Cloudera Edge Management 1.3.0

Cloudera Edge Management Installation

Date published: 2019-04-15
Date modified: 2021-10-18

https://docs.cloudera.com/

Legal Notice

Contents

Before you begin installing CEM

To start using Cloudera Edge Management (CEM), you need to install it. Learn how to obtain the CEM software bits, install Edge Flow Manager (EFM), install MiNiFi agents, and configure security. Also learn the system requirements to do so.

System requirements for EFM and NiFi Registry

Before you begin your installation of Cloudera Edge Management (CEM) software, carefully review the system requirements for Edge Flow Manager (EFM) and NiFi Registry to understand operating system, database, browser, and JDK support.

Operating system support

Operating System	Version
RHEL/CentOS	7.x, 8.2, 8.4
Debian	9
Ubuntu	16.04, 18.04

JDK support

JDK	Version
OpenJDK	JDK8, JDK11
Oracle JDK	JDK8, JDK11

Supported databases

Database	Version
PostgreSQL	9.x (x >= 6), 10.x, 11.x, 12.x
MySQL	5.6, 5.7, 8.0
MariaDB	10.2, 10.3, 10.4, 10.5, 10.6

Browser support

Browser	Version
Chrome	>=76.0 (latest version 80.0 at time of release)
Firefox	>=68.0 (latest version 72.0 at time of release)

Other support

Package	Version
Stunnel	5.x

Install Java for CEM

You should install Java on the machine where you will install the EFM Server and NiFi Registry, and on each machine where you will install a MiNiFi Java agent.

Procedure

1. Download JDK from the appropriate website.
2. Run the installation command appropriate for your operating system:

   For RHEL/CentOS:

   ```
   yum install java-1.8.0-openjdk
   ```

   For Debian and Ubuntu:

   ```
   apt-get install openjdk-8-jre
   ```

Related Information

OpenJDK Download

Oracle JDK Download

Installing the EFM server

Learn how to install the Edge Flow Manager (EFM) server. You need to install database, install NiFi Registry, and then install and configure EFM server.

Installing databases for CEM

Learn how to install a database and configure it with your Cloudera Edge Management (CEM) software to manage, control, and monitor dataflows.

The EFM Server requires a relational database. An H2 database is bundled with EFM that is used by default. While this is fine for development and test environments, for production environments, an external database is recommended.

For an external database, you can use either MySQL, MariaDB, or PostgreSQL. These topics describe how to install and configure MySQL, PostgreSQL, and MariaDB.

Note:

You should install either PostgreSQL or MySQL or MariaDB; all are not necessary.

Related Information

System requirements for CEM

Installing and configuring MySQL

Learn how to install and configure MySQL for Cloudera Edge Management (CEM).

For supported database versions, see *System Requirements*.

Install MySQL

If you want to use PostgreSQL or MariaDB instead of MySQL, you may skip these steps. See the instructions for PostgreSQL and MariaDB in the respective sections.

1. Log in to the machine on which you want to install MySQL to use for the EFM Server.
2. Install MySQL and the MySQL community server, and start the MySQL service. For installation instructions, check https://dev.mysql.com/doc/refman/5.7/en/installing.html.
3. Obtain the randomly generated MySQL root password.

   ```
   grep 'A temporary password is generated for root@localhost' \
   ```

```
/var/log/mysqld.log |tail -1
```

4. Reset the MySQL root password. Enter the following command. You are prompted for the password you obtained in the previous step. MySQL then asks you to change the password.

```
/usr/bin/mysql_secure_installation
```

5. Download the MySQL JDBC Connector and place it in the EFM lib directory: /path/to/efm-1.2.0/lib/.

   Download the MySQL database driver from https://dev.mysql.com/downloads/connector/j/.

   It is recommended to select the Platform Independent offering, download one of the archives, extract, and then copy the connector JAR to the lib directory.

Configure MySQL for use by EFM

1. Launch the MySQL shell:

```
mysql -u root -p
```

2. Create the database for the EFM service to use:

```
CREATE DATABASE efm CHARACTER SET latin1;
```

3. Create the efm user account, replacing the final IDENTIFIED BY string with your password:

```
CREATE USER 'efm'@'%' IDENTIFIED BY 'efmPassword';
```

4. Assign privileges to the efm account:

```
GRANT ALL PRIVILEGES ON efm.* TO 'efm'@'%';
```

5. Commit the operation:

```
FLUSH PRIVILEGES;
```

Configure the EFM database properties

1. Configure the database properties in the efm.properties file:

```
efm.db.url=jdbc:mysql://localhost:3306/efm
efm.db.driverClass=com.mysql.cj.jdbc.Driver
efm.db.username=efm
efm.db.password=efmPassword
```

   The URL should match the host and port of the machine running MySQL. The password should match the value that you set using the following command:

```
CREATE USER 'efm'@'%' IDENTIFIED BY 'efmPassword';
```

Installing and configuring PostgreSQL

Learn how to install and configure PostgreSQL for Cloudera Edge Management (CEM).

For supported database versions, see *System Requirements*.

Install PostgreSQL

If you are using MySQL or MariaDB instead of PostgreSQL, you may skip these steps. See the instructions for MySQL and MariaDB in the respective sections.

1. Install Red Hat Package Manager (RPM) according to the requirements of your operating system:

```
sudo yum install -y https://download.postgresql.org/pub/repos/yum/reporp
ms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

2. Install PostgreSQL version 9.6:

```
yum install postgresql96-server postgresql96-contrib postgresql96
```

3. Initialize the database:

   For example, if you are using CentOS 7, use the following syntax:

```
/usr/pgsql-9.6/bin/postgresql96-setup initdb
```

4. Start PostgreSQL.

   For example, if you are using CentOS 7, use the following syntax:

```
systemctl enable postgresql-9.6.service
systemctl start postgresql-9.6.service
```

5. Verify that you can log in:

```
sudo su postgres
psql
```

Configure PostgreSQL to allow remote connections

It is critical that you configure PostgreSQL to allow remote connections before you deploy CEM. If you do not perform these steps in advance of installing the EFM Server, the installation fails.

1. Open /var/lib/pgsql/9.6/data/pg_hba.conf and update to the following:

```
# "local" is for Unix domain socket connections only
local all all trust


# IPv4 local connections:
host all all 0.0.0.0/0 trust

# IPv6 local connections:
host all all ::/0 trust
```

2. Open /var/lib/pgsql/9.6/data/postgresql.conf and update to the following:

```
listen_addresses = '*'
```

3. Restart PostgreSQL.

```
systemctl stop postgresql-9.6.service
systemctl start postgresql-9.6.service
```

Configure PostgreSQL for use by EFM

1. Log in to PostgreSQL:

```
sudo su postgres
psql
```

2. Create a database for the EFM service to use:

```
create database efm;
CREATE USER efm WITH PASSWORD 'efmPassword';
GRANT ALL PRIVILEGES ON DATABASE "efm" to efm;
```

Configure the EFM database properties

1. Configure the database properties in the efm.properties file:

```
efm.db.url=jdbc:postgresql://localhost:5432/efm
efm.db.driverClass=org.postgresql.Driver
efm.db.username=efm
efm.db.password=efmPassword
```

The URL should match the host and port of the machine running PostgreSQL. The password should match the value that you set using the following command:

```
CREATE USER efm WITH PASSWORD 'efmPassword';
```

Installing and configuring MariaDB

Learn how to install and configure MariaDB for Cloudera Edge Management (CEM). MariaDB should act as a drop-in binary for MySQL so configuration should be very similar.

For supported database versions, see *System Requirements*.

Install MariaDB

If you are using MySQL or PostgreSQL instead of MariaDB, you may skip these steps. See the instructions for MySQL and PostgreSQL in the respective sections.

1. Log in to the machine on which you want to install MariaDB to use for the EFM Server.
2. Install MariaDB and the MariaDB server, and start the MariaDB service.

   For install instructions, check https://mariadb.com/kb/en/getting-installing-and-upgrading-mariadb/.
3. Use the following command to reset the MariaDB root password:

```
/usr/bin/mysql_secure_installation
```

   MariaDB then asks you to change the password.
4. Download the MariaDB JDBC connector and place it in the EFM lib directory:

```
/path/to/efm-1.3.0/lib/
```

5. Download the MariaDB database driver from https://mariadb.com/kb/en/about-mariadb-connector-j/.

Note: Cloudera recommends to select the platform independent offering, download one of the archives, extract, and then copy the connector JAR to the lib directory.

Configure MariaDB for use by EFM

1. Launch the MySQL shell:

```
mysql -u root -p
```

2. Create the database for the EFM service to use:

```
CREATE DATABASE efm CHARACTER SET latin1;
```

3. Create the efm user account, replacing the final IDENTIFIED BY string with your password:

```
CREATE USER 'efm'@'%' IDENTIFIED BY 'efmPassword';
```

4. Assign privileges to the efm account:

```
GRANT ALL PRIVILEGES ON efm.* TO 'efm'@'%';
```

5. Commit the operation:

```
FLUSH PRIVILEGES;
```

Configure the EFM database properties

1. Configure the database properties in the efm.properties file:

```
efm.db.url=jdbc:mariadb://localhost:3306/efm?useMysqlMetadata=true
efm.db.driverClass=org.mariadb.jdbc.Driver
efm.db.username=efm
efm.db.password=efmPassword
```

The URL should match the host and port of the machine running MariaDB. The password should match the value that you set using the following command:

```
CREATE USER 'efm'@'%' IDENTIFIED BY 'efmPassword';
```

Installing NiFi registry

Using Cloudera Edge Management (CEM) requires a NiFi Registry instance to store your dataflows. You can configure Edge Flow Manager (EFM) to use an existing NiFi Registry instance or install and run a new one.

Procedure

1. Locate the NiFi Registry tarball included with the EFM binaries download from https://www.cloudera.com/downloads.html.
2. Extract the NiFi Registry software to the desired directory on the target host.
3. Start NiFi Registry:

```
bin/nifi-registry.sh start
```

Installing EFM server

Learn how to install the Edge Flow Manager (EFM) server.

Procedure

1. Extract the Cloudera Edge Management (CEM) tars tarball:

```
tar -xzf CEM-version-platform-tars-tarball.tar.gz
```

2. Locate the EFM tarball efm-version-bin.tar.gz and move it to the desire host and installation directory.
3. Extract the EFM tarball in the desired installation directory:

```
tar -xzf efm-version-bin.tar.gz
```

Installing EFM as an operating system service

The Edge Flow Manager (EFM) executable supports installation as a service on most Linux distributions. This is an optional installation step that is not required if you prefer to start the EFM server from the efm.sh executable included in the EFM bin directory.

You can start the application as a service by using either init.d or systemd.

Install EFM as an init.d service

To install EFM as an init.d service, symlink bin/efm.sh to init.d.

```
$ sudo ln -s /path/to/efm/bin/efm.sh /etc/init.d/efm
```

Once installed, you can start and stop the service as you would other OS services. For example:

```
$ service efm start
```

To configure EFM to start automatically on system boot, use update-rc.d. See man update-rc.d for information on using this utility.

Note: The EFM application runs as the user who owns the efm.sh launch script. It is recommended to never run as root. The recommended best practice is to create a specific user for running efm. Then use chown to make that user the owner of efm.sh. For example:

```
$ chown efm:efm /path/to/efm/bin/efm.sh
```

It is also recommended to use Unix or Linux filesystem permissions in order to secure the EFM installation. The rule of setting minimal access permissions applies. All files in the EFM installation should only be accessible to the EFM run-as user. Configuration files should be made read-only (for example, chmod 40 0 <file>). Executable files, such as those in the bin directory, should be made read and executable only (for example, chmod 500 <file>). Directories in the EFM install location should be readable and writable to the EFM user (for example, chmod 600 <dir>).

Install EFM as a systemd service

Most modern Linux distributions now use systemd as the successor to init.d (System V). In many cases you can continue to use init.d, but it is also possible to launch EFM using systemd as a service configuration.

To install EFM as a systemd service, create a file named efm.service in the /etc/systemd/system directory. For example:

```
[Unit]
Description=efm
After=syslog.target

[Service]
User=efm
ExecStart=/path/to/efm/bin/efm.sh
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

Note: When using systemd, the run-as user, the PID file, and the console log file are managed by systemd and therefore must be configured by using appropriate fields in the service script. Consult the service unit configuration man page for more details.

To configure EFM to start automatically on system boot, use systemctl. See man systemctl for information on using this utility.

Configuring the EFM server

Once the Edge Flow Manager (EFM) server is installed, you can configure it by editing the efm.properties file. At minimum, you should edit the EFM server address, set up the connection to NiFi Registry, and configure the connection to your database.

Procedure

1. Open the efm.properties file located in $EFM_HOME/conf/efm.properties.

   Note: The EFM home directory is the root directory where you installed the EFM binary.

2. Configure the EFM Server address. Change efm.server.address=localhost to efm.server.address={EFM_IP_OR_HOSTNAME}, or use 0.0.0.0 to listen on all network interfaces.
3. Configure your connection by editing the following two properties:

   Change efm.nifi.registry.enabled=false to efm.nifi.registry.enabled=true.

   Change efm.nifi.registry.url to point to the base URL of your NiFi Registry server.
4. Configure one of the following properties to identify the NiFi Registry bucket you want EFM to use. Do not set both.

   efm.nifi.registry.bucketId=

   efm.nifi.registry.bucketName=

Set the encryption password for EFM

You need to set the efm.encryption.password property which specifies a master password used for encrypting sensitive data saved to the Edge Flow Manager (EFM) server.

You can set it through the efm.properties file, a command line argument, or an OS environment variable.

By default, the EFM application uses AES encryption. The encryption key used is deterministically derived from an encryption password that the admin user must provide to the application at runtime. The property that is read for the encryption password is efm.encryption.password. You can set the value for this property in following ways:

- As a command line argument: ./bin/efm.sh --efm.encryption.password=myEfmPassword
- As a Java System Property: -Defm.encryption.password=myEfmPassword
- As an OS environment variable: export EFM_ENCRYPTION_PASSWORD=myEfmPassword
- As a key/value pair in the efm.properties file: efm.encryption.password=myEfmPassword

Note: The master encryption password must be at least 12 characters long. It must be the same for all EFM instances.

The derived encryption key length is determined by your Java Runtime Environment encryption strength profiles.

- Unlimited Strength Encryption active: AES 256-bit key
- Unlimited Strength Encryption inactive: AES 128-bit key

It is strongly recommended that you enable Unlimited Strength Encryption in your Java Runtime Environment.

Open network ports for EFM

You should ensure that the required ports are available for the Edge Flow Manager (EFM) Server and its components.

Component	Port number
EFM Server HTTP	10090
EFM Server Constraint application protocol (CoAP)	8989

Starting the EFM server

After you install and configure the Edge Flow Manager (EFM) server, you can start it. In general, you should start the EFM server before you install and configure your MiNiFi agents. Learn how to start the EFM server.

Procedure

1. From the EFM server home directory, run:

```
bin/efm.sh start
```

2. Access the UI by browsing to the following location:

```
http://{EFM_HOST_OR_IP}:10090/efm/
```

Install the MiNiFi agents

Learn how to install the MiNiFi agents. You need to install and configure the MiNiFi agents and then start the MiNiFi agents. You can also do the same on Windows.

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.

Install your MiNiFi agent

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

Procedure

1. Download the tar.gz or zip files for the MiNiFi Java or C++ Agent.

```
wget {java.tar.gz}
```

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

2. To install the MiNiFi Agent, extract the file to your desired home directory.

Install the C++ agent on Windows from command line

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

Procedure

1. To install the MiNiFi C++ agent from the Command Line with the desired extensions, list all the extensions you would like to install:

```
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. Optional. In order to exclude some extensions but otherwise install all other extensions:

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

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

Note: For the agent to be able to connect to the EFM server, you must have the minifi-http-curl extension installed.

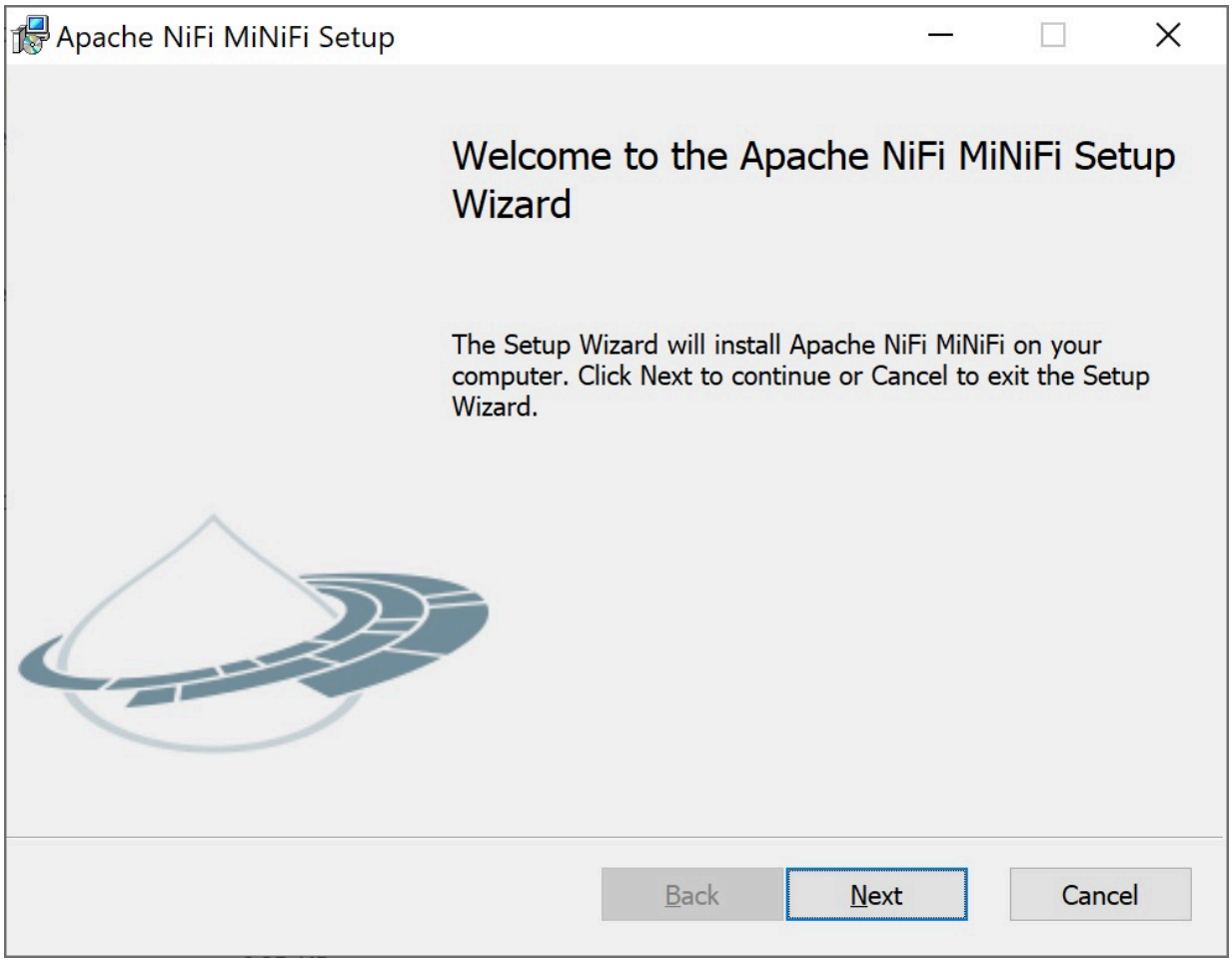
For more information and configuration options for msiexec, see https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/msiexec.

Install the C++ agent on Windows using the MSI

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

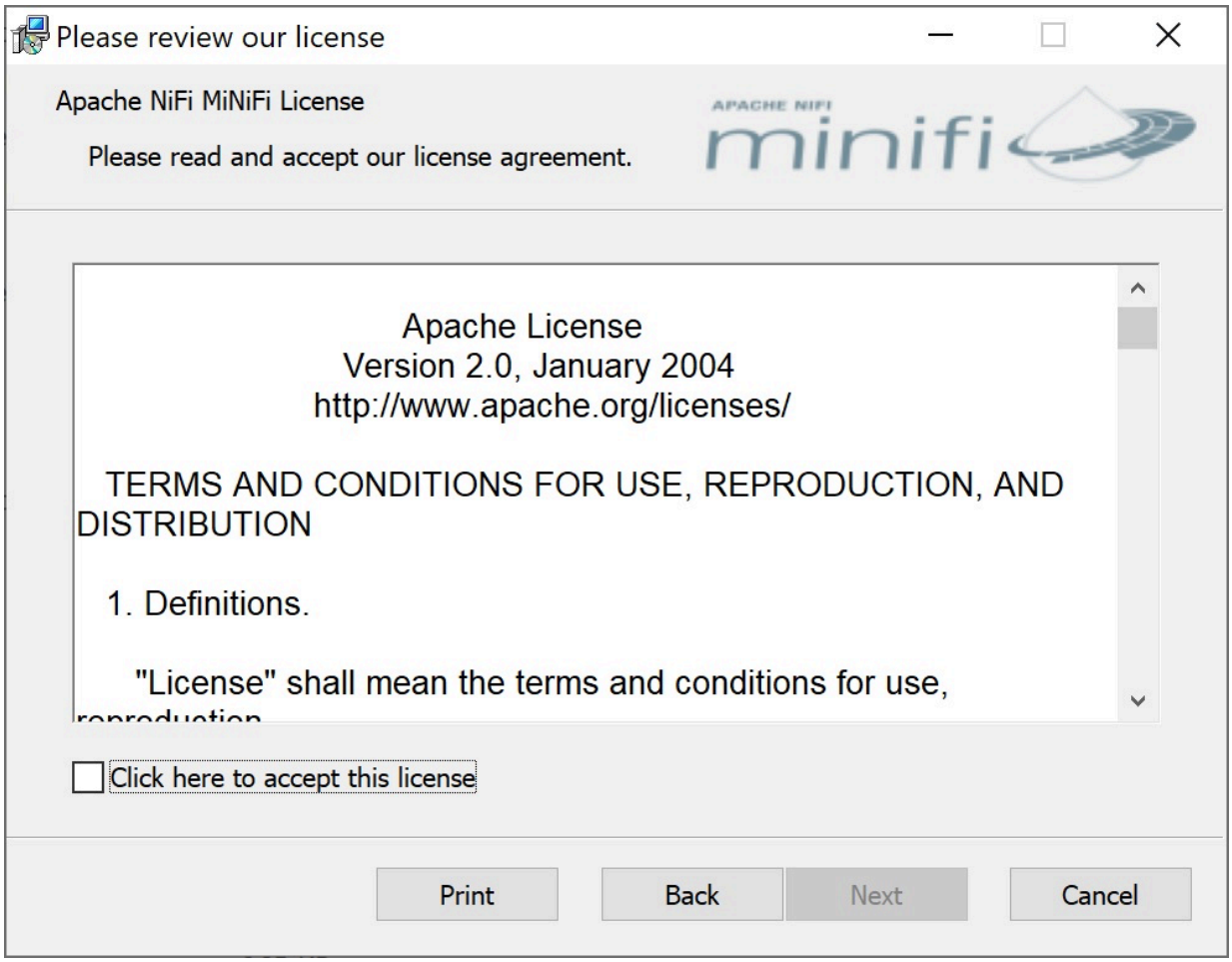
Procedure

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



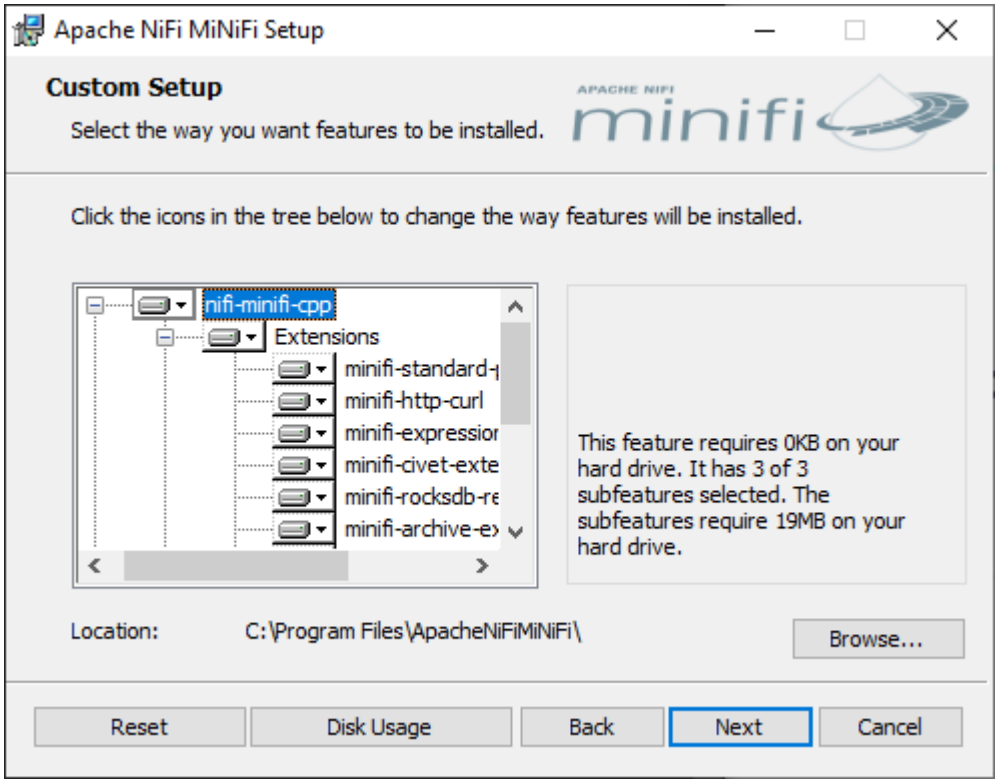
The Apache NiFi MiNiFi License page appears.

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



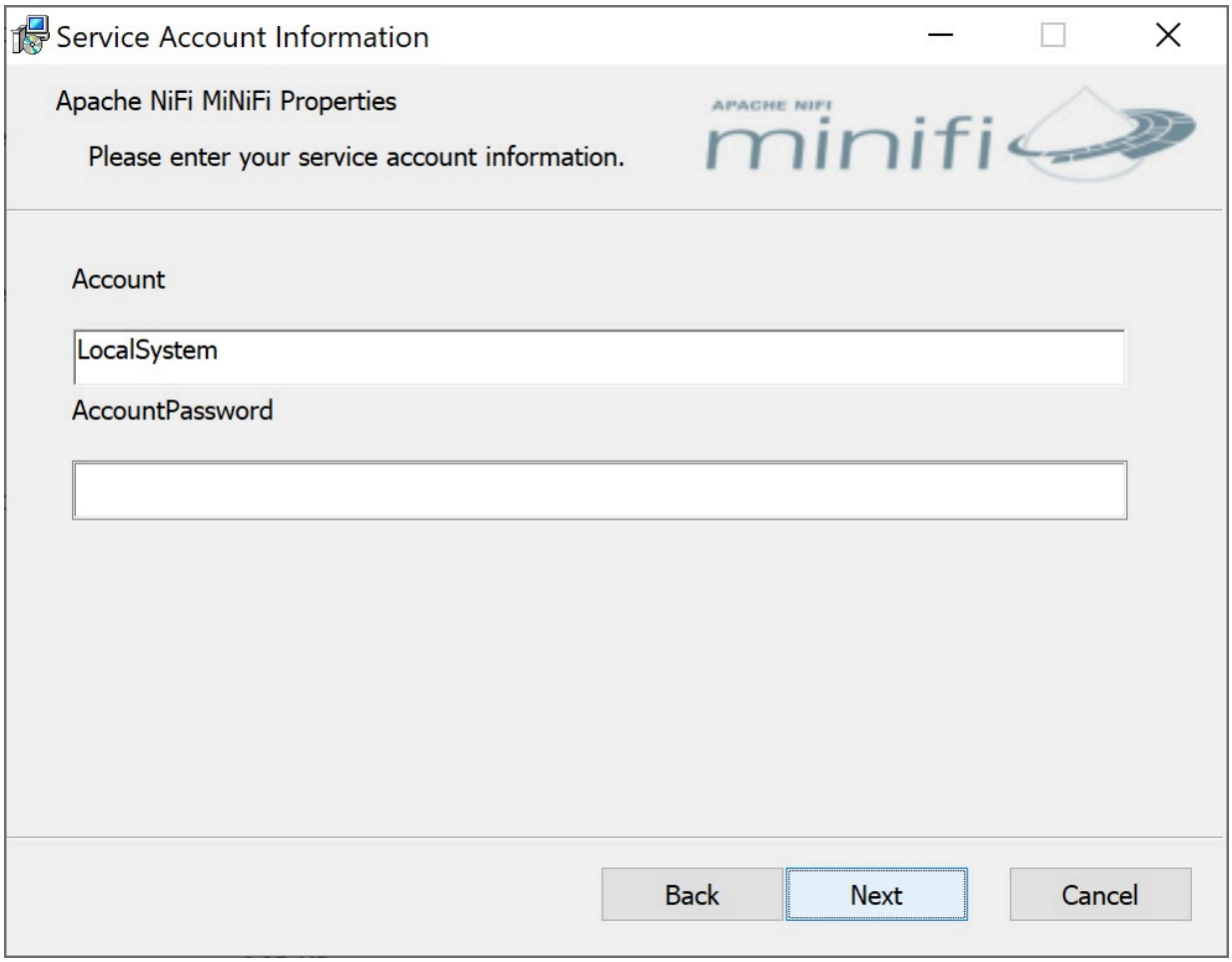
The Custom Setup page appears.

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

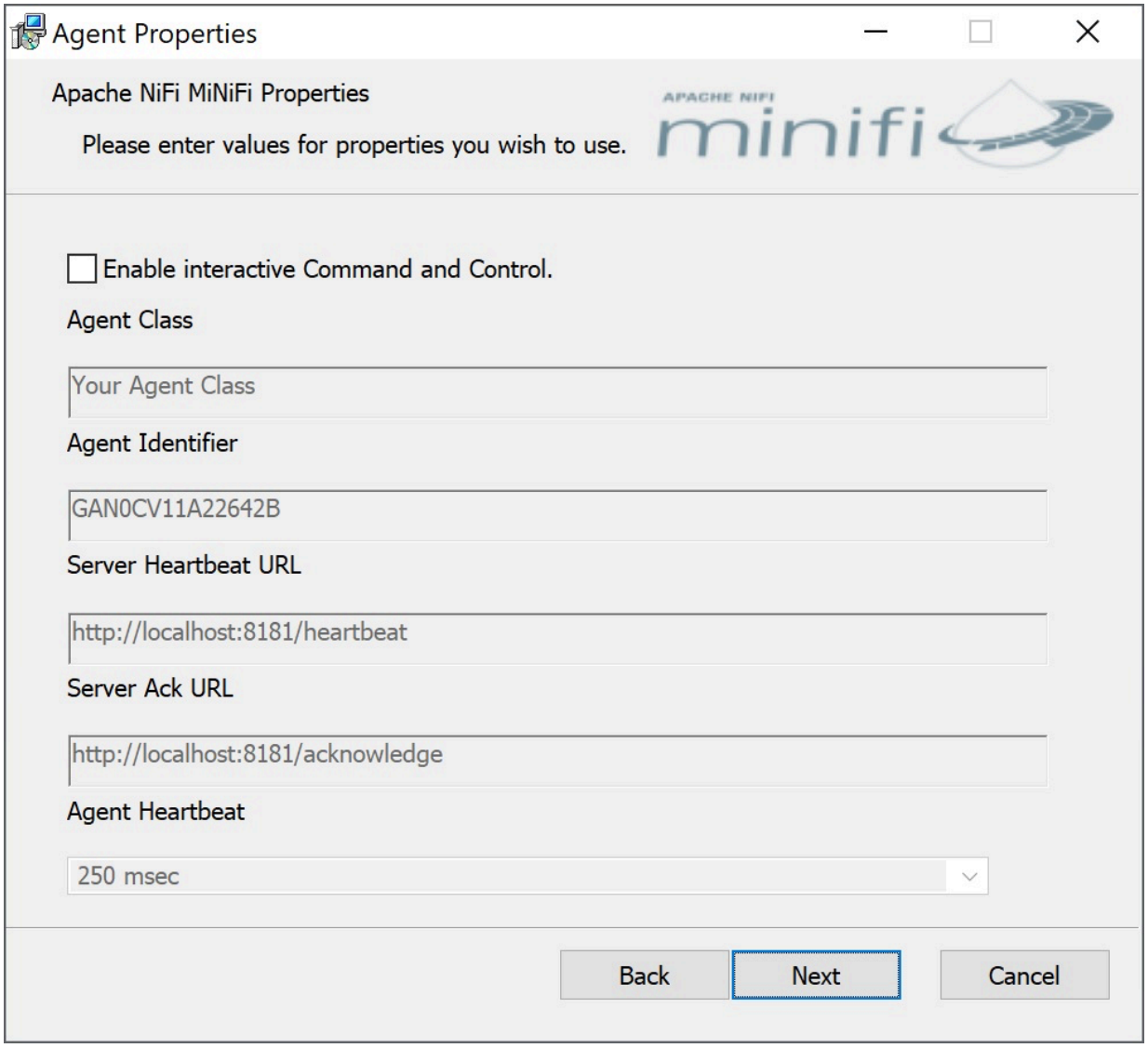


4. Click Next.
   The Service Account Information page appears.

5. Enter a user for the windows service that is installed and a password.

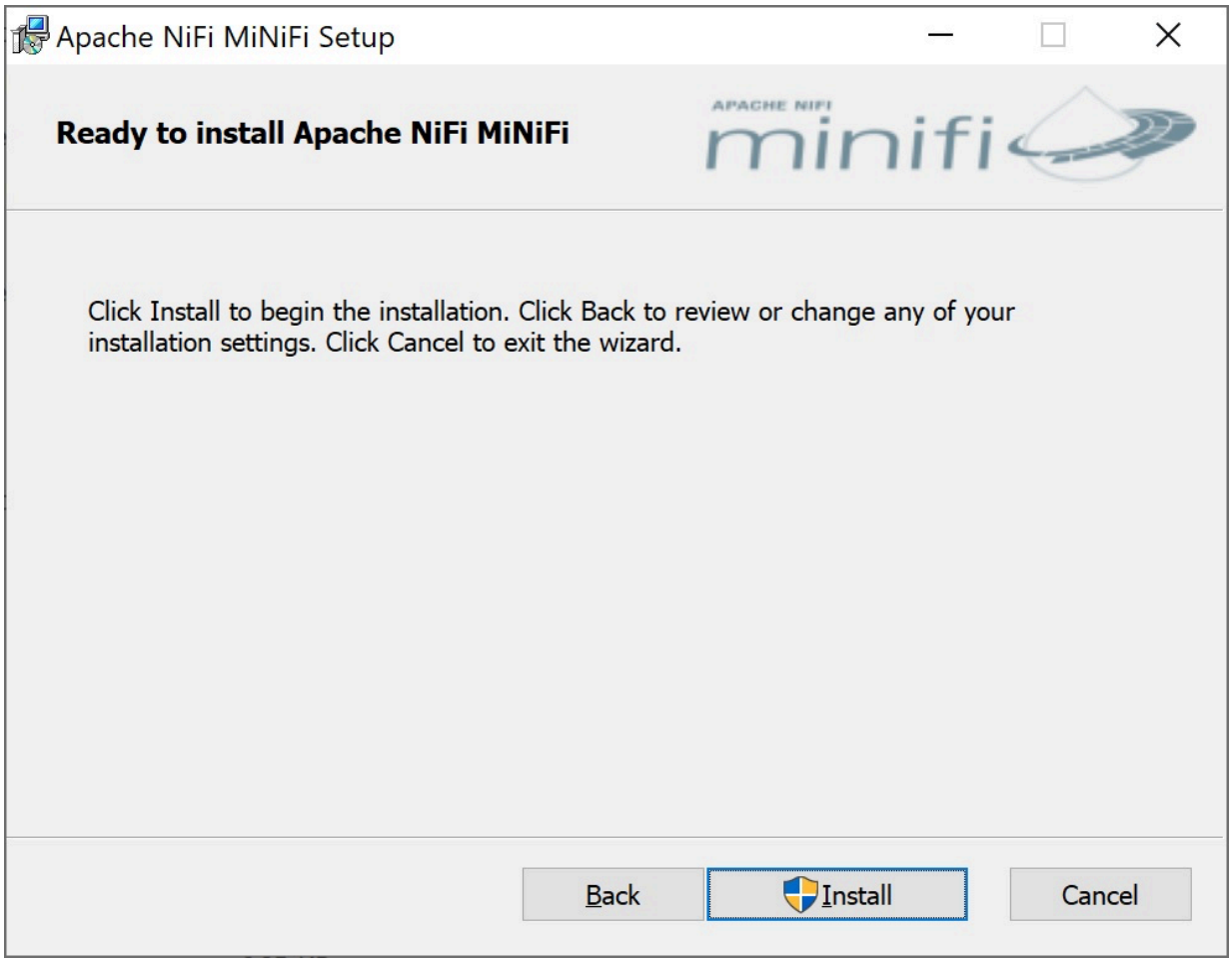


6. Click Next.
   The Agent Properties page appears as shown in the following image:



7. Check the Enable interactive Command and Control option.
   After you enable this option, you can edit 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.

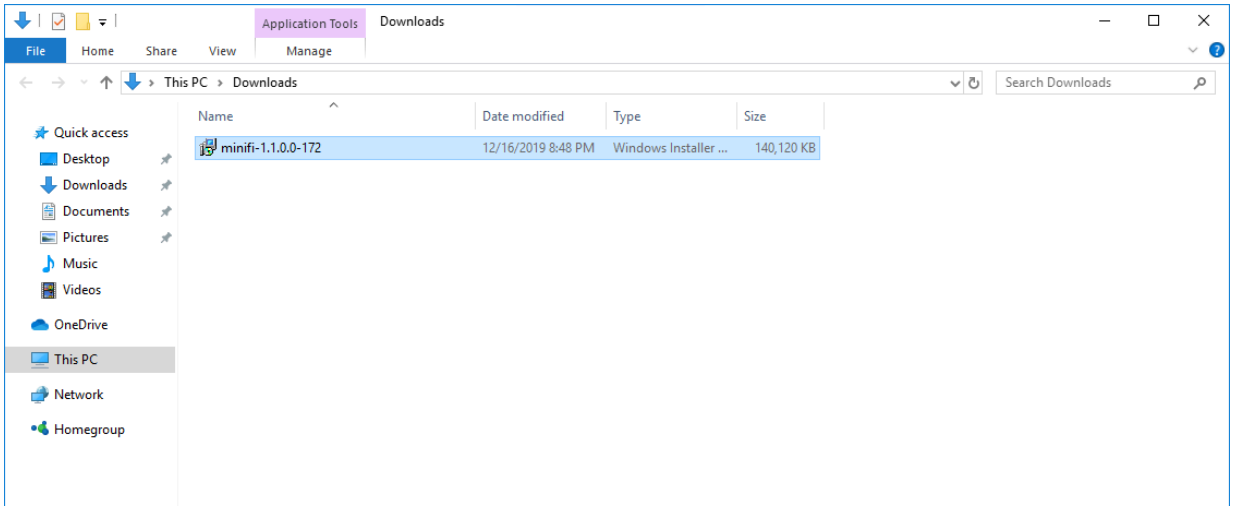


Install the Java agent on Windows using the MSI

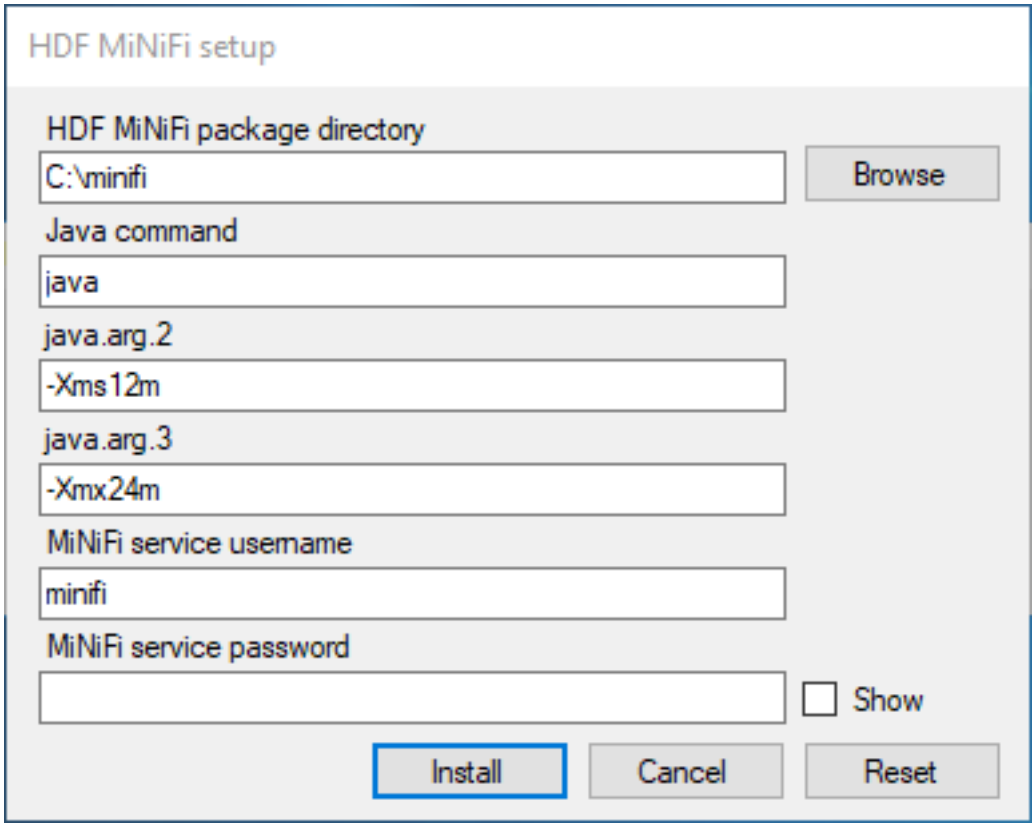
Learn how to install and configure the MiNiFi Java agent on Windows by using the provided MSI.

Procedure

1. Download the MiNiFi Java installer in your PC through the paywall.



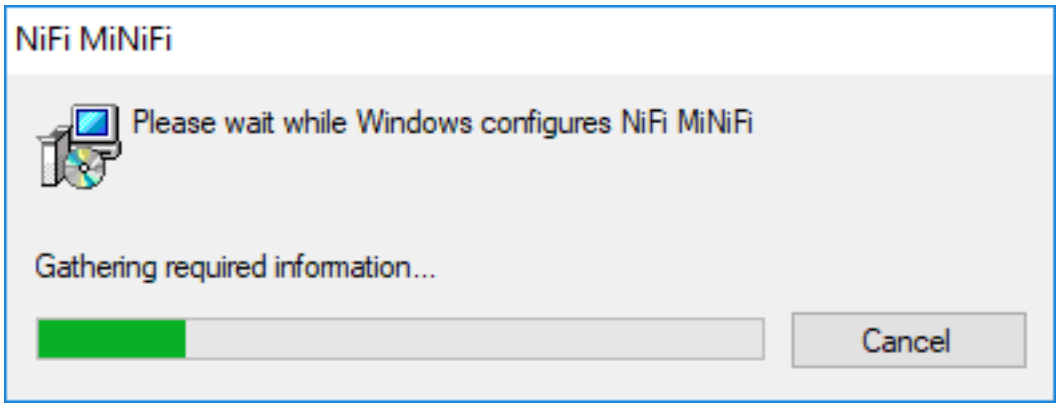
2. Double-click the executable file.
   The HDF MiNiFi setup wizard appears, as shown in the following image:



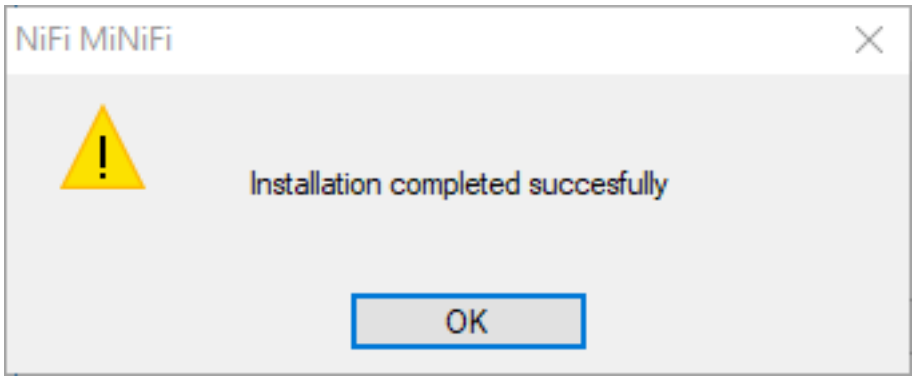
3. Configure the Java command option to the path of the java installation you would like to use.

   Note: By default, CEM assumes that java is on your system's path.

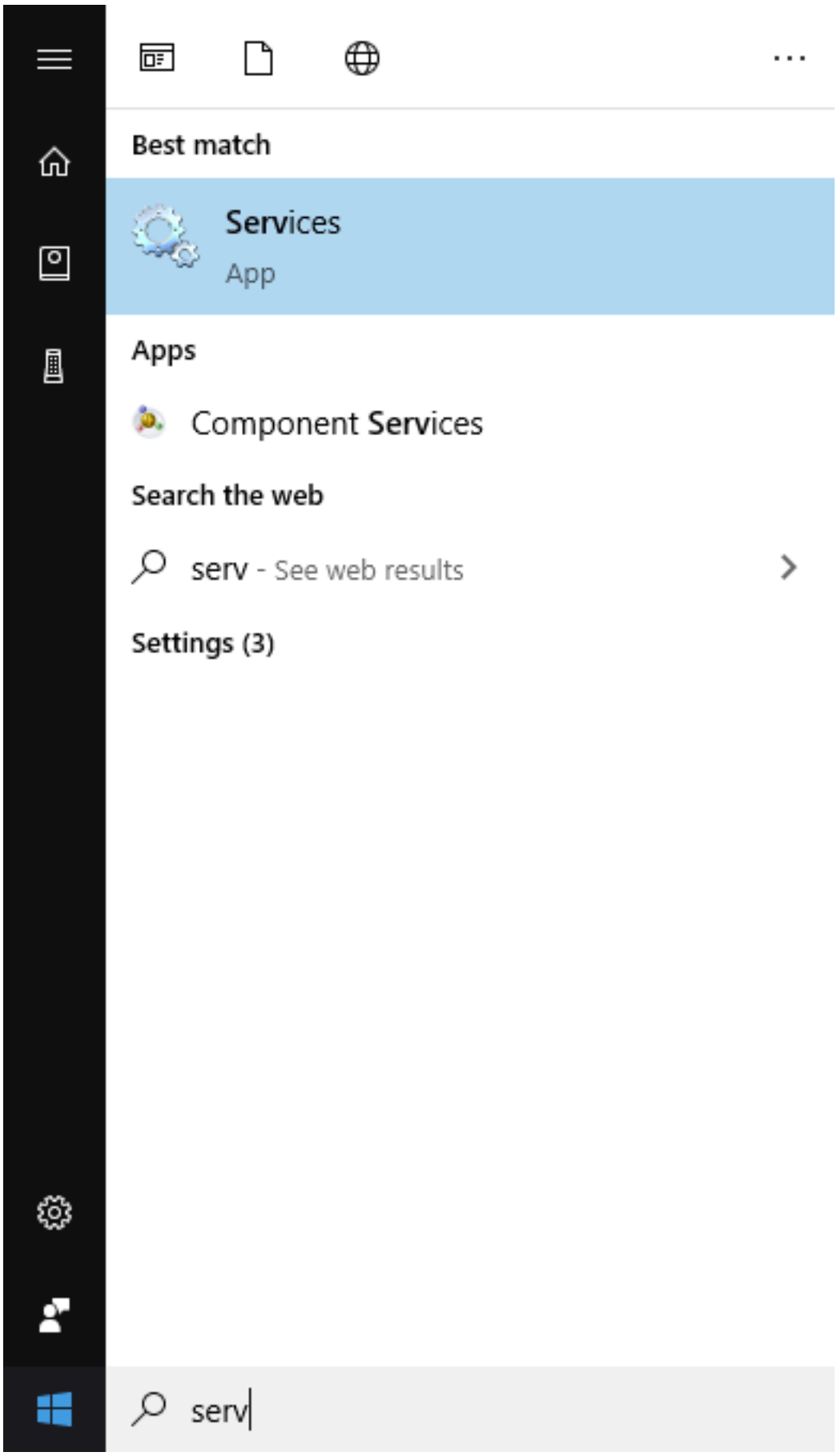
4. Click Install.
   Windows starts configuring NiFi MiNiFi, as shown in the following image:



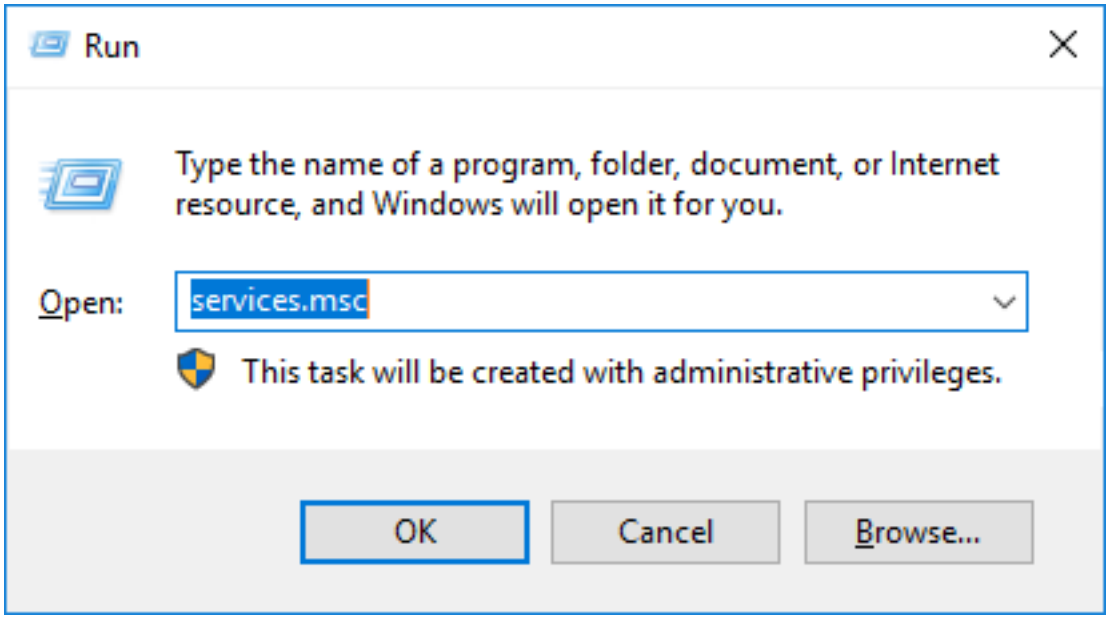
5. Click OK when installation completes.



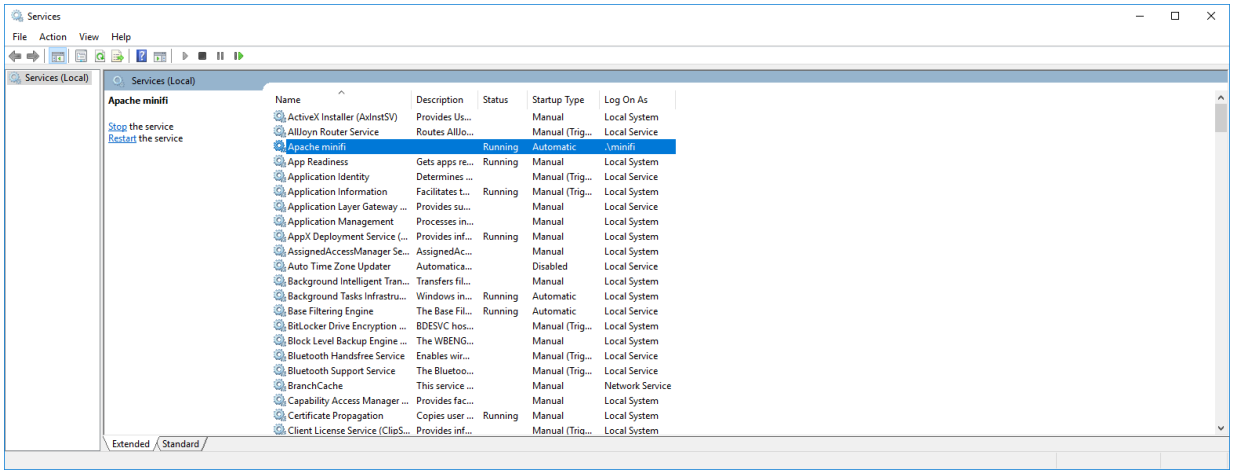
6. Search for services in the Start menu of your PC and open it, as shown in the following image:


Best match
Services
App
Apps
Component Services
Search the web
serv - See web results
Settings (3)
serv

You can also run services by pressing [Windows key + R] and then typing services.msc in the Open field, as shown in the following image:



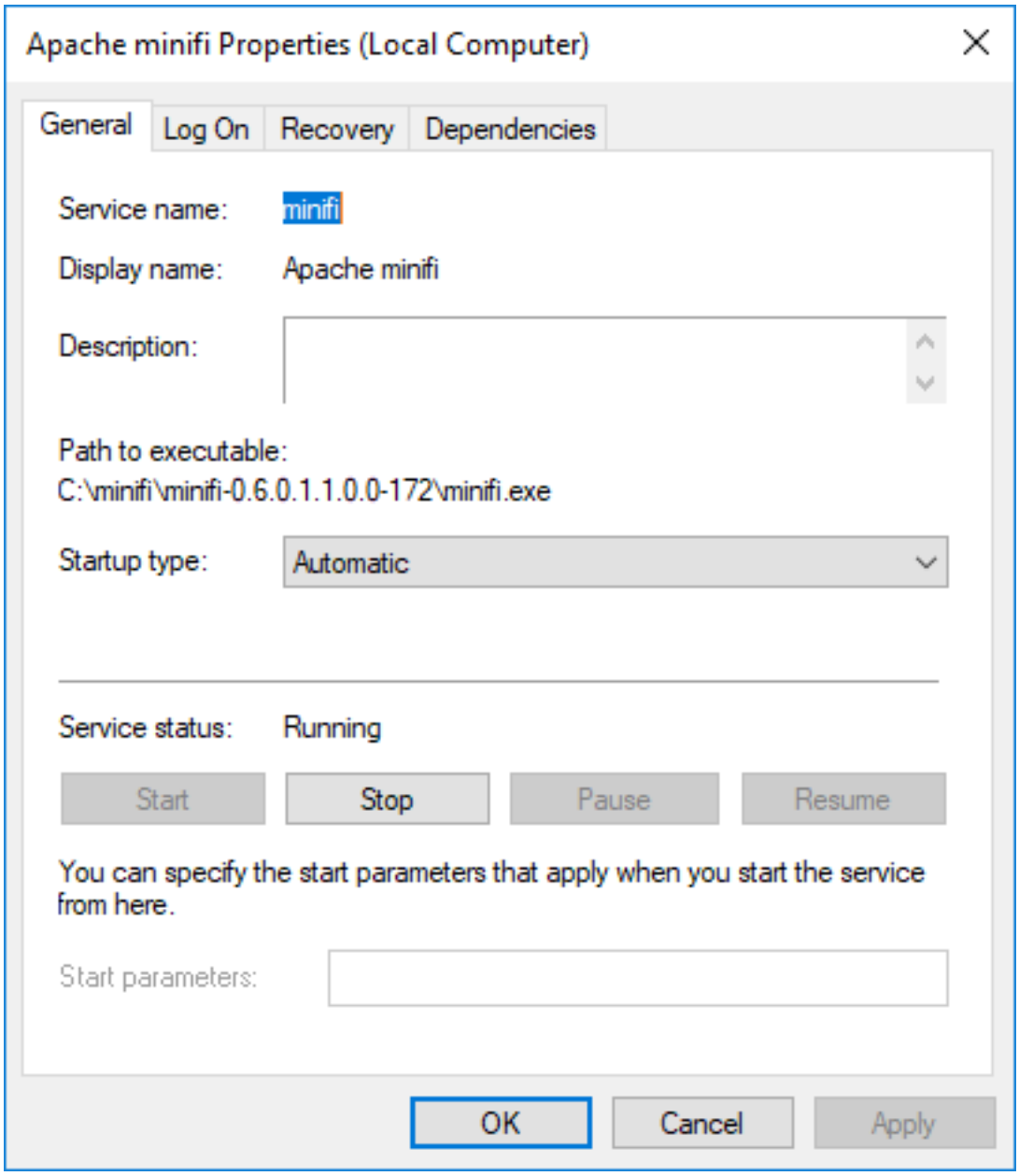
After you click OK, the Services window appears.



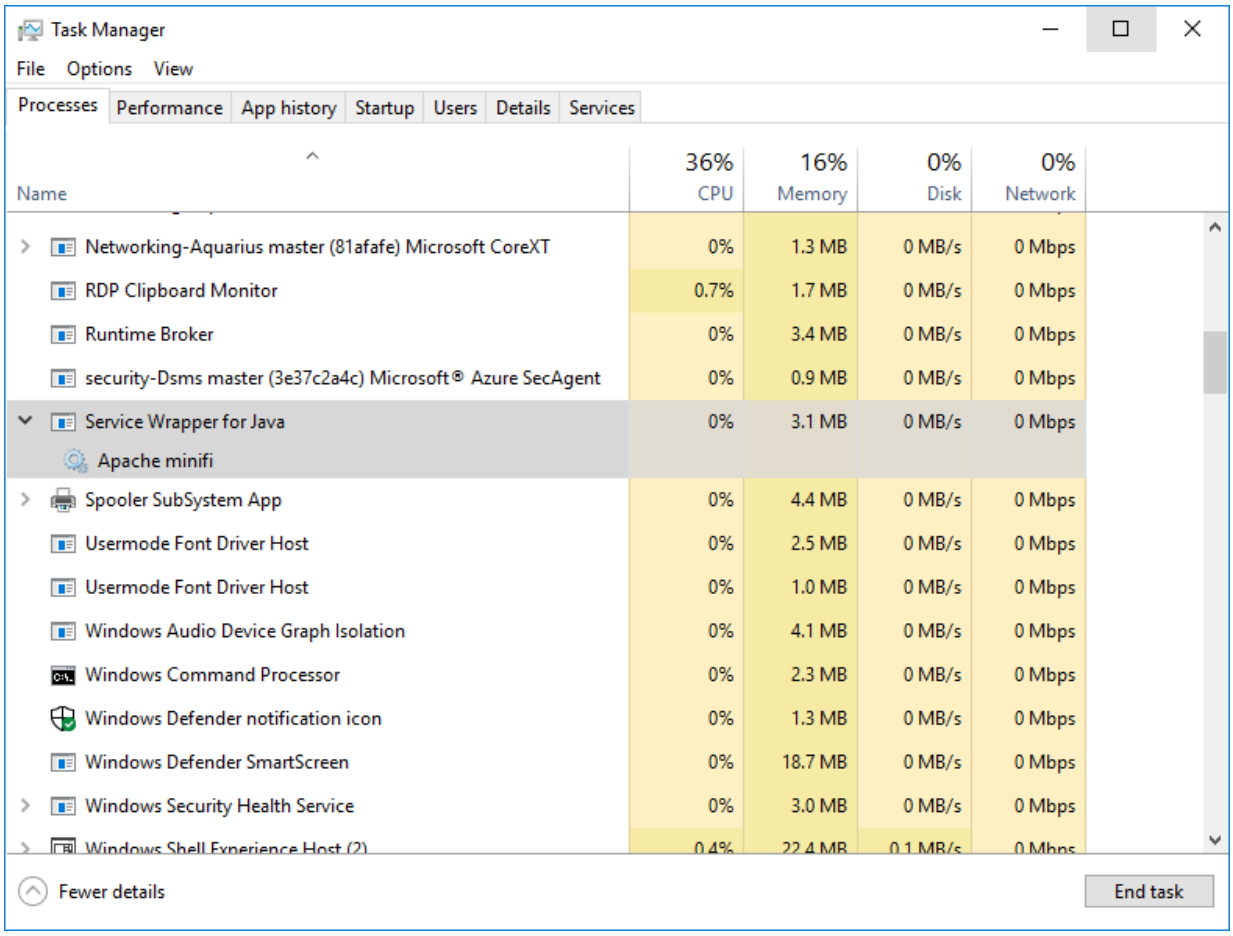
7. In the Services window, double-click Apache minifi.

   The Apache minifi Properties (Local Computer) window appears.

8. Check the services setup and click OK.



9. In the Task Manager window confirm whether the process is running.



10. Exit or close the window when done.

Configure your MiNiFi agents

After you install the MiNiFi agent, you need to update the configuration files.

About this task

If you are configuring a MiNiFi Java agent the configuration file is conf/bootstrap.conf. If you are configuring a MiNiFi C++ agent, the file is conf/minifi.properties. For more information on MiNiFi C++ agent system propties, see *MiNiFi C++ system properties*.

Procedure

1. From the MiNiFi home directory, open the appropriate configuration file.
2. Configure the Agent Class so that you can logically group MiNiFi instances according to their functionality. Specify the agent class:

```
nifi.c2.agent.class={AGENT_CLASS}
```

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

```
nifi.c2.agent.identifier={AGENT_ID}
```

4. Set the nifi.c2.enable property to true to inform MiNiFi that run time flow instructions will be received from EFM.

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

5. Configure your EFM Server endpoint:

```
nifi.c2.rest.url=http://{EFM_SERVER_IP}:10090/efm/api/c2-protocol/heartb
eat
nifi.c2.rest.url.ack=http://{EFM_SERVER_IP}:10090/efm/api/c2-protocol/ac
knowledge
```

6. Configure your heartbeat interval:

```
nifi.c2.agent.heartbeat.period={HEARTBEAT_INTERVAL}
```

7. If you are installing a MiNiFi C++ Agent, you may also configure metrics. Metrics are not yet available for the MiNiFi Java Agent.

```
nifi.c2.agent.protocol.class=RESTSender
```

   For supported processors for MiNiFi Java and MiNiFi C++, see *MiNiFi Java agent processor support* and *MiNiFi C++ Agent processor support*.

   .

Related Information

MiNiFi Java agent processor support
MiNiFi C++ agent processor support
MiNiFi C++ system properties

Communicating with secured EFM

To communicate with secured EFM, you need to configure additional properties.

About this task

If you are configuring a MiNiFi Java agent the configuration file is conf/bootstrap.conf. If you are configuring a MiNiFi C++ agent, the file is conf/minifi.properties.

Procedure

1. For the Java agent, configure the following additional properties in the bootstrap.conf file to communicate with a secured EFM:

```
nifi.c2.security.truststore.location=
nifi.c2.security.truststore.password=
nifi.c2.security.truststore.type=
nifi.c2.security.keystore.location=
nifi.c2.security.keystore.password=
nifi.c2.security.keystore.type=
nifi.c2.security.need.client.auth=
```

2. For the C++ agent, configure the following additional properties in the minifi.properties file to communicate 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=
```

Start MiNiFi agents

After MiNiFi is installed and configured, you can start it. Learn how to start your MiNiFi agent.

Procedure

1. From a terminal window, navigate to the MiNiFi installation directory.
2. To start MiNiFi in the foreground, enter:

```
bin/minifi.sh run
```

3. To start MiNiFi in the background, enter:

```
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

Configure MiNiFi C++ repositories

Learn how to configure and encrypt MiNiFi C++ repositories.

Persistent repositories, such as the Flow File repository, use a configurable path to store data. The repository locations and their defaults are defined below. By default the MINIFI_HOME environment variable is used. If this variable is not specified, you extrapolate the path and use the root installation folder. You may specify your own path in place of these defaults in the minifi.properties file.

```
nifi.provenance.repository.directory.default=${MINIFI_HOME}/provenance_repos
itory
nifi.flowfile.repository.directory.default=${MINIFI_HOME}/flowfile_reposi
tory
nifi.database.content.repository.directory.default=${MINIFI_HOME}/content_r
epository
```

You can also use a single database to store multiple repositories with the minifidb:// scheme. This could help with migration and centralize 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, and in that repository a logically separate subdatabase is created under the name flow file.

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

Repository encryption

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

You can provide the rocksdb-backed repositories 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 causes Flow File repository to use the specified 256 bit key. The second line triggers the generation of a random 256 bits key persisted back into conf/bootstrap.conf, which the Database Content repository then uses for encryption. In this way, you can request encryption while not bothering with 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.

In-memory repositories

Each of the repositories can be configured to be volatile (state is kept in memory and flushed upon restart). This can increase the performance but also cause data loss in case of restart while data is being processed by the agent.

To configure the repositories in the minifi.properties file:

```
# For Volatile Repositories:

nifi.flowfile.repository.class.name=VolatileFlowFileRepository
nifi.provenance.repository.class.name=VolatileProvenanceRepository
nifi.content.repository.class.name=VolatileContentRepository

# 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

# For NO-OP Repositories:
nifi.provenance.repository.class.name=NoOpRepository
```

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 also limits the number of events that can be stored. If you are limiting 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.

The content repository has a default option for minimal.locking to set to true. This attempts to use lock free structures. This may or may not be optimal as this requires additional searching of the underlying vector. This may be optimal for cases where max.count is not excessively high. In cases where object permanence is low within the repositories, minimal locking results in better performance. If there are many processors so that the content repository fills up quickly, performance may be reduced. In all cases a locking cache is used to avoid the worst case complexity of O(n) for the content repository, however, this caching is more heavily used when minimal.locking is set to false.

Encrypt sensitive data

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

MiNiFi comes with a tool called encrypt-config (encrypt-config.exe on Windows) which can be found in the bin directory of the installation, next to the main minifi binary. It enables the encryption of sensitive configuration properties in the minifi.properties file along with the encryption of the flow configuration (config.yml by default).

The security of the encryption depends on the security of the bootstrap.conf file, which contains 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 as specified to encrypt-config by the --minifi-home option.
- 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, or if not specified it defaults to <minifi home>/conf/config.yml.
- bootstrap file

  The file <minifi home>/conf/bootstrap.conf.
- sensitive property

  All property in the properties file that we 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. Generates a new encryption key.
2. Creates a bootstrap.conf file in your configuration directory, and write the encryption key to this file.
3. Encrypts the sensitive properties using this encryption key.
4. Adds a something.protected encryption marker after each encrypted property.

After running the tool, the bootstrap.conf and minifi.properties files 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
...
```

You should protect the bootstrap.conf file to make sure it is only readable by the user who runs MiNiFi.

Additional sensitive properties

By default, encrypt-config encrypts a (short) list of default sensitive properties. If you want more properties to be encrypted, you can 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 the 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.

Modifying sensitive properties

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

1. Replace the encrypted value with the new unencrypted value.
2. Delete the something.protected=... line which was 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.

Encryption of the flow definition

Pass the flag --encrypt-flow-config to encrypt-config so that it also encrypts the flow configuration file, not just the sensitive properties.

Updating the encryption key

If you want to change the encryption key, perform the following steps:

1. If the 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), otherwise you have to 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 you would like to use, add it to the bootstrap.conf file (add the line nifi.boo tstrap.sensitive.key=<your encryption 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 boot strap.conf.
4. Re-run the encrypt-config tool.

Take special care when changing the encryption key and the flow configuration is encrypted, so that you also re-encrypt it before deleting the old key (you get 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/bootstrap.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
```