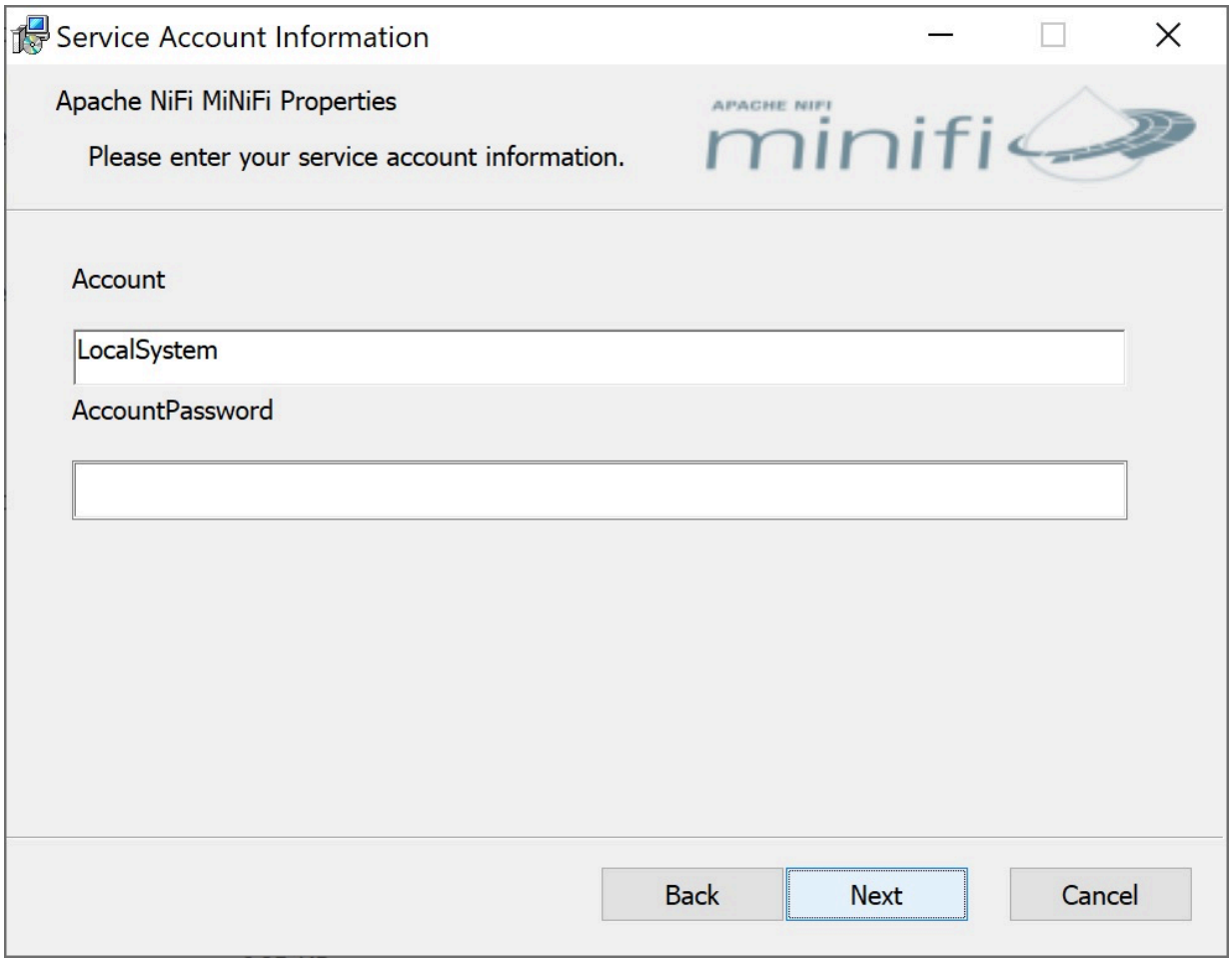
The Custom Setup page is displayed.

3. Select the extensions you want to install.
   By default, all extensions are enabled.

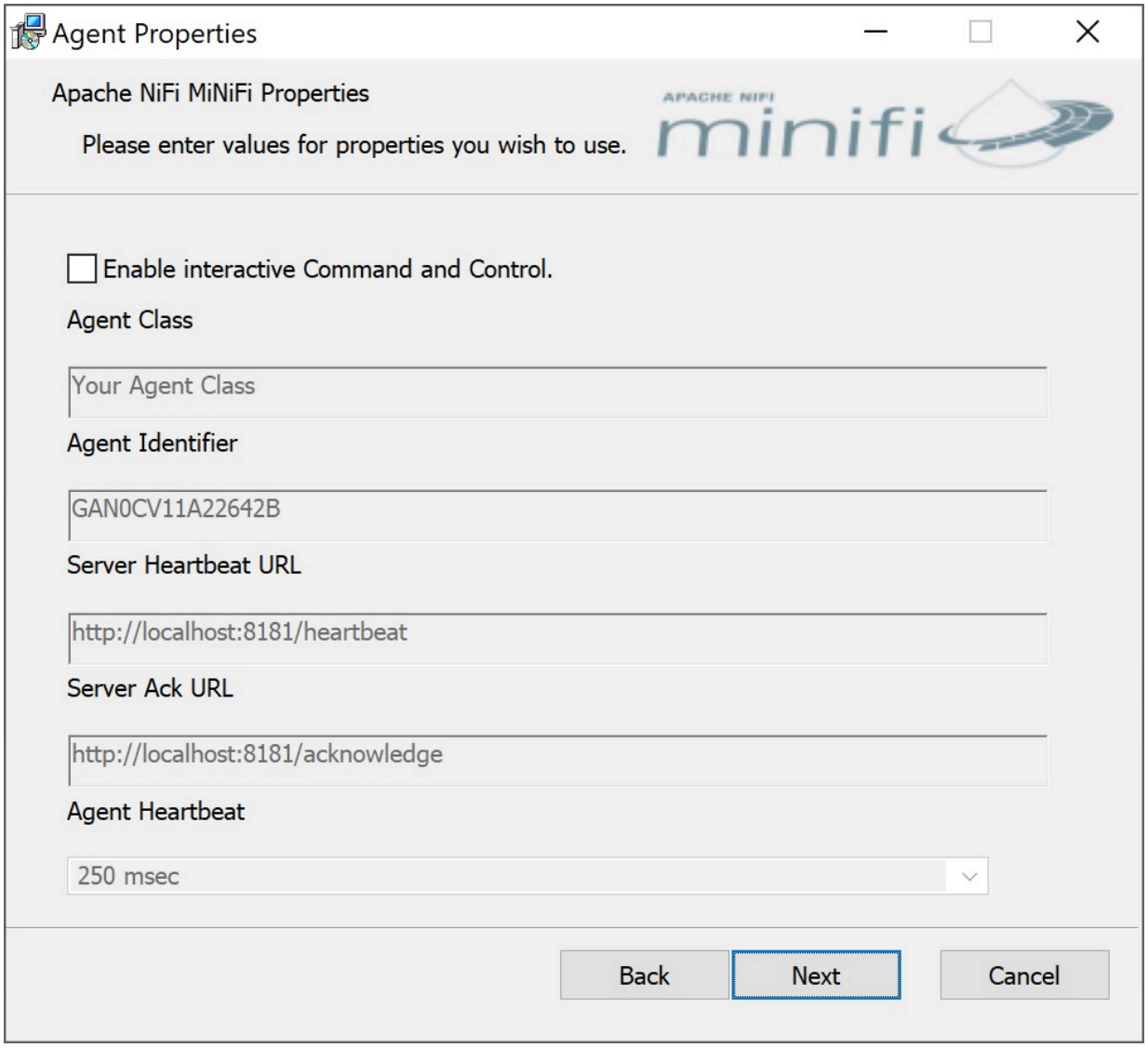


4. Click Next.
   The Service Account Information page is displayed.

5. Provide a user for the Windows service that will be installed and a corresponding password.

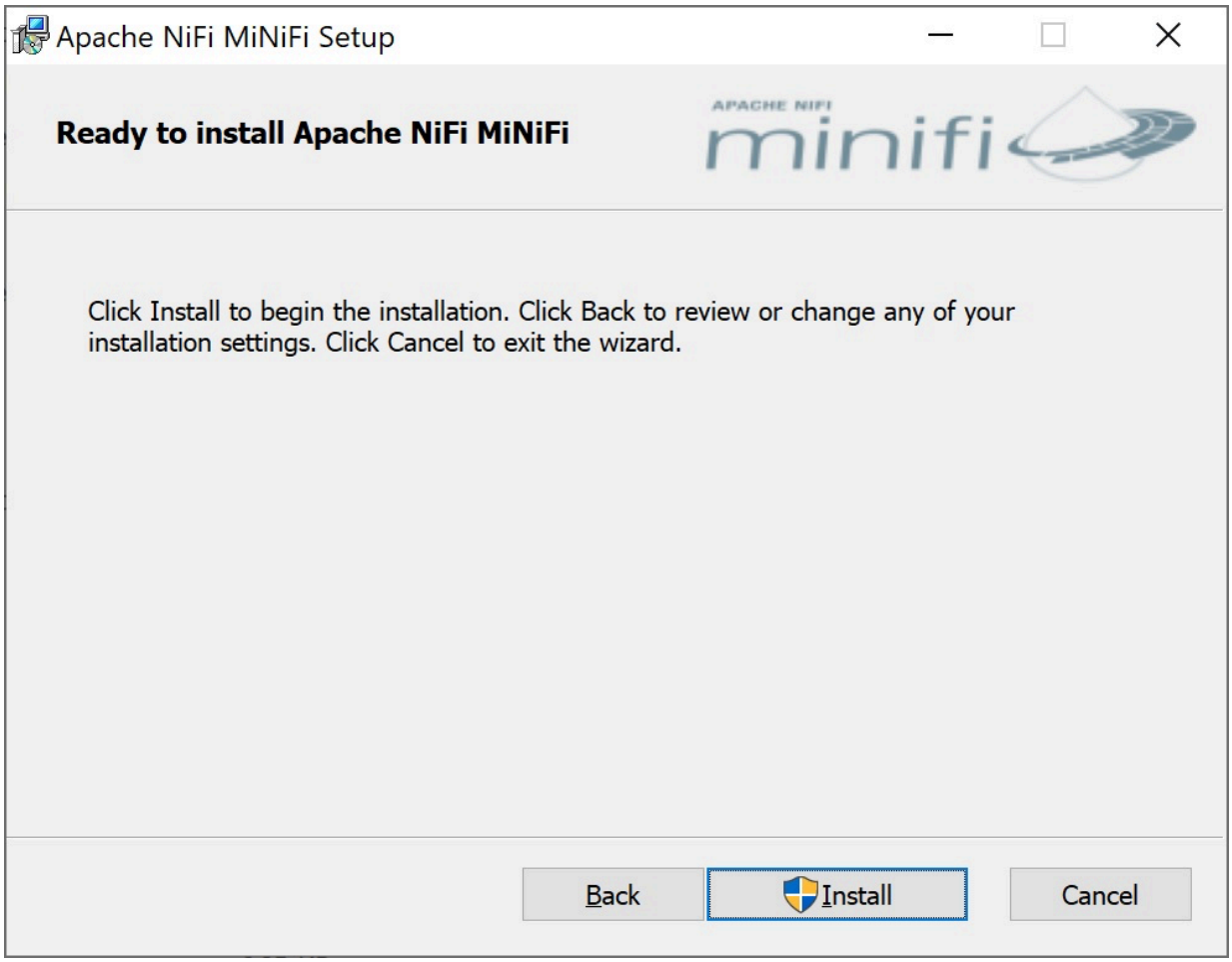


6. Click Next.
   The Agent Properties page is displayed.



7. Check the Enable interactive Command and Control option.
   Enabling this option allows you to customize your properties.
8. Edit the following properties:
   - Agent Class
   - Agent Identifier
   - Server Heartbeat URL
   - Server Ack URL
   - Agent Heartbeat
9. Click Next.

10. Click Install to initiate the installation process.



MiNiFi C++ agent configurations

Learn about different configurations that you can perform for your MiNiFi C++ agent.

Configuring your MiNiFi C++ agent

After you have installed the MiNiFi C++ agent, you need to update the configuration files for proper operation.

About this task

MiNiFi C++ relies on two configuration files: minifi.properties for general settings and config.yml for flow configuration.

Note: MiNiFi C++ does not automatically generate or overwrite settings in conf/minifi.properties, and it does not support the NiFi Properties Overrides syntax in config.yml.

Follow these steps to configure the MiNiFi C++ agent:

Procedure

1. From the MiNiFi home directory, locate and open the minifi.properties configuration file.

2. Configure the agent class to logically group MiNiFi C++ instances based on their functionality using the following setting:

```
nifi.c2.agent.class=[***AGENT_CLASS***]
```

Replace *[***AGENT_CLASS***]* with your agent class name. For example: nifi.c2.agent.class=my_agent_class

Note: Leading and trailing whitespaces are accepted for Agent Class names so consider this when configuring agents.

3. Configure the ID of the agent. If you do not specify an agent ID, MiNiFi generates a unique ID per agent instance.

```
nifi.c2.agent.identifier=[***AGENT_ID***]
```

Replace *[***AGENT_ID***]* with your chosen ID. For example: nifi.c2.agent.identifier=minifi_cpp_13

4. Enable MiNiFi to receive runtime flow instructions from EFM by setting the nifi.c2.enable property to true.

```
nifi.c2.enable=true
```

5. Configure your EFM server endpoint.

```
nifi.c2.rest.url=http://[***EFM SERVER_IP ADDRESS***]:10090/efm/api/c2-pr
otocol/heartbeat
nifi.c2.rest.url.ack=http://[***EFM SERVER_IP ADDRESS***]:10090/efm/api/c
2-protocol/acknowledge
```

Replace *[***EFM SERVER_IP ADDRESS***]* with the IP address of the EFM server endpoint. For example with efm-host.corp.local

6. Configure the heartbeat interval.

```
nifi.c2.agent.heartbeat.period=[***HEARTBEAT_INTERVAL***]
```

Replace *[***HEARTBEAT_INTERVAL***]* with the time period you want to set as the frequency for the the agent to send a heartbeat to EFM. The format can be an integer in milliseconds, or an integer followed by a time period unit.

For example: nifi.c2.agent.heartbeat.period=5000

5 sec is also a valid value.

7. Configure the metrics for your MiNiFi C++ agent.

For more information, see *Metrics in the Apache NiFi - MiNiFi - C++ C2 Readme*.

```
nifi.c2.root.class.definitions=metrics
nifi.c2.root.class.definitions.metrics.name=metrics
nifi.c2.root.class.definitions.metrics.metrics=runtimemetrics,loadmetrics
,processorMetrics
nifi.c2.root.class.definitions.metrics.metrics.runtimemetrics.name=Runtime
Metrics
nifi.c2.root.class.definitions.metrics.metrics.runtimemetrics.classes=De
viceInfoNode,FlowInformation
nifi.c2.root.class.definitions.metrics.metrics.loadmetrics.name=LoadMet
rics
nifi.c2.root.class.definitions.metrics.metrics.loadmetrics.classes=Queu
eMetrics,RepositoryMetrics
nifi.c2.root.class.definitions.metrics.metrics.processorMetrics.name=Proc
essorMetric
nifi.c2.root.class.definitions.metrics.metrics.processorMetrics.classes=
GetFileMetrics
```

8. Restart the agent to apply the changes.

   For the list of supported MiNiFi C++ processors, see *MiNiFi C++ agent processor support*.

Related Information

MiNiFi C++ agent processor support
Metrics in the Apache NiFi - MiNiFi - C++ C2 Readme

Configuring communication with secured EFM

When communicating with a secured Edge Flow Manager (EFM), you must configure additional properties.

About this task

If you are configuring a MiNiFi C++ agent, use the minifi.properties file located in the conf directory.

For the C++ agent, configure the following additional properties in the minifi.properties file for communication with a secured EFM:

```
# Security Properties #
# enable tls #
nifi.remote.input.secure=true
# if you want to enable client certificate base authorization #
nifi.security.need.ClientAuth=true
# setup the client certificate and private key PEM files #
nifi.security.client.certificate=
nifi.security.client.private.key=
# setup the client private key passphrase file #
nifi.security.client.pass.phrase=./conf/password
# setup the client CA certificate file #
nifi.security.client.ca.certificate=
```

Configuring MiNiFi C++ repositories

Learn how to configure and encrypt MiNiFi C++ repositories for efficient data storage.

Repository types

Similarly to NiFi, the MiNiFi C++ agent uses repositories to store various types of data. There are three primary repository types:

Content repository

It stores the actual contents of each flowfile.

Flow File repository

It contains flowfile metadata, including the current state and attributes of every flowfile, and a content repository identifier.

Provenance repository

It tracks changes and stores all events related to flowfiles. Note that this repository is disabled by default and is not well-maintained.

Configuring repository storage locations

Persistent repositories use configurable paths to store data. The repository locations and their defaults are defined below. By default the MINIFI_HOME environment variable is used. If you do not specify this variable, the root installation folder is used. You can specify your own path replacing the default variables in the minifi.properties file.

```
nifi.database.content.repository.directory.default=${MINIFI_HOME}/content_re
pository
nifi.flowfile.repository.directory.default=${MINIFI_HOME}/flowfile_reposito
ry
```

```
nifi.provenance.repository.directory.default=${MINIFI_HOME}/provenance_re
pository
```

You can also use a single database to store multiple repositories with the minifidb:// scheme. This can help with migration and centralizing agent state persistence. In the scheme, the final path segment designates the column family in the repository, while the preceding path indicates the directory where the rocksdb database is created.

For example: In minifidb:///home/user/minifi/agent_state/flowfile, a directory is created at /home/user/minifi/agent_state populated with rocksdb-specific content. In that repository, a logically separate subdatabase is created under the name flowfile.

```
nifi.flowfile.repository.directory.default=minifidb://${MINIFI_HOME}/agent_s
tate/flowfile nifi.database.content.repository.directory.default=minifidb://
${MINIFI_HOME}/agent_state/content
nifi.state.manangement.provider.local.path=minifidb://${MINIFI_HOME}/agen
t_state/processor_states
```

Note: You should not use the same directory with and without the minifidb:// scheme simultaneously. Moreover, the default name is restricted and should not be used.

Configuring repository backends

You can configure all three repositories (Content, Flow File, and Provenance) to be either volatile or persistent. Configuring volatile repositories means that their state is kept in memory and flushed upon restart. This can increase the performance, but can also cause data loss in case of restart while data is being processed by the agent.

Content repository

By default, the content repository uses a RocksDB backend, which is a persistent, in-process key-value store. There is another, filesystem-based alternative backend for this repository. In most use cases, the RocksDB backend offers better performance and transactional guarantees over the filesystem backend, but it has a limitation: it cannot store flow file contents larger than 4 GB.

You can configure the repository backends in the minifi.properties file as follows:

nifi.content.repository.class.name

Possible values:

- DatabaseContentRepository: RocksDB backend - This is the default value.
- FilesystemRepository: file system backend - Use it if you work with flow files larger than 4 GB.
- VolatileContentRepository: in-memory, non-persistent storage backend - You should only use it for testing.

Flow File repository

By default, the flow file repository uses a RocksDB backend, which is a persistent, in-process key-value store.

You can configure the repository backends in the minifi.properties file as follows:

nifi.flowfile.repository.class.name

Possible values:

- FlowFileRepository: RocksDB backend - this is the default value
- VolatileFlowFileRepository: in-memory, non-persistent storage backend - you should only use it for testing.

Provenance repository

Cloudera does not recommend changing the provenance repository backend. If you need NiFi-style provenance data, contact your Cloudera representative with the feature request.

You can configure the repository backends in the minifi.properties file as nifi.provenance.repository.class.name.

Possible values:

- NoOpRepository: disable provenance - This is the default value.
- ProvenanceRepository: RocksDB backend - You should not use this backend repository as it is not thoroughly tested. Cloudera advises caution if you decide to use it.
- VolatileProvenanceFileRepository: in-memory, non-persistent storage backend - You should only use it for testing.

Other configuration options for volatile repositories:

```
# configuration options
# maximum number of entries to keep in memory
nifi.volatile.repository.options.flowfile.max.count=10000

# maximum number of bytes to keep in memory
nifi.volatile.repository.options.flowfile.max.bytes=1M

# maximum number of entries to keep in memory
nifi.volatile.repository.options.provenance.max.count=10000
# maximum number of bytes to keep in memory
nifi.volatile.repository.options.provenance.max.bytes=1M


# maximum number of entries to keep in memory
nifi.volatile.repository.options.content.max.count=100000
# maximum number of bytes to keep in memory
nifi.volatile.repository.options.content.max.bytes=1M

# limits locking for the content repository
nifi.volatile.repository.options.content.minimal.locking=true
```

Systems that have limited memory must be cognisant of the above options. Limiting the maximum count for the number of entries limits memory consumption but it also limits the number of events that can be stored. If you limit the amount of volatile content you are configuring, you may have excessive session rollback due to invalid stream errors that occur when a claim cannot be found.

Repository encryption

You can encrypt the repository starting with the CEM Agents 1.21.06 release.

You can provide the RocksDB-backed repositories with a key to request their encryption (using AES-256-CTR) in the conf/bootstrap.conf file:

```
nifi.flowfile.repository.encryption.key=805D7B95EF44DC27C87FFBC4DFDE376DAE60
4D55DB2C5496DEEF5236362DE62E
nifi.database.content.repository.encryption.key=
# nifi.state.management.provider.local.encryption.key=
```

In the above configuration, the first line makes the Flow File repository use the specified 256-bit key. The second line triggers the generation of a random 256-bit key persisted back into conf/bootstrap.conf, which the Content repository then uses for encryption. This way, you can request encryption without being concerned about what key to use. Finally, as the last line is commented out, it makes the state manager use plaintext storage and not trigger encryption.

When multiple repositories use the same directory (as with minifidb:// scheme), the repositories should either be all plaintext or all encrypted with the same key.

Encrypting sensitive data

You can prevent accidental exposure of passwords by encrypting sensitive configuration properties in the minifi.properties file. Learn how to encrypt sensitive data.

MiNiFi provides a encrypt-config tool (encrypt-config.exe on Windows) located in the bin directory of your installation, next to the main MiNiFi binary. This tool allows you to encrypt sensitive configuration properties in the minifi.properties file and also encrypts f the flow configuration (config.yml by default).

The security of the encryption relies on the security of the bootstrap.conf file, which stores the encryption key.

Note: This feature is available starting with the release of MiNiFi C++ 1.20.09.

The terminologies used in this section are as follows:

minifi home

The directory specified by the --minifi-home option to encrypt-config.

configuration directory

The <minifi home>/conf directory.

properties file

The <minifi home>/conf/minifi.properties file.

flow configuration

The file specified in the properties file with the key nifi.flow.configuration.file. If not specified, it defaults to <minifi home>/conf/config.yml.

bootstrap file

The <minifi home>/conf/bootstrap.conf file.

sensitive property

Any property in the properties file that you wish to encrypt.

Encryption of the configuration properties

If you have a minifi.properties file in your MiNiFi configuration directory /var/tmp/minifi-home/conf containing the following sensitive properties:

```
minifi-properties
...
nifi.security.client.pass.phrase=my_pass_phrase
...
nifi.rest.api.user.name=admin
nifi.rest.api.password=password123
...
```

You can run the encrypt-config tool as shown in the following example:

```
$ ./bin/encrypt-config --minifi-home /var/tmp/minifi-home

Generating a new encryption key...
Wrote the new encryption key to /var/tmp/minifi-home/conf/bootstrap.conf
Encrypted property: nifi.security.client.pass.phrase
Encrypted property: nifi.rest.api.password
Encrypted 2 sensitive properties in /var/tmp/minifi-home/conf/minifi.propert
ies
```

The tool performs the following actions:

1. It generates a new encryption key.
2. It creates a bootstrap.conf file in your configuration directory and writes the encryption key to this file.
3. It encrypts the sensitive properties using this encryption key.
4. It adds a something.protected encryption marker after each encrypted property.

After running the tool, the bootstrap.conf and minifi.properties files will look like as shown in the following examples:

```
bootstrap.conf
nifi.bootstrap.sensitive.key=77cd3f88ab997f7ae99b13c70877c5274c3b7b495f601f
290042b14e7db4d542
```

```
minifi.properties
...
nifi.security.client.pass.phrase=STBmfU0uk5hgSYG5O3uJM3HeZjrYJz//||vE/V65Qi
MgSatzScaPYkraVrpWnBExVgVX/CwyXx
nifi.security.client.pass.phrase.protected=xsalsa20poly1305
...
nifi.rest.api.user.name=admin
nifi.rest.api.password=q8XNjJMoVABXz7sks5O6nhaTqqRay4gF||U3762djgMVguHI6GjRl
+iCCDSkIdTFzKDCXi
nifi.rest.api.password.protected=xsalsa20poly1305
...
```

Note: Protect the bootstrap.conf file to make sure it is only readable by the user running MiNiFi.

Additional sensitive properties

By default, encrypt-config encrypts a short list of default sensitive properties. To encrypt more properties, add a nifi .sensitive.props.additional.keys setting with a comma-separated list of additional sensitive properties to your minifi.properties file before running the encrypt-config tool. For example:

```
minifi.properties
...
nifi.sensitive.props.additional.keys=nifi.rest.api.user.name,controller.soc
ket.host,controller.socket.port
...
```

The tool encrypts these additional properties. You can also do this after you have already encrypted some properties. In that case, the tool encrypts the additional properties using the existing encryption key and leaves the other, already encrypted, sensitive properties intact.

Modifying sensitive properties

If you later need to modify the value of a sensitive property that was encrypted earlier, perform the following steps:

1. Replace the encrypted value with the new unencrypted value.
2. Delete the something.protected=... line added by the tool.
3. Re-run the encrypt-config tool.

The tool encrypts the modified property using the existing encryption key in bootstrap.conf and leaves the other, already encrypted, sensitive properties intact.

Encryption of the flow definition

To encrypt the flow configuration file, pass the --encrypt-flow-config flag to encrypt-config. This encrypts the flow configuration file, not just the sensitive properties.

Updating the encryption key

To change the encryption key, perform the following steps:

1. If files are already encrypted, there should be a nifi.bootstrap.sensitive.key=... line in the bootstrap.conf file (that is, have access to the original key). If not, manually replace all encrypted data (sensitive properties and flow configuration) with their original, unencrypted values (or some other new value).
2. If present, rename the nifi.bootstrap.sensitive.key=... property in bootstrap.conf to nifi.bootstrap.sensitive.key.old =... (that is, add .old suffix to the property name).
3. If you have a specific encryption key, add it to the bootstrap.conf file (as nifi.bootstrap.sensitive.key=<your encry ption key here>). If you provide no encryption key (no nifi.bootstrap.sensitive.key property in bootstrap.conf, or no bootstrap.conf at all), a new key is randomly generated and written to bootstrap.conf.
4. Re-run the encrypt-config tool.

Note: Take care when changing the encryption key, especially if the flow configuration is encrypted. Re-encrypt it before deleting the old key. You will receive a warning if you do not request its re-encryption.

```
$ cat /var/tmp/minifi-home/conf/bootstrap.conf

nifi.bootstrap.sensitive.key.old=0728061a041edb09445ae4dbd95f11bd255bb0b467
b8efb239e665aea5ace46b
nifi.bootstrap.sensitive.key=46af2c11a3f24c8c875ab4bee65e18a75f825fc3a4e0
3abdc8ce49d405b0b730

$ ./bin/encrypt-config --minifi-home /var/tmp/minifi-home

Old encryption key found in conf/bootstrap.conf
Using the existing encryption key found in conf/bootstrap.conf
Successfully decrypted property "nifi.security.client.pass.phrase" using old
 key.
Encrypted property: nifi.security.client.pass.phrase
Encrypted 1 sensitive property in conf/minifi.properties
WARNING: you did not request the flow config to be updated, if it is curre
ntly encrypted and the old key is removed, you won't be able to recover the
flow config.
```

If you forgot to specify the --encrypt-flow-config flag, you can re-run encrypt-config with the flag, and it re-encrypts the flow configuration file, as well.

It is always safe to re-run encrypt-config. If it does not find anything new to encrypt, it does not do anything.

When you have successfully re-encrypted all sensitive properties and the flow configuration file(s), you can delete the nifi.bootstrap.sensitive.key.old line from the bootstrap file.

Automatic encryption

Specify the property nifi.flow.configuration.encrypt=true, in the properties file to have the new flow configuration written to the disk encrypted after a flow update (originating from a C2 server). It requires that you have a conf/boo tstrap.conf in your minifi home, containing an encryption key (nifi.bootstrap.sensitive.key). This master key is also used on agent startup to decrypt the flow configuration file.

Integrating with the Windows certificate store

Learn how to enable MiNiFi C++ to get certificates from truststore of the OS.

If you want MiNiFi to communicate with EFM (C2) securely using HTTPS, you need a server certificate that EFM uses to identify itself and a client certificate that MiNiFi uses to identify itself, as well as a private key corresponding to the client certificate.

Manual setup of the client and server certificates on the MiNiFi side:

```
nifi.remote.input.secure=true
nifi.security.need.ClientAuth=true
nifi.security.client.certificate=C:\opt\nifi\data\ssl\client-certificate.pem
nifi.security.client.private.key=C:\opt\nifi\data\ssl\client-certificate.key
#nifi.security.client.pass.phrase=
nifi.security.client.ca.certificate=C:\opt\nifi\data\ssl\server-certificat
e.pem
#nifi.security.use.system.cert.store=
```

If both client and server certificates are in the LocalMachine (= "Local Computer") system certificate store (in MY = "Personal" and ROOT = "Trusted Root Certification Authorities", respectively), then you can simply do:

```
nifi.remote.input.secure=true
nifi.security.need.ClientAuth=true
#nifi.security.client.certificate=
#nifi.security.client.private.key=
#nifi.security.client.pass.phrase=
#nifi.security.client.ca.certificate=
nifi.security.use.system.cert.store=true
```

Ensure that the client certificate is exportable.

If you need to select the client certificate by CN, you can add the following property:

```
nifi.security.windows.client.cert.cn=<myCertificateIssuedToName>
```

If you need to select the client certificate by Extended (= "Enhanced") Key Usage, you can add the following property:

```
nifi.security.windows.client.cert.key.usage=Client Authentication, Server Au
thentication
```

You can also use a different system store location or a different system store for the client and server certificates, if needed:

```
# instead of LocalMachine
nifi.security.windows.cert.store.location=CurrentUser
# instead of MY
nifi.security.windows.client.cert.store=TrustedPeople

# instead of ROOT
nifi.security.windows.server.cert.store=TrustedPublisher
```

Starting MiNiFi C++ Agent

After MiNiFi is installed and configured, you can start it using the following steps.

Before you begin

Ensure that you have completed the installation steps. For istructions, see Installing your MiNiFi C++ Agent.

Procedure

1. Open a terminal window and navigate to the MiNiFi installation directory.

2. Start the MiNiFi agent:
   - To start MiNiFi in the foreground, use the following command:

     ```
     bin/minifi.sh run
     ```
   - To start MiNiFi in the background, use the following command:

     ```
     bin/minifi.sh start
     ```

Troubleshooting MiNiFi C++ agent

You might need to perform extra configuration steps for the MiNiFi C++ agent.

If you are using CENTOS7 operating system, you might get the following error when you run the agent:

```
PID 15860 is stale, removing pid file at /grid/0/cloudera_flow_management/cp
p-minifi/bin/.minifi.pid

Starting MiNiFi with PID 10075 and pid file /grid/0/cloudera_flow_manageme
nt/cpp-minifi/bin/.minifi.pid

-bash-4.2$ [2019-07-infol 14:36:04.369] [main] [info] Using MINIFI_HOME=/
grid/0/cloudera_flow_management/cpp-minifi from environment.

[2019-07-infol 14:36:04.369] [org::apache::nifi::minifi::Properties] [info]
 Using configuration file located at /grid/0/cloudera_flow_management/cpp-m
inifi/conf/minifi-log.properties, from ./conf/minifi-log.properties
setting default dir to /grid/0/cloudera_flow_management/cpp-minifi/content_
repository

Could not find platform independent libraries <prefix>

Could not find platform dependent libraries <exec_prefix>

Consider setting $PYTHONHOME to <prefix>[:<exec_prefix>]
Fatal Python error: Py_Initialize: Unable to get the locale encoding

ImportError: No module named 'encodings'
```

The error occurs when you use an incorrect path for Python. To troubleshoot, update the following property values:

- PYTHONHOME=/usr
- PYTHONPATH=/usr/lib64/python3.4