Installation 1.1.1

Cloudera Edge Management Installation

Date published: 2020-02-07
Date modified: 2020-02-07

https://docs.cloudera.com/

Legal Notice

Contents

Before You Begin

System Requirements

Before you begin your installation, carefully review the system requirements to understand operating system, database, browser, and JDK support.

Components included in CEM 1.1.1

- EFM 1.1.1
- MiNiFi Java Agent 0.6.0
- MiNiFi C++ 0.7.0
- NiFi Registry 0.5.0

Operating System Support

Operating System	Version
RHEL/CentOS	7.0, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7
Debian	9
Ubuntu	14.04, 16.04, 18.04
Windows	8, 10, Server 2012, Server 2012 R2, Server 2016, Server 2019

Note: You may install MiNiFi C++ or Java Agents on Windows operating systems, but the EFM Server and NiFi Registry are not supported on Windows operating systems

JDK Support

JDK	Version
OpenJDK	JDK8
Oracle JDK	JDK8

Supported Databases

Database	Version
PostgreSQL	9.6
MySQL	5.6, 5.7

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)

Install Java

You should install Java on the machine on which you will install the EFM Server, NiFi Registry, and each machine onto which 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

Obtaining the CEM Software

To download the CEM software, you must first register for a Cloudera account if you have not already done so.

About this task

Perform the following steps to obtain the software bits:

Procedure

1. Go to https://www.cloudera.com/downloads.html.
2. Navigate to Cloudera DataFlow.
3. On the Cloudera DataFlow downloads page, select Download CEM under Edge Management.
4. When prompted, log in with your Cloudera account.
5. Specify whether you want to download the Source or Binary files, and then download:
   - EFM to download the EFM Server component. This also includes the NiFi Registry server component.
   - Either MiNiFi Java or MiNiFi C++ to download the MiNiFi agent appropriate for your operational objectives.
6. When downloading binary components, you have a choice of what package type to download. The platform-independent .tar.gz tarball packages are recommended, and that will be the package type used in the installation instructions in this guide.

Installing the EFM Server

Installing Databases

The EFM Server requrires 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 or PostgreSQL. These topics describe how to install and prepare MySQL and PostgreSQL.

Note:

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

Related Information

System Requirements

Installing and Configuring MySQL

You can install and configure MySQL. For supported database versions, see *System Requirements*.

Install MySQL

If you are using PostgreSQL instead of MySQL, you may skip these steps. See the instructions for PostgreSQL in the next section.

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 install 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-1.1.1/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;
```

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

You can install and configure PostgreSQL. For supported database versions, see *System Requirements*.

Install PostgreSQL

If you are using MySQL instead of PostgreSQL, you may skip these steps. See the instructions for MySQL in the previous section.

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

```
yum install https://yum.postgresql.org/9.6/redhat/rhel-7-x86_64/pgdg-red
hat96-9.6-3.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';
```

Install NiFi Registry

Using CEM requires a NiFi Registry instance to store your dataflows. You can configure 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
```

Install EFM Server

Procedure

1. Extract the 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 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.

Configure the EFM Server

Once the 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

You need to set the efm.encryption.password property.

The efm.encryption.password property specifies a master password used for encrypting sensitive data saved to the 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

You should ensure that the following ports are available for the EFM Server and its components.

Component	Port number
EFM Server HTTP	10080
EFM Server Constraint application protocol (CoAP)	8989

Start the EFM Server

Once you have installed and configured the EFM Server you can start it. In general, you should start the EFM Server before you install and configure your MiNiFi Agents.

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}:10080/efm/
```

Install the MiNiFi Agents

Before you Begin MiNiFi Agent Installation on Windows

Go through the following details 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.

Configure your MiNiFi Agents

Once you have installed the MiNiFi agent, 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.

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. Configure your EFM Server endpoint:

```
nifi.c2.rest.url=http://{EFM_SERVER_IP}:10080/efm/api/c2-protocol/heartb
eat
nifi.c2.rest.url.ack=http://{EFM_SERVER_IP}:10080/efm/api/c2-protocol/ac
knowledge
```

5. Configure your heartbeat interval:

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

6. 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

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:

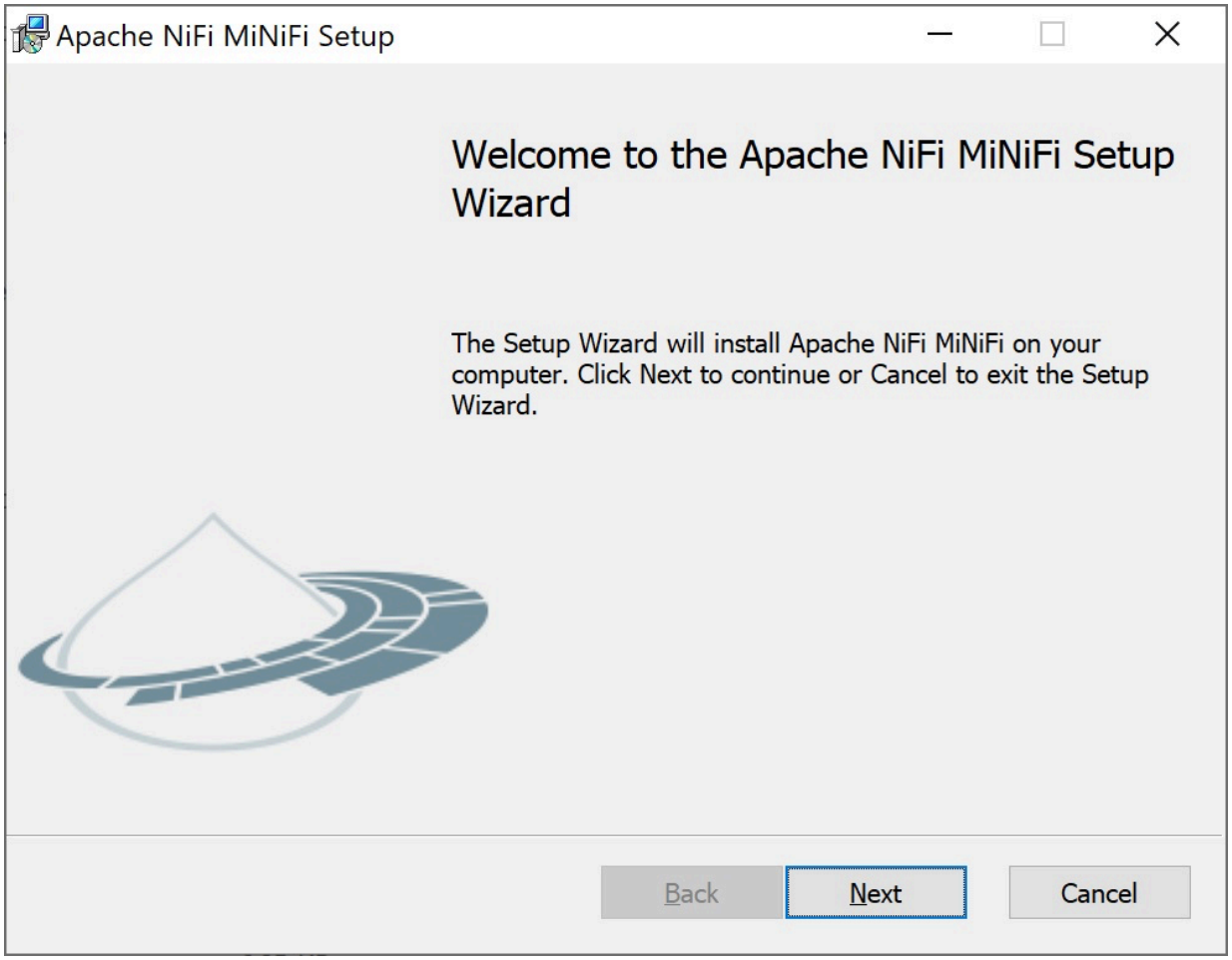
```
# Security Properties #
# enable tls #
nifi.remote.input.secure=true
# if you want to enable client certificate base authorization #
nifi.security.need.ClientAuth=true
# setup the client certificate and private key PEM files #
nifi.security.client.certificate=
nifi.security.client.private.key=
# 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=
```

Configure your MiNiFi Agents in Windows

Once you have installed the MiNiFi agent, update the configuration files.

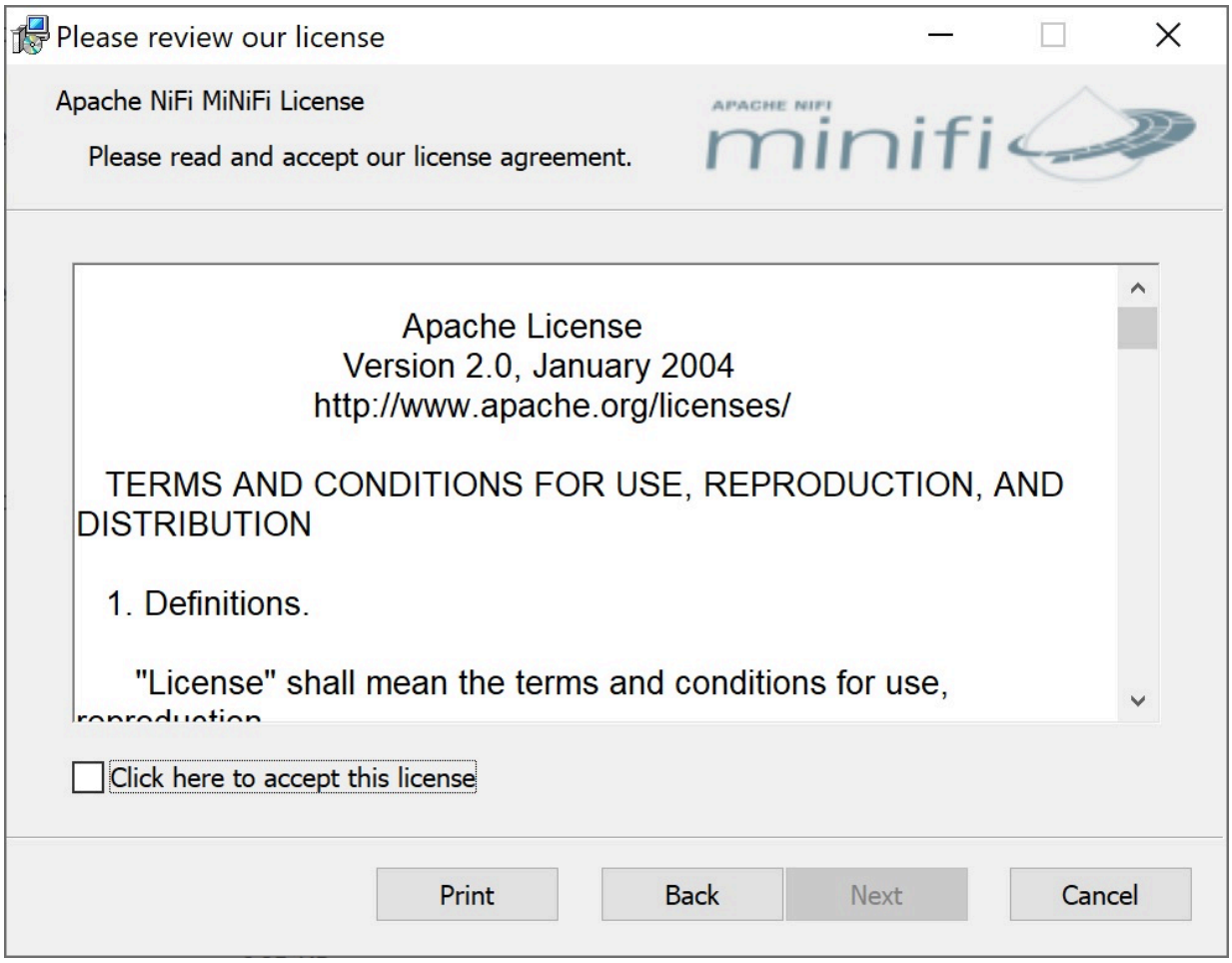
Configure MiNiFi C++ Agent

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



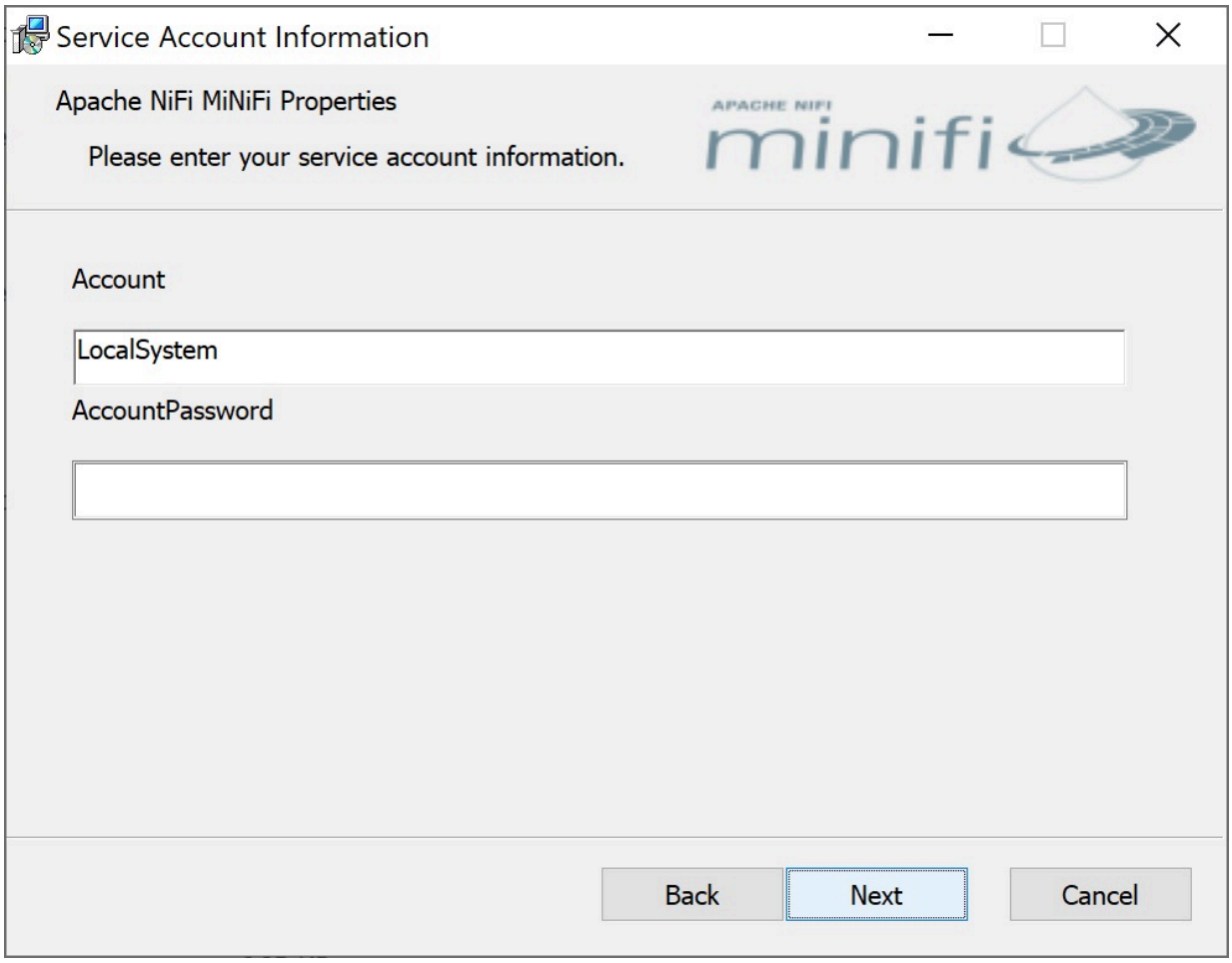
The Apache NiFi MiNiFi License page appears.

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



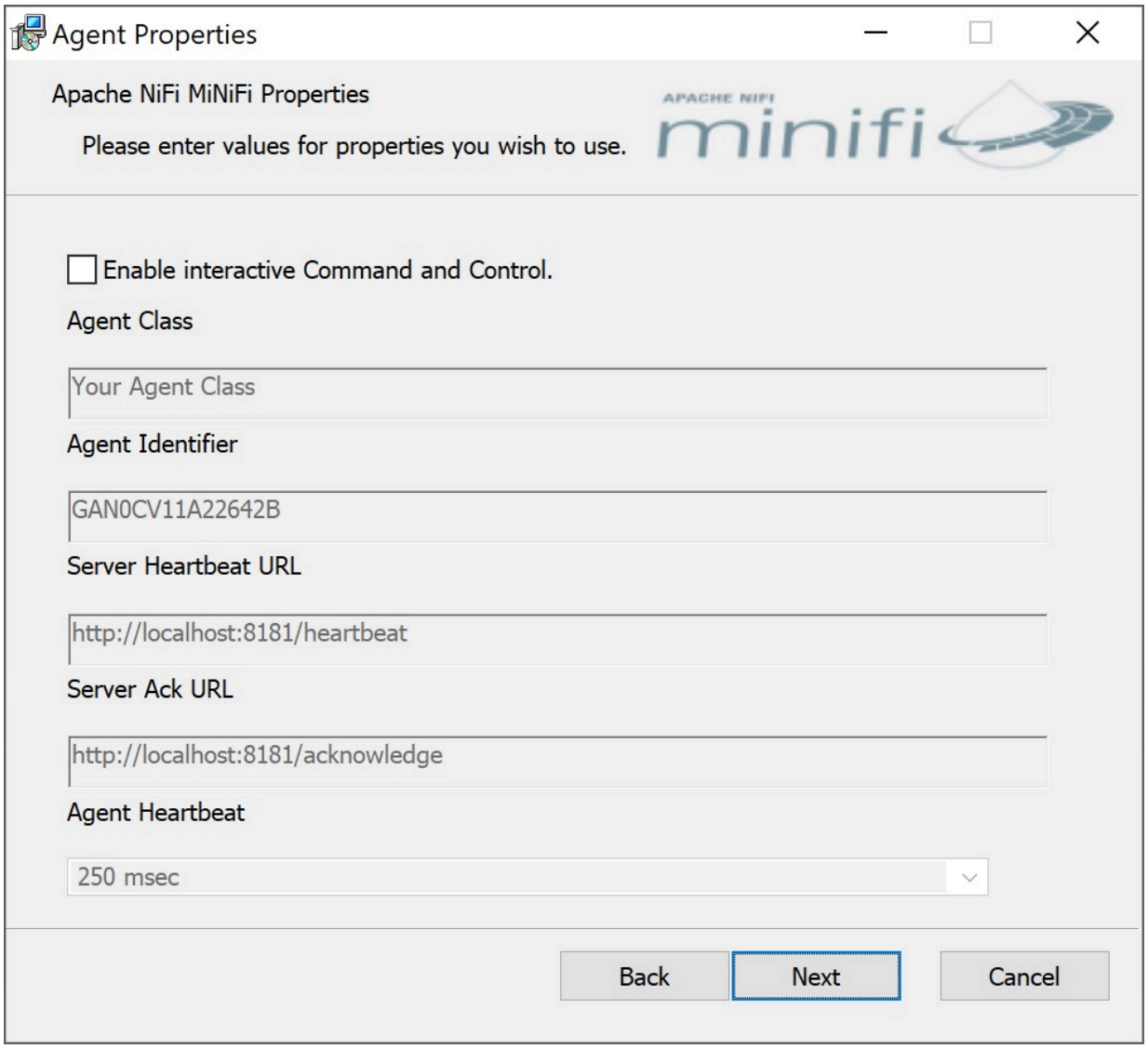
The Service Account Information page appears.

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



4. Click Next.

   The Agent Properties page appears as shown in the following image:



5. Check the Enable interactive Command and Control option.

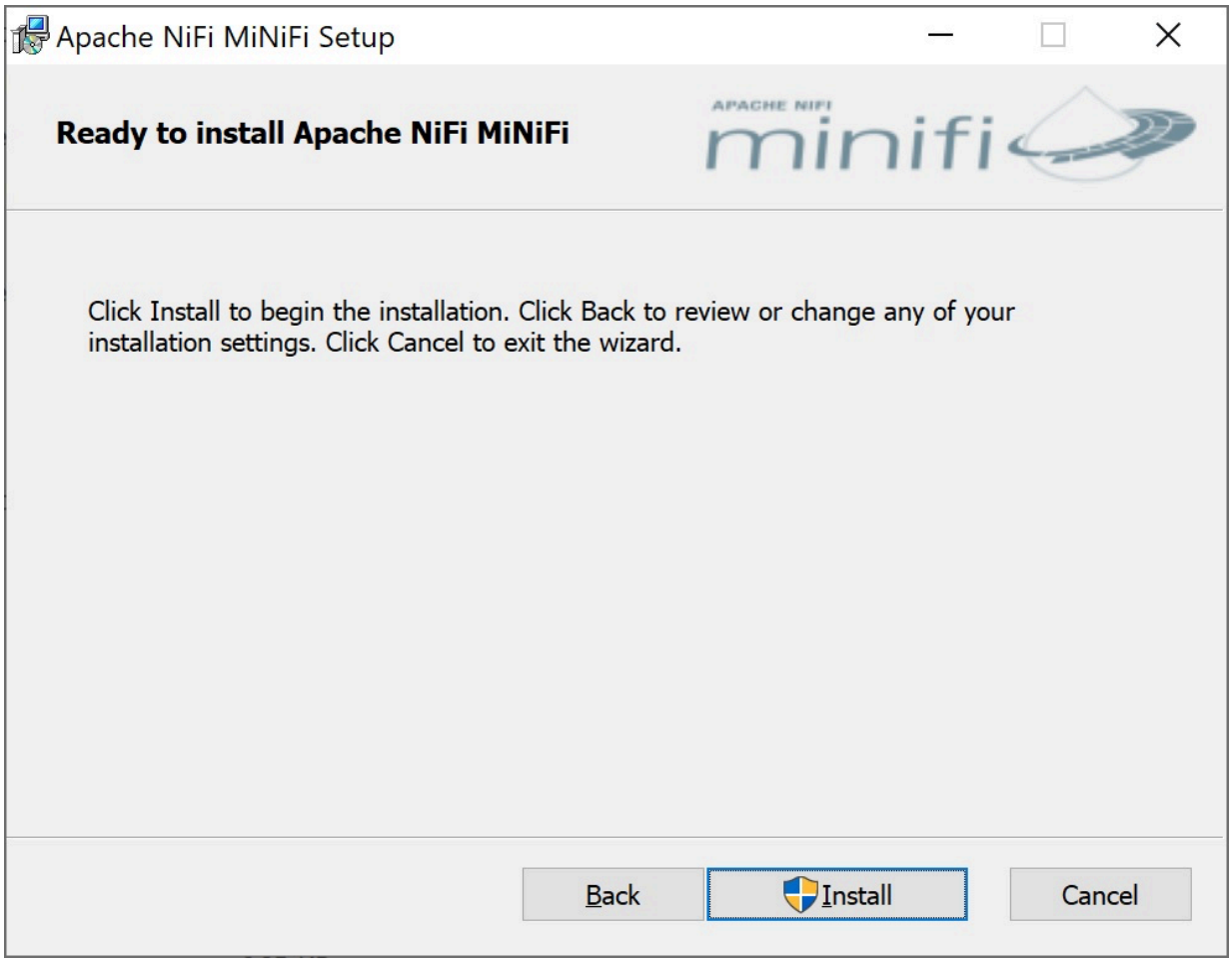
   After you enable this option, you can edit your properties.

6. Edit the following properties:

   - Agent Class
   - Agent Identifier
   - Server Heartbeat URL
   - Server Ack URL
   - Agent Heartbeat

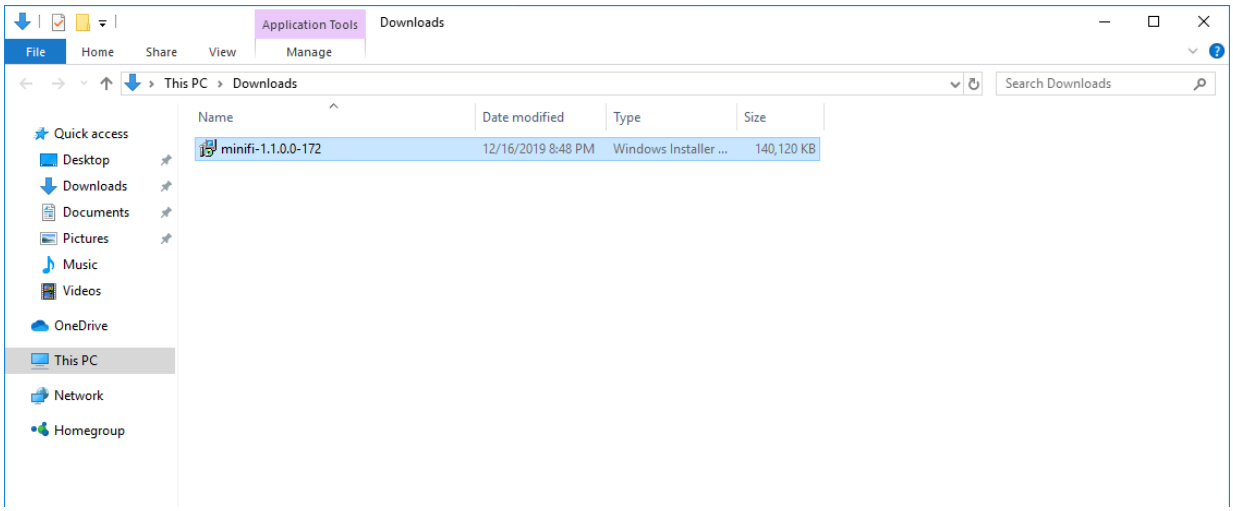
7. Click Next.

8. Click Install.



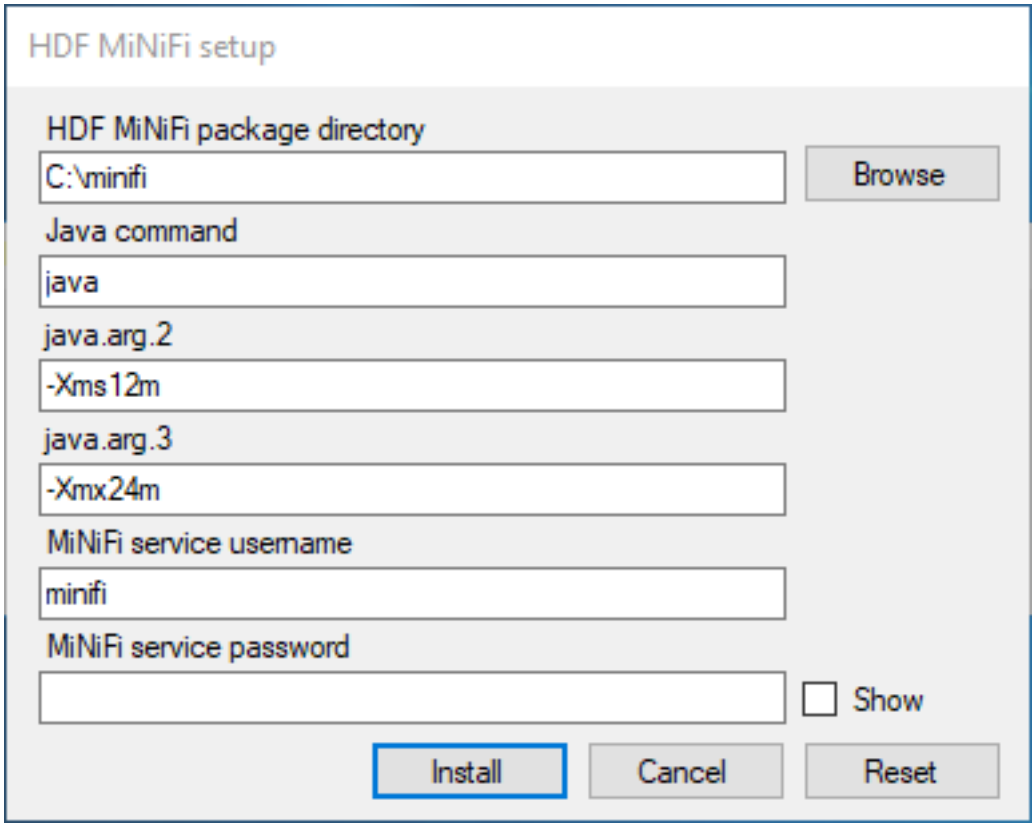
Configure MiNiFi Java Agent

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:

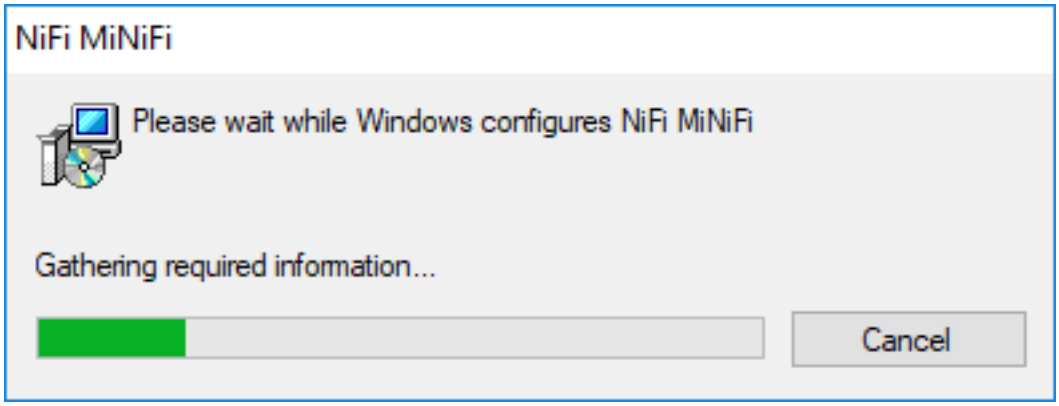


3. Configure the Java command option to the path of the java installation you would like to use.

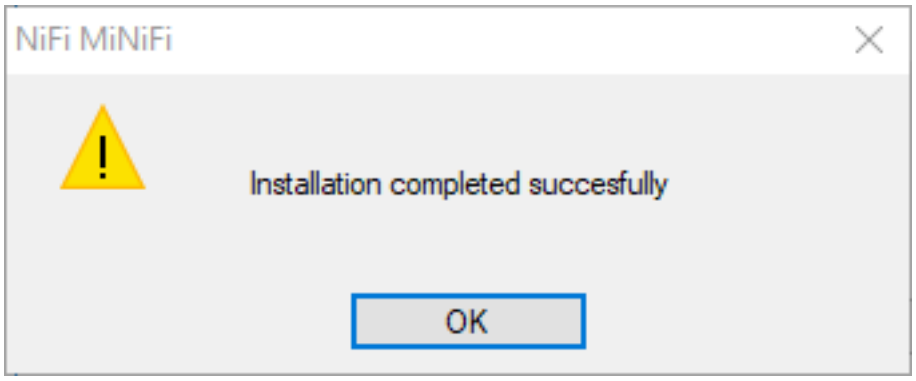
   Note: By default, CEM assumes that java is on your system's path.

4. Click Install.

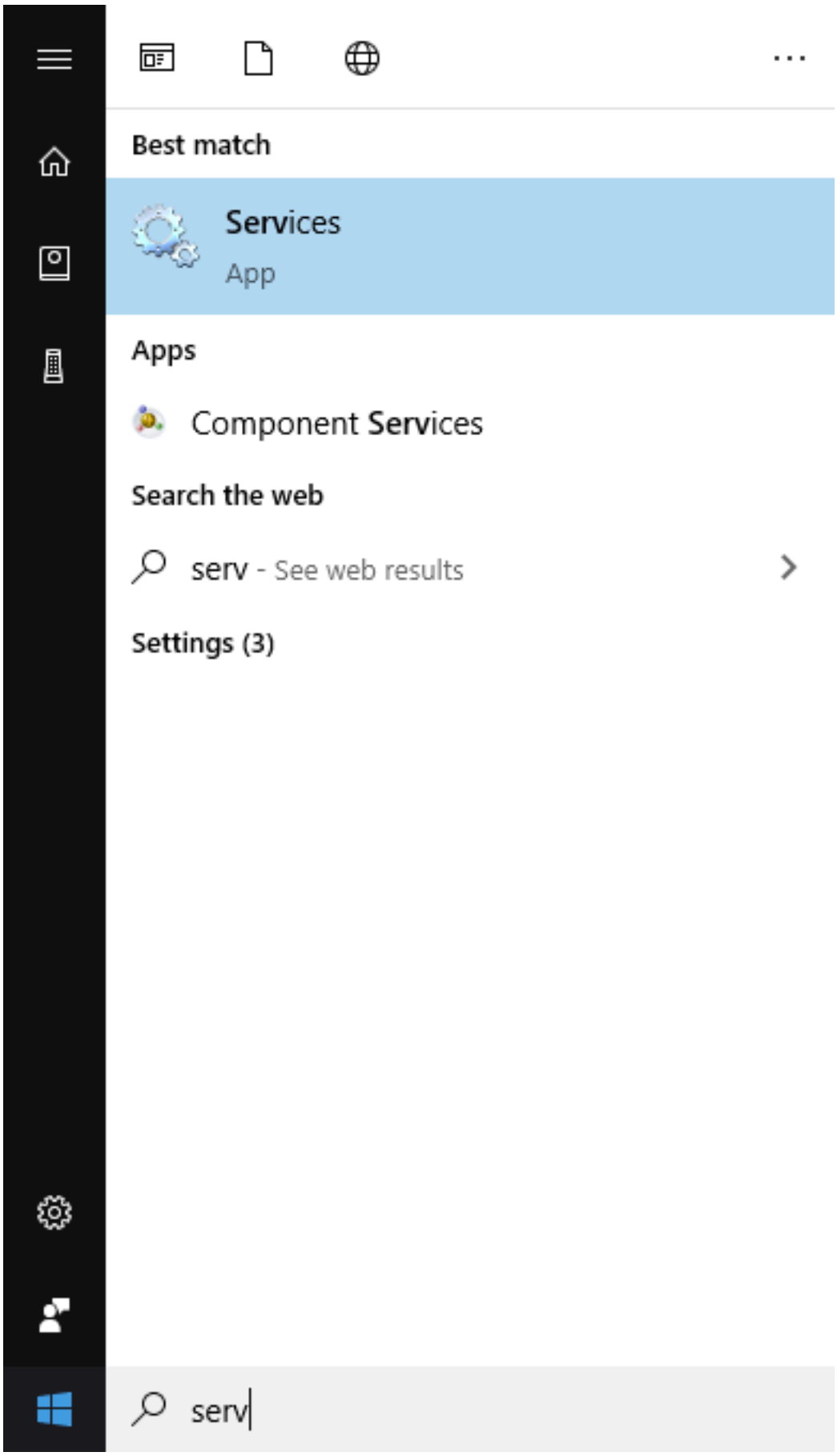
   Windows starts configuring NiFi MiNiFi, as shown in the following image:



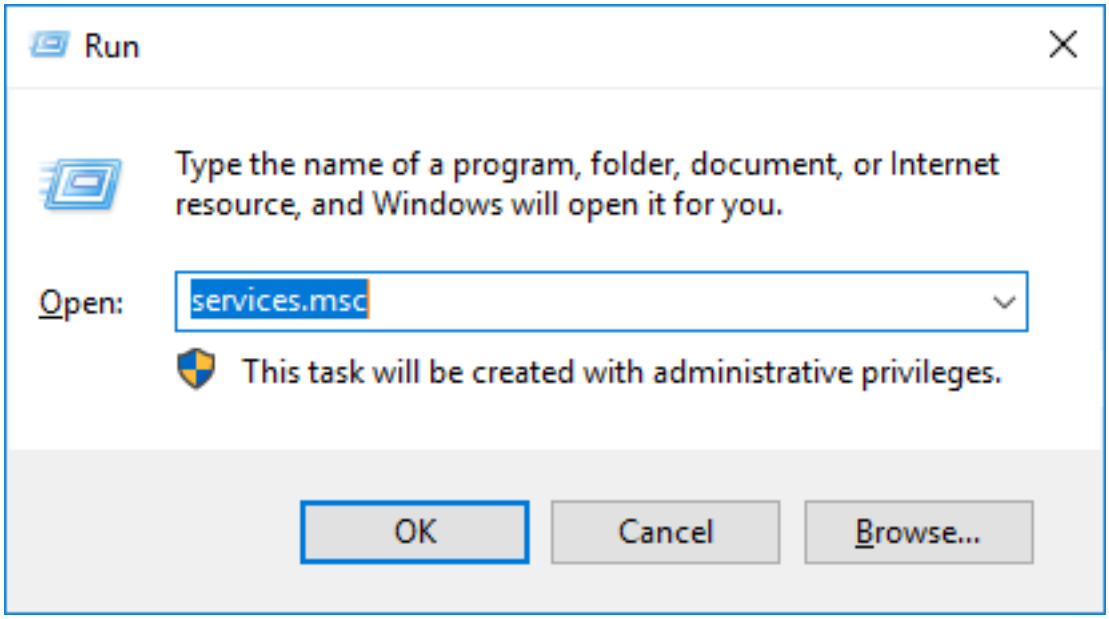
5. Click OK when installation completes.



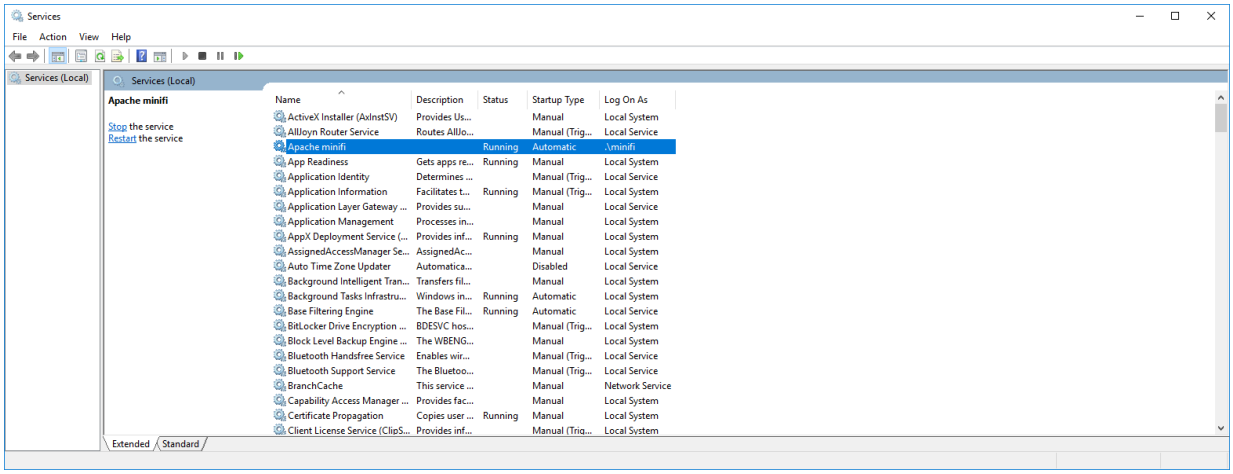
6. Search for services in the Start menu of your PC and open it, as shown in the following image:


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:



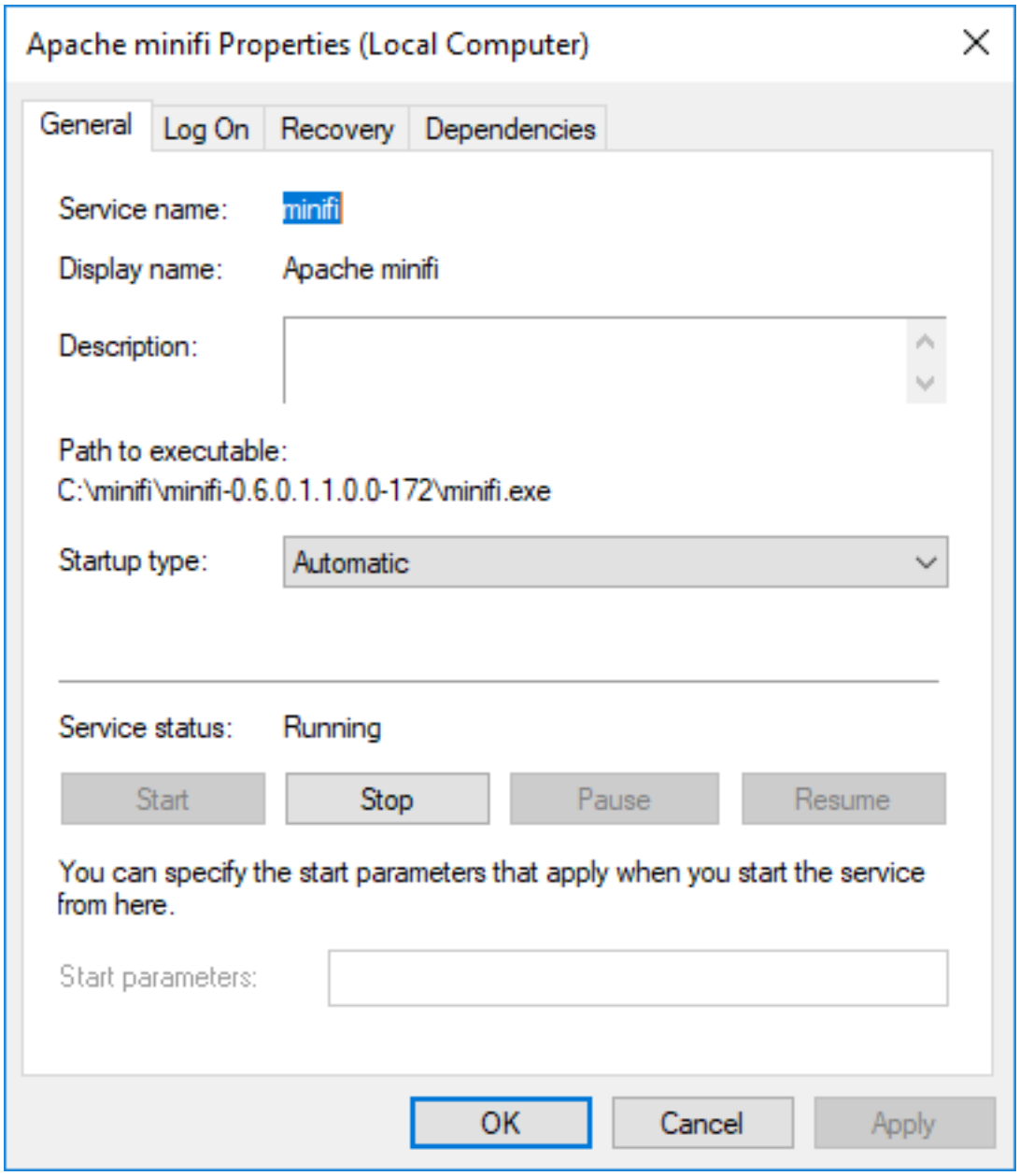
After you click OK, the Services window appears.



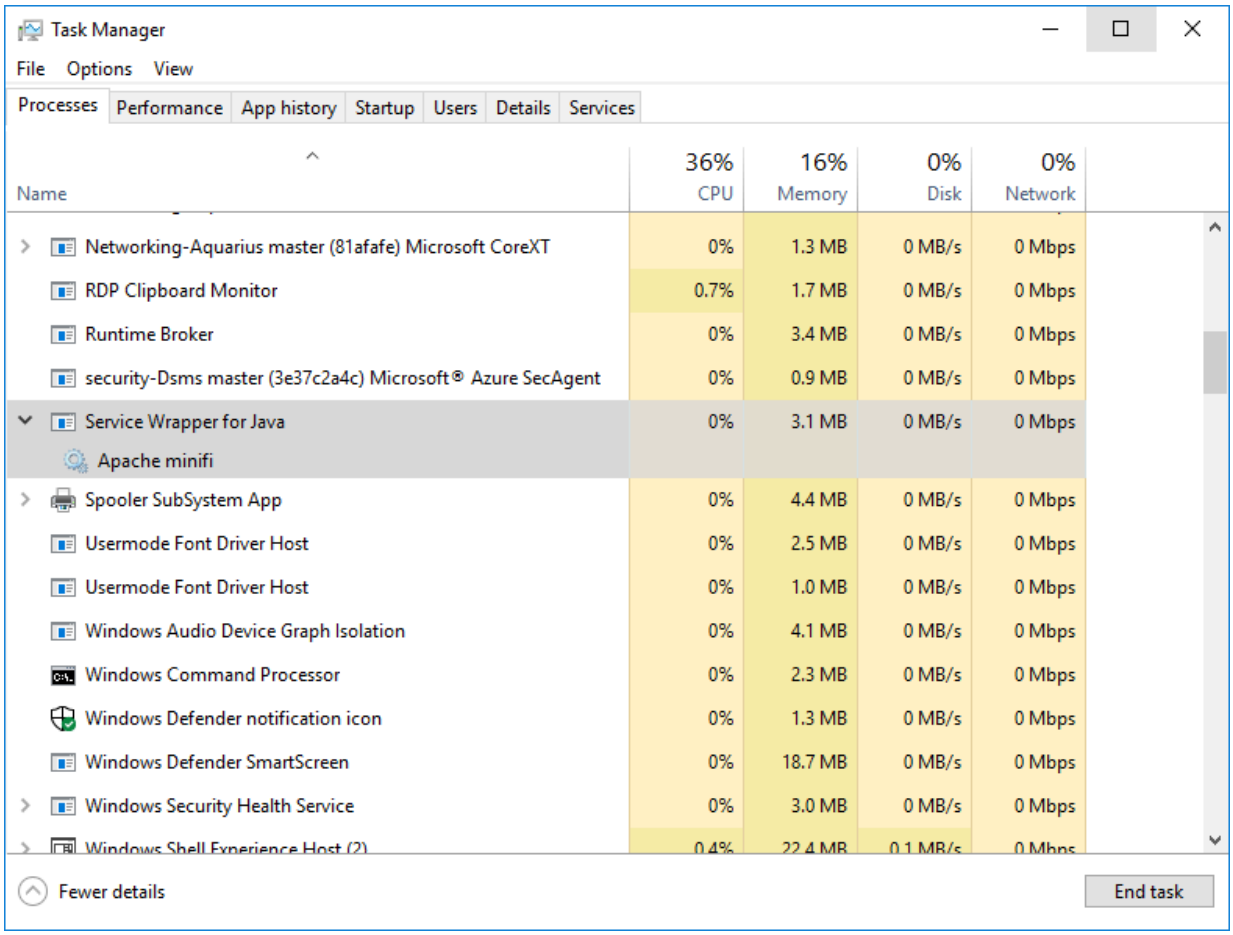
7. In the Services window, double-click Apache minifi.

   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.

Start MiNiFi Agents

Once MiNiFi is installed and configured, you can start it.

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 Security

Configure EFM Server for TLS

You can configure the server to authenticate users based on a client certificate provided for TLS mutual authentication. The server's TLS settings, including what certificates it will trust, are configured using the efm.server.ssl.* prefixed properties in the efm.properties file.

Procedure

1. Open the efm.properties file.

   The properties pertaining to TLS configuration are the following:

   ```
   efm.server.ssl.enabled=false
   efm.server.ssl.keyStore=/path/to/keystore.jks
   efm.server.ssl.keyStoreType=jks
   efm.server.ssl.keyStorePassword=
   efm.server.ssl.keyPassword=
   efm.server.ssl.trustStore=/path/to/truststore.jks
   efm.server.ssl.trustStoreType=jks
   efm.server.ssl.trustStorePassword=
   efm.server.ssl.clientAuth=WANT
   ```

2. Change efm.security.user.certificate.enabled=false to efm.security.user.certificate.enabled=true.
3. Change efm.server.ssl.clientAuth=WANT to efm.server.ssl.clientAuth=NEED.

Configure MiNiFi Agent TLS

You can configure MiNiFi Agent TLS by updating 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 configuration file is conf/minifi.properties.

Procedure

1. Open your MiNiFi Agent configuration file in a text editor.
2. Edit the security properties for Java. For example:

```
# Security Properties #
# These properties take precedence over any equivalent properties specif
ied in config.yml file #
nifi.minifi.security.keystore=
nifi.minifi.security.keystoreType=
nifi.minifi.security.keystorePasswd=
nifi.minifi.security.keyPasswd=
nifi.minifi.security.truststore=
nifi.minifi.security.truststoreType=
nifi.minifi.security.truststorePasswd=
nifi.minifi.security.ssl.protocol=
nifi.minifi.sensitive.props.key=
nifi.minifi.sensitive.props.algorithm=
nifi.minifi.sensitive.props.provider=
```

Note: If you specify any of the security properties and the nifi.minifi.sensitive.props.key property remains empty, MiNiFi fails to start with the following error message:

```
Failed to parse the config YAML while creating the nifi.properties
```

3. Edit the security properties for C++. For example:

```
# 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=./conf/client.pem
nifi.security.client.private.key=./conf/client.pem
# 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=./conf/nifi-cert.pem

# if you do not want to enable client certificate base authorization #
nifi.security.need.ClientAuth=false
```

You have the option of specifying an SSL Context Service definition for the RPGs instead of the preceding properties. This links to a corresponding SSL Context Service defined in the flow.

To do this, specify the SSL Context Service Property in your RPGs and link it to a defined controller service. For example:

```
Remote Processing Groups:
```

```
- name: NiFi Flow
  id: 2438e3c8-015a-1000-79ca-83af40ec1998
  url: http://127.0.0.1:8080/nifi
  timeout: 30 secs
  yield period: 5 sec
  Input Ports:
      - id: 2438e3c8-015a-1000-79ca-83af40ec1999
        name: fromnifi
        max concurrent tasks: 1
        Properties:
            SSL Context Service: SSLServiceName
  Output Ports:
      - id: ac82e521-015c-1000-2b21-41279516e19a
        name: tominifi
        max concurrent tasks: 2
        Properties:
 SSL Context Service: SSLServiceName
Controller Services:
- name: SSLServiceName
  id: 2438e3c8-015a-1000-79ca-83af40ec1974
  class: SSLContextService
  Properties:
      Client Certificate: <client cert path>
      Private Key: < private key path >
      Passphrase: <passphrase path or passphrase>
      CA Certificate: <CA cert path>
```

If you do not take this approach, the preceding properties will be used for TCP and secure HTTPS communications.

Note: We recommend not to use the properties approach listed above when the agents use C2.