

CFM 2.0.4

## CFM Security

Date published: 2020-06-30

Date modified: 2020-10-02

# CLOUDERA

<https://docs.cloudera.com/>

# Legal Notice

© Cloudera Inc. 2024. 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

<b>CFM Security.....</b>	<b>5</b>
<b>Authentication.....</b>	<b>5</b>
Kerberos Authentication.....	5
LDAP Authentication.....	6
Identity-Mapping Properties.....	7
<b>Authorization.....</b>	<b>8</b>
Ranger Authorization.....	8
Understanding the Ranger Authorization Process for CFM.....	8
Before you begin.....	8
Add user to a pre-defined Ranger access policy.....	9
Create a custom Ranger access policy.....	10
Authorization example.....	15
Predefined Ranger Access Policies for Apache NiFi.....	15
Predefined Ranger Access Policies for Apache NiFi Registry.....	16
File-Based Authorization.....	17
LDAP Integration.....	18
LDAP and Ranger Policies.....	18
LDAP and File-Based Policies.....	19
LDAP User Group Provider Properties.....	20
<b>Migrate file-based authorization to Ranger.....</b>	<b>21</b>
Migrate NiFi File-Based Authorization to Ranger.....	22
Migrate NiFi Registry File-Based Authorization to Ranger.....	22
<b>TLS Configuration.....</b>	<b>22</b>
Enable Auto-TLS.....	22
Manually configure TLS.....	23
TLS certificate requirements and recommendations.....	23
Configure TLS encryption manually for NiFi and NiFi Registry.....	24
NiFi TLS Properties.....	25
NiFi Registry TLS Properties.....	26
<b>Integrate NiFi and Atlas.....</b>	<b>27</b>
Manually Integrate with Atlas when Auto-TLS is not Enabled.....	27
Manually Integrate with Atlas when Auto-TLS is Enabled.....	27
<b>Integrate NiFi and NiFi Registry with Knox.....</b>	<b>28</b>
<b>Enhance or Overwrite Properties in Cloudera Manager.....</b>	<b>28</b>

<b>Pairing LDAP with a Composite Group Provider.....</b>	<b>29</b>
<b>Default Ports for NiFi and NiFi Registry.....</b>	<b>30</b>

## CFM Security

Flow management users are authenticated automatically when they log into CDP. Other aspects of security such as enabling Auto-TLS, Kerberos, and managing access policies depend on the way the SDX and compute clusters are created.

Cloudera recommends the following security options:

- Enable Auto-TLS.
- Enable Kerberos.
- Use Apache Atlas for dataset level lineage graphs.
- Use Apache Ranger to authorize NiFi and NiFi Registry users.
- Use Knox as a single entry point to securely access all NiFi and NiFi Registry nodes, and switch nodes if one fails.

## Authentication

TLS/SSL must be enabled before NiFi can support any form of user authentication. The primary mechanisms of authenticating to NiFi and NiFi Registry in a cluster are Kerberos and LDAP.

### Kerberos Authentication

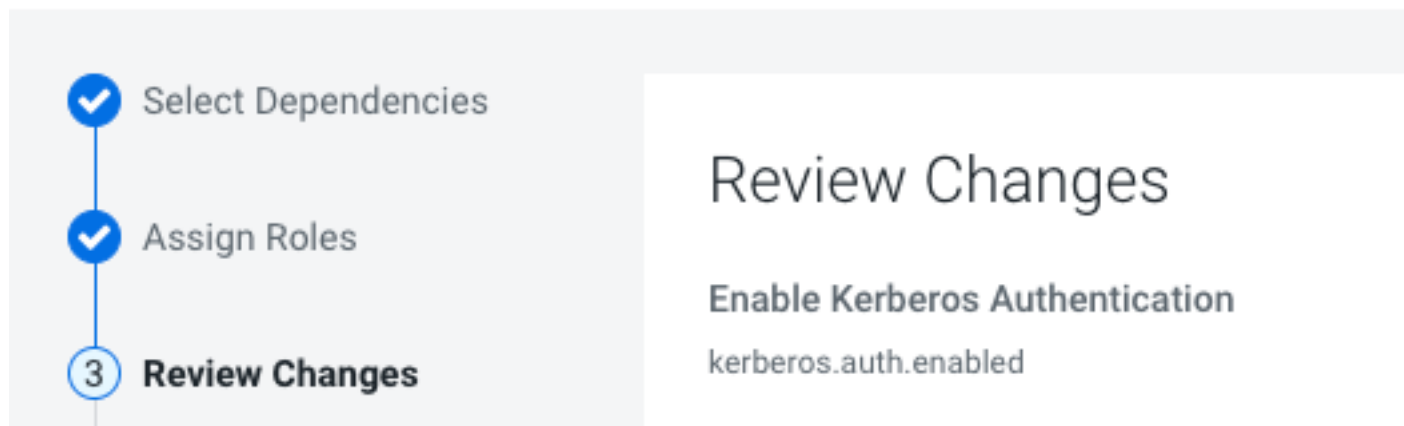
Authenticate your cluster by enabling Kerberos.



**Important:** You must enable TLS/SSL for NiFi to support authentication.

When you add NiFi or NiFi Registry to a kerberized environment, Cloudera Manager provides the Enable Kerberos Authentication option for the NiFi and NiFi Registry services.

## Add NiFi Service to Cluster 1



The Enable Kerberos Authentication option is the UI label for the `kerberos.auth.enabled` parameter.

By default, the option is checked (parameter set to true) when you add NiFi or NiFi Registry and selecting a dependent service that is kerberized.

The parameter enables the Kerberos Login Identity Provider which allows access to the NiFi/NiFi Registry UI using a Kerberos principal and password.

When the option is enabled, NiFi and NiFi Registry use Kerberos to interact with external systems such as Ranger and Atlas. If the option is not enabled in a kerberized environment, NiFi and NiFi Registry fail to authenticate to external systems.

Alternatively, when Kerberos is enabled you may also authenticate to the KDC from the command line and then configure your browser to forward your credentials to authenticate through SPNEGO.

## LDAP Authentication

After you install NiFi or NiFi Registry, you can enable LDAP authentication.



**Important:** You must enable TLS/SSL for NiFi to support authentication.

In a kerberized environment, enabling the LDAP Login Identity Provider takes precedence over the Kerberos Login Identity Provider.

Set the following required LDAP parameters for NiFi:

LDAP Parameters for NiFi	Sample Value
Enable TLS/SSL for NiFi Node	Checked
LDAP Enabled	Checked
Login Identity Provider: Default LDAP Provider Class	org.apache.nifi.ldap.LdapProvider
Initial Admin Identity	admin
Login Identity Provider ID	ldap-provider
LDAP Authentication Strategy	SIMPLE
LDAP Manager DN	uid=admin,ou=people,dc=hadoop,dc=apache,dc=org
LDAP Manager Password	admin-password
LDAP URL	ldap://<ldap-hostname>:33389
LDAP User Search Base	ou=people,dc=hadoop,dc=apache,dc=org
Login Identity Provider: Default LDAP User Search Filter	uid={0}
Login Identity Provider: Default LDAP Identity Strategy	USE_USERNAME

Set the following required LDAP parameters for NiFi Registry:

LDAP Parameter for NiFi Registry	Sample Value
Enable TLS/SSL for NiFi Registry	Checked
LDAP Enabled	Checked
Identity Provider: Default LDAP Provider Class	org.apache.nifi.registry.security.ldap.LdapIdentityProvider
Initial Admin Identity	admin
Identity Provider Identifier	ldap-provider
LDAP Authentication Strategy	SIMPLE
LDAP Manager DN	uid=admin,ou=people,dc=hadoop,dc=apache,dc=org
LDAP Manager Password	admin-password
LDAP URL	ldap://<ldap-hostname>:33389
LDAP User Search Base	ou=people,dc=hadoop,dc=apache,dc=org

LDAP Parameter for NiFi Registry	Sample Value
Identity Provider: Default LDAP User Search Filter	uid={0}
Identity Provider: Default LDAP Identity Strategy	USE_USERNAME
Client Authentication Required	Unchecked



**Note:** If during the initial install of NiFi and NiFi Registry, you did not set Initial Admin Identity to the correct LDAP admin user, then for each service select **Actions** **Reset File-based Authorizer Users and Policies**. This will cause a new `users.xml` and `authorizations.xml` file to be generated at start up and archives the previous `users.xml` and `authorizations.xml` files.

## Identity-Mapping Properties

Identity-mapping properties can be utilized to normalize user identities. When implemented, identities authenticated by different identity providers (certificates, LDAP, Kerberos) are treated the same internally in NiFi. As a result, duplicate users are avoided and user-specific configurations such as authorizations only need to be setup once per user.

The following examples demonstrate normalizing DN's from certificates and principals from Kerberos:

```
nifi.security.identity.mapping.pattern.dn=^CN=(.*?), OU=(.*?), O=(.*?), L=(.*?), ST=(.*?), C=(.*?)$
nifi.security.identity.mapping.value.dn=$1@$2
nifi.security.identity.mapping.transform.dn=NONE
nifi.security.identity.mapping.pattern.kerb=^(.*)/instance@(.*?)$
nifi.security.identity.mapping.value.kerb=$1@$2
nifi.security.identity.mapping.transform.kerb=NONE
```

The last segment of each property is an identifier used to associate the pattern with the replacement value. When a user makes a request to NiFi, their identity is checked to see if it matches each of those patterns in lexicographical order. For the first one that matches, the replacement specified in the `nifi.security.identity.mapping.value.xxxx` property is used. So a login with

```
CN=localhost, OU=Apache NiFi, O=Apache, L=Santa Monica, ST=CA,
C=US
```

matches the DN mapping pattern above and the DN mapping value `$1@$2` is applied. The user is normalized to `localhost@Apache NiFi`.

In addition to mapping, a transform may be applied. The supported versions are `NONE` (no transform applied), `LOWER` (identity lowercased), and `UPPER` (identity uppercased). If not specified, the default value is `NONE`.



**Note:** These mappings are also applied to the "Initial Admin Identity", "Cluster Node Identity", and any legacy users in the `authorizers.xml` file as well as users imported from LDAP.

Group names can also be mapped. The following example will accept the existing group name but will lowercase it. This may be helpful when used in conjunction with an external authorizer.

```
nifi.security.group.mapping.pattern.anygroup=^(.*)$
nifi.security.group.mapping.value.anygroup=$1
nifi.security.group.mapping.transform.anygroup=LOWER
```



**Note:** These mappings are applied to any legacy groups referenced in the `authorizers.xml` as well as groups imported from LDAP.

# Authorization

Authorization can occur via Ranger, or via NiFi and NiFi Registry's internal file-based authorizer.

## Ranger Authorization

Leverage Apache Ranger access policies to administer permissions for groups or individual users.

A Ranger access policy for flow management contains one or more access rights to NiFi or NiFi Registry resources in a cluster. You can add users and groups to a pre-defined policy or you can create a custom policy to add users and groups to.

## Understanding the Ranger Authorization Process for CFM

Selecting Ranger as a dependency during installation, indicates that Ranger must be used for NiFi and NiFi Registry authorization.

When Ranger is selected, the NiFi and NiFi Registry CSD scripts perform the following steps:

- Create a new repository/service in Ranger to store policies for the given NiFi or NiFi Registry instance. Each instance appears on the Ranger UI with a unique name in the following format: <CM cluster name>\_nifi or <CM cluster name>\_nifiregistry.

Example: myCFMcluster\_nifi

- Create policies for the following Initial Admin Identity and Initial Admin Groups:
  - For NiFi: nifi.initial.admin.identity and nifi.initial.admin.groups
  - For NiFi Registry: nifi.registry.initial.admin.identity and nifi.registry.initial.admin.groups
- Create policies for proxies specified by nifi.proxy.group or nifi.registry.proxy.group.

Each authorizers.xml file produced in NiFi and NiFi Registry when using Ranger, contains the following logical configuration:

- CompositeConfigurableUserGroupProvider
  - FileUserGroupProvider
  - CMUserGroupProvider
- RangerAuthorizer
  - Configured with CompositeConfigurableUserGroupProvider

The CMUserGroupProvider has the following purposes:

- Obtain the NiFi node identities (and Knox identity if present) from Cloudera Manager.
- Associate the NiFi node identities with a group.

The group associated with the identities is used as the proxy group that is placed in the Ranger policy for the/proxy resource.



**Note:** The CMUserGroupProvider is only aware of hostnames. The default identity map maps a configured DN to its CN value. Disabling this identity map will cause the CMUserGroupProvider to stop working.

## Before you begin

Meet the prerequisites before you assign policies to a user.

Ensure that you meet the following prerequisites:

- You created a Flow Management cluster.
- You determined the permission level for each user.



## Add user to a pre-defined Ranger access policy

When a user attempts to view or modify a NiFi or NiFi Registry resource, the system checks whether the user has privileges to perform that action. These privileges are determined by the Ranger access policies that a user is associated with.

### About this task

Determine what the user can command, control, and observe in a NiFi dataflow or in NiFi Registry and accordingly add the user or a group of users to the appropriate pre-defined Ranger access policies.

Each pre-defined Ranger access policy confers specific rights to NiFi or NiFi Registry resources.

For more information, see:

- *Pre-defined Ranger access policies for NiFi resources*
- *Pre-defined Ranger access policies for NiFi Registry resources*

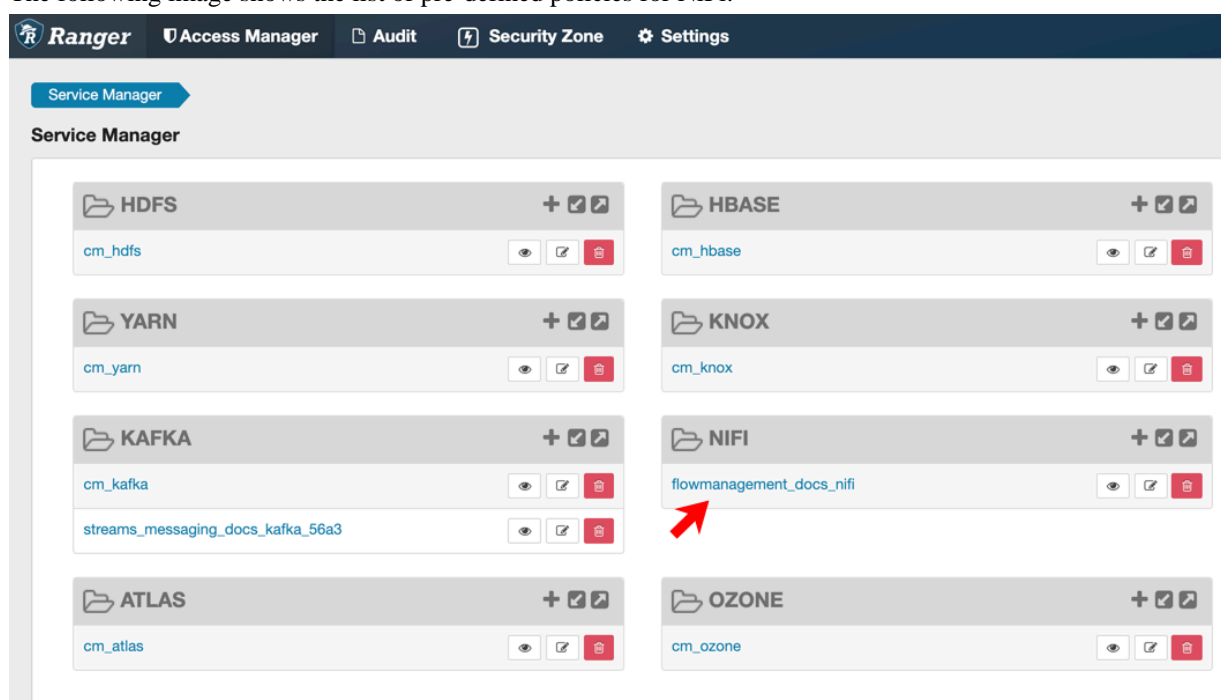
### Procedure

1. From the base cluster with Ranger, click the Ranger icon.  
The **Ranger Service Manager** page appears.

Each cluster in the environment is listed under its respective service. For example, the NiFi clusters in the environment are listed under NiFi.

2. Select a cluster from either the NiFi or NiFi Registry section.

The following image shows the list of pre-defined policies for NiFi:



The **List of Policies** page appears.

### 3. Click the ID for a policy.

The following image shows the list of pre-defined policies for NiFi:

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users	Action
52	all - nifi-resource	--	Enabled	Enabled	--	c_ranger_admins_a44480d	rangerlookup	
53	Restricted Components	--	Enabled	Enabled	--	c_nifi_admins_a44480d	--	
54	Tenants	--	Enabled	Enabled	--	c_nifi_admins_a44480d	--	
55	Controller	--	Enabled	Enabled	--	c_nifi_admins_a44480d	--	
56	Flow	--	Enabled	Enabled	--	c_nifi_admins_a44480d	--	
57	Policies	--	Enabled	Enabled	--	c_nifi_admins_a44480d	--	
58	Proxies	--	Enabled	Enabled	--	nifi	--	
66	Root Process Group	--	Enabled	Enabled	--	c_nifi_admins_a44480d	--	
67	Root Group Data	--	Enabled	Enabled	--	nifi c_nifi_admins_a44480d	--	

The **Edit Policy** page appears.

### 4. In the Allow Conditions section, add the user or the user group to the Select User field.

### 5. Click Save.

## Results

The user now has the NiFi and NiFi Registry rights according to the policies you added the user or user group to. These rights are inherited down the hierarchy unless there is a more specific policy on a component.

## Create a custom Ranger access policy

A user might need access to specific NiFi or NiFi Registry resources such as a process group or bucket. If the user cannot access the component through an inherited Ranger access policy, then you must create a custom Ranger access policy for the specific component and add the user to this policy. If all the users in a group require the same access, you can add the user group to the Ranger access policy.

### About this task

Each custom Ranger access policy provides access to a specific component.

First determine which NiFi or NiFi Registry components a user needs access to. Then create a new policy for each component and add the user or user group to the new policy.

When you create a new policy, you must specify the ID of the component that the user requires access to.



#### Note:

If a user requires permission to view or modify data for a specific component, you must create a custom data access policy and add the user and the nifi group to that policy.

The nifi group is a dynamically-managed group that exists on all Flow Management hosts and contains the identities of NiFi and Knox nodes. When you add the nifi group to the data policy for a specific component, you authorize the nodes to access data on behalf of the user.

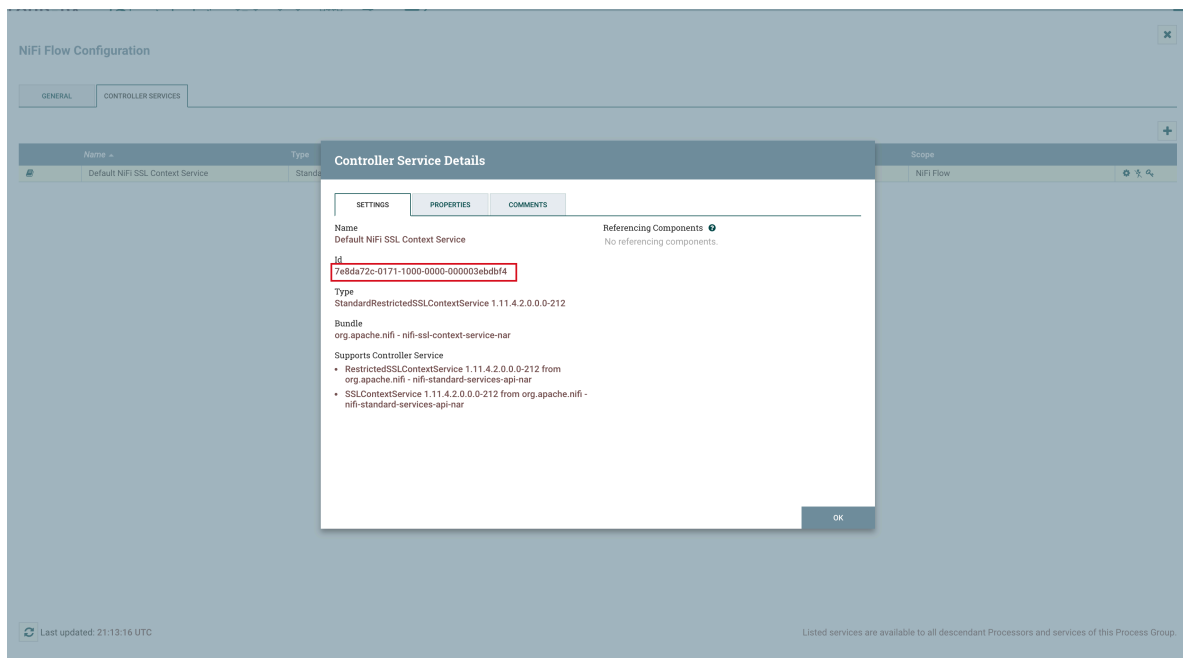
**Procedure**

1. From the NiFi canvas, copy the ID of the process group, SSL Context Service, or controller service for reporting tasks that the user needs access to.
2. To locate the ID for a process group:
  - a) Click the process group.  
The ID appears in the **Operate** pane.

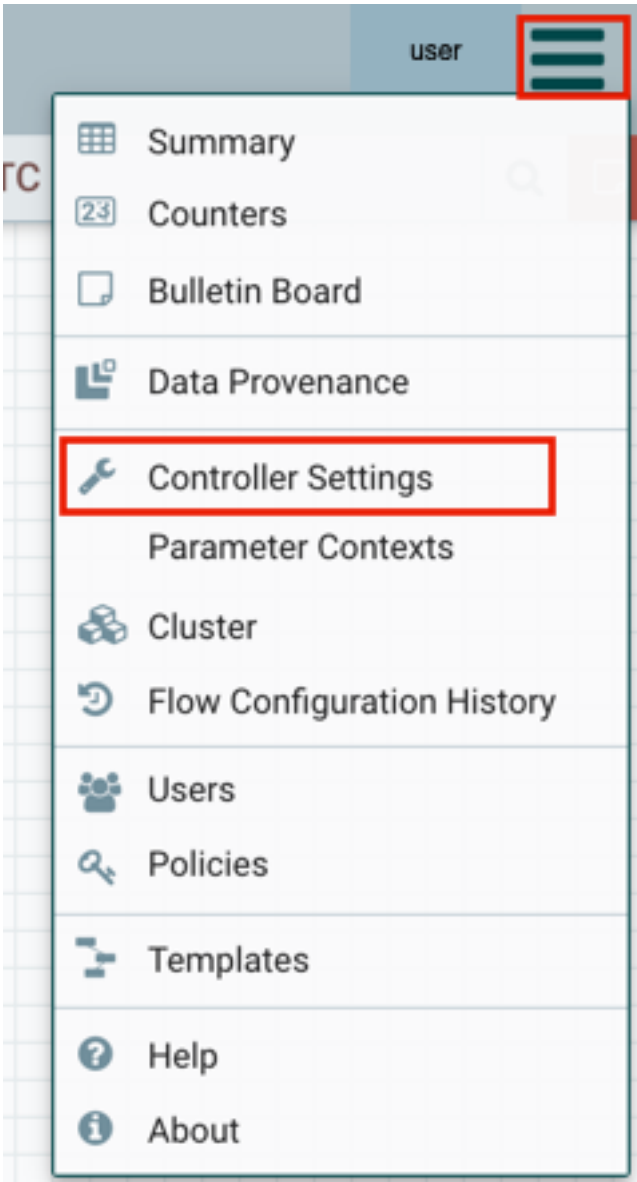


- b) Copy the ID.

3. To locate the ID of the SSL Context Service:
  - a) Click the settings icon on the process group.  
The **NiFi Flow Configuration** appears.
  - b) Click the **Controller Services** tab.
  - c) Click the **Settings** icon for the Default NiFi SSL Context Service.  
The **Controller Service Details** window appears.
  - d) From the **Settings** tab, copy the ID from the Id field.



- 4. To locate the ID of a controller service for reporting tasks:
  - a) Click the process group.
  - b) Click the menu on the top right of the UI and select Controller Settings.



The **NiFi Settings** page appears.

- c) Click the **Reporting Tasks Controller Services** tab.
- d) Click the Settings icon for the controller service.



The **Controller Service Details** page appears.

- e) From the **Settings** tab, copy the ID from the Id field.

## Controller Service Details

SETTINGS
PROPERTIES
COMMENTS

**Name**  
Default Reporting Task SSL Context Service

**Id**  
05ad168c-0171-1000-ffff-ffffe39561d8

**Type**  
StandardSSLContextService 1.11.3.2.0.0-195

**Bundle**  
org.apache.nifi - nifi-ssl-context-service-nar

**Supports Controller Service**

- SSLContextService 1.11.3.2.0.0-195 from org.apache.nifi - nifi-standard-services-api-nar

**Referencing Components** ⓘ  
No referencing components.

OK

5. Go back to the **Ranger List of Policies** page.
6. Click Add New Policy.

**Ranger**
Access Manager
Audit
Security Zone
Settings

Service Manager
docs\_flowm\_nifi Policies

**List of Policies : docs\_flowm\_nifi**

Add New Policy

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users	Action
52	all - nifi-resource	--	Enabled	Enabled	--	_c_ranger_admins_a44480d	rangerlookup	
53	Restricted Components	--	Enabled	Enabled	--	_c_nifi_admins_a44480d	--	
54	Tenants	--	Enabled	Enabled	--	_c_nifi_admins_a44480d	--	
55	Controller	--	Enabled	Enabled	--	_c_nifi_admins_a44480d	--	
56	Flow	--	Enabled	Enabled	--	_c_nifi_admins_a44480d	--	
57	Policies	--	Enabled	Enabled	--	_c_nifi_admins_a44480d	--	
58	Proxies	--	Enabled	Enabled	--	nifi	--	
66	Root Process Group	--	Enabled	Enabled	--	_c_nifi_admins_a44480d	--	
67	Root Group Data	--	Enabled	Enabled	--	nifi _c_nifi_admins_a44480d	--	

The **Create Policy** page appears.

7. Enter a unique name for the policy.
8. Optionally, enter a keyword in the Policy Label field to aid in searching for a policy.
9. Enter the resource descriptor and the resource ID in the NiFi Resource Identifier or NiFi Registry Resource Identifier field in the following format: <resource descriptor>/<resource ID>  
To determine a NiFi resource descriptor, see *Pre-defined Ranger access policies for Apache NiFi*.  
To determine a NiFi Registry resource descriptor, see *Pre-defined Ranger access policies for Apache NiFi Registry*.
10. Optionally, enter a description.

### 11. Add a user or a group.



**Note:** If a user requires permission to view or modify the data for a specific component, you must create a data policy with `/data/<component-type>/<component-UUID>` as the resource identifier. Then add the user and the nifi group to the policy to authorize the NiFi and Knox nodes to access data on behalf of the user.

### 12. Set the permission level for the user or group.

### 13. Click Add.

## Results

The user or group of users can now access the component specified in the custom policy.

## Authorization example

You can review an example to understand how you can enable a flow-management user to perform specific tasks like setting up version control for a flow, by assigning the appropriate Ranger policies.

UserA must be able to do the following tasks:

- Access the NiFi UI.
- Export a flow.
- View data queued in connections.
- View data flowing through.
- Use a NiFi SSLContextService to connect to SSL-enabled systems.
- Set up version control for a flow.

Complete the following steps to enable UserA to perform the required tasks:

### 1. Add UserA to the predefined Ranger access policy for NiFi, Flow. Set the permissions to Read.

The Flow policy gives the user the right to view the NiFi UI.

### 2. Create a Ranger access policy for NiFi with:

- Resource descriptor: `/data/process-groups/<ID of process-group>`
- Permission: Read and Write

Add UserA to this custom policy. The policy gives the user the right to export the data, view the data that is queued and flowing through the connections.

### 3. Create a Ranger access policy for NiFi with:

- Resource descriptor: `/controller-service/<ID of SSL Context Service>`
- Permission: Read

Add UserA to this custom policy. The policy gives the user the right to use the specified SSLContextService in their flows to connect to SSL-enabled systems.

### 4. Create a Ranger access policy for NiFi Registry with:

- Resource descriptor: `/buckets/<ID of bucket>`
- Permission: Read, Write, and Delete

Add UserA to this custom policy. The policy gives the user the right to set up version control for a flow.

## Predefined Ranger Access Policies for Apache NiFi

You can review the predefined Ranger policies for NiFi to determine the appropriate policy to assign to a user.

The following table lists the predefined Ranger access policies for NiFi. If you create a custom policy, refer to the Resource Descriptor column in this table to enter the value in the NiFi Resource Identifier field on the **New Policy** page.




**Important:** Do not rename the default policies as some cluster operations rely on these policy names.

**Note:**

The NiFi and Knox nodes have permission to the following Ranger policies:

- Proxies; /proxy
- Root Group Data; /data/process-groups

Ranger Policy	Description	Resource Descriptor
Controller	Allows users to view and modify the controller including Reporting Tasks, Controller Services, Parameter Contexts and Nodes in the Cluster.	/controller
Flow	Allows users to view the NiFi UI.	/flow
Policies	Allows users to view the policies for all components.	/policies
Provenance	Allows users to submit a Provenance Search and request Event Lineage.	/provenance
Proxies	Allows NiFi and Knox hosts to proxy user requests. Does not apply to users or user groups.	/proxy
Restricted Components	<p>Allows users to create/modify restricted components assuming other permissions are sufficient.</p> <p>The restricted components may indicate the specific permissions that are required.</p> <p>Permissions can be granted for specific restrictions or be granted regardless of restrictions. If permission is granted regardless of restrictions, the user can create/modify all restricted components.</p> <p>Some examples of restricted components are ExecuteScript, List/FetchHDFS, and TailFile.</p>	/restricted-components
Root Group Data	<p>Allows users and the nifi group to view and delete data from the root group and down the hierarchy unless there is a more specific policy on a component.</p> <p> <b>Note:</b> The nifi group is a dynamically managed list of Knox and NiFi node identities and has permissions to this policy.</p>	/data/process-groups
Root Group Provenance Data	Allows users to view provenance data.	/provenance-data/process-groups/
Root Process Group	<p>Allows users to view and modify the root process group including adding/removing processors to the canvas.</p> <p>This policy is inherited down the hierarchy unless there is a more specific policy on a component.</p>	/process-groups
Tenants	Allows users to view and modify user accounts and user groups.	/tenants

## Predefined Ranger Access Policies for Apache NiFi Registry

You can review the predefined Ranger policies for NiFi Registry to determine the appropriate policy to assign to a user.

The following table lists the pre-defined Ranger access policies for NiFi Registry. If you create a custom policy, refer to the Resource Descriptor column in this table to enter the value in the NiFi Registry Resource Identifier field on the **New Policy** page.





**Important:** Do not rename the default policies as some cluster operations rely on these policy names.



**Note:** The NiFi Registry and Knox nodes have permission to the Proxies (/proxy) Ranger policy.

Ranger Policy	Description	Resource Descriptor
Actuator	Allows users to access the Spring Boot Actuator end-points.	/actuator
Buckets	Allows users to view and modify all buckets.	/buckets
Policies	Allows users to view the policies for all components.	/policies
Proxies	Allows NiFi Registry and Knox hosts to proxy user requests. Does not apply to users or user groups.	/proxy
Swagger	Allows users to access the self-hosted Swagger UI.	/swagger
Tenants	Allows users to view and modify user accounts and user groups.	/tenants

## File-Based Authorization

When Ranger is not selected as a dependency during installation, NiFi or NiFi Registry's internal file-based authorizer will be used for authorization.

When Ranger is not selected, the NiFi and NiFi Registry CSD scripts will perform the following steps:

- By default, during start-up, NiFi and NiFi Registry will create the following files in /var/lib/nifi and /var/lib/nifi-registry:
  - users.xml
  - authorizations.xml

These files will include the users and policies for the Initial Admin Identity, Initial Admin Groups, and proxy group.

- Create policies for the following Initial Admin Identity and Initial Admin Groups:
  - For NiFi: nifi.initial.admin.identity and nifi.initial.admin.groups
  - For NiFi Registry: nifi.registry.initial.admin.identity and nifi.registry.initial.admin.groups
- Create policies for proxies specified by nifi.proxy.group or nifi.registry.proxy.group.

Each authorizers.xml file produced in NiFi and NiFi Registry when using file-based authorization contains the following logical configuration:

- CompositeConfigurableUserGroupProvider
  - FileUserGroupProvider
  - CMUserGroupProvider
- FileAccessPolicyProvider
  - Configured with the CompositeConfigurableUserGroupProvider
- StandardManagedAuthorizer
  - Configured with FileAccessPolicyProvider

## LDAP Integration

After Ranger or file-based authorizations are implemented, the authorizations can be configured to integrate with LDAP.

However, after installation, the authorization configuration can be re-configured to setup an LDAPUserGroupProvider.

When you setup an LDAPUserGroupProvider, the FileUserGroupProvider is replaced with the LDAPUserGroupProvider.



**Note:** The following is an important distinction between the FileUserGroupProvider and the LDAPUserGroupProvider:

- When using the FileUserGroupProvider, the composite provider is the CompositeConfigurableUserGroupProvider.
- When using the LDAPUserGroupProvider, the provider is the non-configurable CompositeUserGroupProvider.

## LDAP and Ranger Policies

Set up the LDAP and Ranger integration in NiFi and NiFi Registry.

### About this task

Each authorizers.xml file produced in NiFi and NiFi Registry when using LDAP with Ranger policies, contain the following logical configuration:

- CompositeUserGroupProvider
  - LdapUserGroupProvider
  - CMUserGroupProvider
- RangerAuthorizer
  - Configured with CompositeUserGroupProvider



**Note:** Confer with your Active Directory/LDAP team to get the values you would need to set the LDAP User Group Provider properties. For a list of the properties, see *LDAP User Group Provider Properties*.

### Procedure

1. From Cloudera Manager, select the NiFi/NiFi Registry Service, and click the **Configuration** tab.
2. Uncheck Authorizers: Enable File User Group Provider to disable the file-user-group-provider.
3. Uncheck Authorizers: Enable Composite Configurable User Group Provider to disable the composite-configurable-user-group-provider.
4. Check Authorizers: Enable Composite User Group Provider to enable composite-user-group-provider.
  - a) Enter ldap-user-group-provider for Authorizers: Composite User Group Provider Property - User Group Provider 1.
  - b) Enter cm-user-group-provider for Authorizers: Composite User Group Provider Property - User Group Provider 2.
5. Check LDAP Enabled to enable ldap-user-group-provider.
6. In the Search field, enter ldap-user-group-provider to see the list of the LDAP User Group Provider properties. For a list of the properties, see *LDAP User Group Provider Properties*.
7. Update the LDAP User Group Provider properties.
8. Update Authorizers: Ranger Authorizer Property - User Group Provider to use the composite-user-group-provider instead of the configurable one.
9. Save the changes.

10. Locate the Login Identity Provider ID and verify that it is set to your authentication provider. Either:

- kerberos-provider

or

- ldap-provider

### Related Information

[LDAP User Group Provider Properties](#)

## LDAP and File-Based Policies

Set up the LDAP and file-based integration in NiFi and NiFi Registry.

### About this task

Each `authorizers.xml` file produced in NiFi and NiFi Registry when using LDAP with file-based policies, contain the following logical configuration:

- CompositeUserGroupProvider
  - LdapUserGroupProvider
  - CMUserGroupProvider
- FileAccessPolicyProvider
  - Configured with CompositeUserGroupProvider
- StandardManagedAuthorizer
  - Configured with FileAccessPolicyProvider



**Note:** Confer with your Active Directory/LDAP team to get the values you would need to set the LDAP User Group Provider properties. For a list of the properties, see *LDAP User Group Provider Properties*.

### Procedure

1. From Cloudera Manager, select the NiFi/NiFi Registry Service, and click the **Configuration** tab.
2. Uncheck **Authorizers: Enable File User Group Provider** to disable the file-user-group-provider.
3. Uncheck **Authorizers: Enable Composite Configurable User Group Provider** to disable the composite-configurable-user-group-provider.
4. Check **Authorizers: Enable Composite User Group Provider** to enable composite-user-group-provider.
  - a) Enter `ldap-user-group-provider` for **Authorizers: Composite User Group Provider Property - User Group Provider 1**.
  - b) Enter `cm-user-group-provider` for **Authorizers: Composite User Group Provider Property - User Group Provider 2**.
5. Check **LDAP Enabled** to enable ldap-user-group-provider.
6. In the **Search** field, enter `ldap-user-group-provider` to see the list of the LDAP User Group Provider properties. For a list of the properties, see *LDAP User Group Provider Properties*.
7. Update the LDAP User Group Provider properties.
8. Update **Authorizers: Default File Access Policy Property - User Group Provider** to use the composite-user-group-provider instead of the configurable one.
9. Save the changes.
10. Locate the Login Identity Provider ID and verify that it is set to your authentication provider. Either:
  - kerberos-provider
 or
  - ldap-provider

## Related Information

### LDAP User Group Provider Properties

## LDAP User Group Provider Properties

After you enable authorization through Ranger or file-based policies, set the LDAP User Group Provider properties to enable NiFi/NiFi Registry to sync users and user groups and determine the association between them.

Set the following LDAP User Group Provider properties (ldap-user-group-provider) in the Cloudera Manager **Configuration** tab.

LDAP User Group Provider Properties	Description
Authorizers: LDAP Authentication Strategy	How the connection to the LDAP server is authenticated. Possible values are ANONYMOUS, SIMPLE, LDAPS, or START_TLS.
Authorizers: LDAP Manager DN	The DN of the manager that is used to bind to the LDAP server to search for users.
Authorizers: LDAP Manager Password	The password of the manager that is used to bind to the LDAP server to search for users.
Authorizers: LDAP TLS - Keystore	Path to the Keystore that is used when connecting to LDAP using LDAPS or START_TLS.
Authorizers: LDAP TLS - Keystore Password	Password for the Keystore that is used when connecting to LDAP using LDAPS or START_TLS.
Authorizers: LDAP TLS - Keystore Type	Type of the Keystore that is used when connecting to LDAP using LDAPS or START_TLS (i.e. JKS or PKCS12).
Authorizers: LDAP TLS - Truststore	Path to the Truststore that is used when connecting to LDAP using LDAPS or START_TLS.
Authorizers: LDAP TLS - Truststore Password	Password for the Truststore that is used when connecting to LDAP using LDAPS or START_TLS.
Authorizers: LDAP TLS - Truststore Type	Type of the Truststore that is used when connecting to LDAP using LDAPS or START_TLS (i.e. JKS or PKCS12).
Authorizers: LDAP TLS - Client Auth	Client authentication policy when connecting to LDAP using LDAPS or START_TLS. Possible values are REQUIRED, WANT, NONE.
Authorizers: LDAP TLS - Protocol	Protocol to use when connecting to LDAP using LDAPS or START_TLS. (i.e. TLS, TLSv1.1, TLSv1.2, etc).
Authorizers: LDAP TLS - Shutdown Gracefully	Specifies whether the TLS should be shut down gracefully before the target context is closed. Defaults to false.
Authorizers: LDAP Referral Strategy	Strategy for handling referrals. Possible values are FOLLOW, IGNORE, THROW.
Authorizers: LDAP Connect Timeout	Duration of connect timeout. (i.e. 10 secs).
Authorizers: LDAP Read Timeout	Duration of read timeout. (i.e. 10 secs).
Authorizers: LDAP Url	Space-separated list of URLs of the LDAP servers (i.e. ldap://<host name>:<port>).
Authorizers: LDAP Page Size	Sets the page size when retrieving users and groups. If not specified, no paging is performed.
Authorizers: LDAP Group Membership - Enforce Case Sensitivity	Sets whether group membership decisions are case sensitive. When a user or group is inferred (by not specifying user or group search base or user identity attribute or group name attribute) case sensitivity is enforced since the value to use for the user identity or group name would be ambiguous. Defaults to false.
Authorizers: LDAP Sync Interval	Duration of time between syncing users and groups. (i.e. 30 mins). Minimum allowable value is 10 secs.
Authorizers: LDAP User Search Base	Base DN for searching for users (i.e. ou=users,o=nifi). Required to search users.

LDAP User Group Provider Properties	Description
Authorizers: LDAP User Object Class	Object class for identifying users (i.e. person). Required if searching users.
Authorizers: LDAP User Search Scope	Search scope for searching users (ONE_LEVEL, OBJECT, or SUBTREE). Required if searching users.
Authorizers: LDAP User Search Filter	Filter for searching for users against the User Search Base (i.e. (memberof=cn=team1,ou=groups,o=nifi)). Optional.
Authorizers: LDAP User Identity Attribute	Attribute to use to extract user identity (i.e. cn). Optional. If not set, the entire DN is used.
Authorizers: LDAP User Group Name Attribute	Attribute to use to define group membership (i.e. memberof). Optional. If not set group membership will not be calculated through the users. Will rely on group membership being defined through Group Member Attribute if set. The value of this property is the name of the attribute in the user ldap entry that associates them with a group. The value of that user attribute could be a dn or group name for instance. What value is expected is configured in the User Group Name Attribute - Referenced Group Attribute.
Authorizers: LDAP User Group Name Attribute - Referenced Group Attribute	If blank, the value of the attribute defined in User Group Name Attribute is expected to be the full dn of the group. If not blank, this property will define the attribute of the group ldap entry that the value of the attribute defined in User Group Name Attribute is referencing (i.e. name). Use of this property requires that Group Search Base is also configured.
Authorizers: LDAP Group Search Base	Base DN for searching for groups (i.e. ou=groups,o=nifi). Required to search groups.
Authorizers: LDAP Group Object Class	Object class for identifying groups (i.e. groupOfNames). Required if searching groups.
Authorizers: LDAP Group Search Scope	Search scope for searching groups (ONE_LEVEL, OBJECT, or SUBTREE). Required if searching groups.
Authorizers: LDAP Group Search Filter	Filter for searching for groups against the Group Search Base. Optional.
Authorizers: LDAP Group Name Attribute	Attribute to use to extract group name (i.e. cn). Optional. If not set, the entire DN is used.
Authorizers: LDAP Group Member Attribute	Attribute to use to define group membership (i.e. member). Optional. If not set group membership will not be calculated through the groups. Will rely on group membership being defined through User Group Name Attribute if set. The value of this property is the name of the attribute in the group ldap entry that associates them with a user. The value of that group attribute could be a dn or memberId for instance. What value is expected is configured in the Group Member Attribute - Referenced User Attribute. (i.e. member: cn=User 1,ou=users,o=nifi vs. memberId: user1)
Authorizers: LDAP Group Member Attribute - Referenced User Attribute	If blank, the value of the attribute defined in Group Member Attribute is expected to be the full dn of the user. If not blank, this property will define the attribute of the user ldap entry that the value of the attribute defined in Group Member Attribute is referencing (i.e. uid). Use of this property requires that User Search Base is also configured. (i.e. member: cn=User 1,ou=users,o=nifi vs. memberId: user1)

## Migrate file-based authorization to Ranger

Both NiFi and NiFi Registry services have the option to convert existing file-based provider policies to Ranger provider policies.

## Migrate NiFi File-Based Authorization to Ranger

You can convert existing file-based provider NiFi policies to Ranger provider policies.

### Before you begin

The following steps assume that the Ranger service is installed in the CDP-DC cluster.

### Procedure

1. Create any users and groups from the NiFi users.xml that do not already exist in Ranger.
2. Select Ranger as a dependency from NiFi configuration.
3. Restart NiFi.
4. Select Migrate File-based Authorizations to Ranger from the Actions drop-down. Confirm the action.
5. After a successful migration, verify that the policies are available in the NiFi Ranger service.

## Migrate NiFi Registry File-Based Authorization to Ranger

You can convert existing file-based provider NiFi Registry policies to Ranger provider policies.

### Before you begin

The following steps assume that the Ranger service is installed in the CDP-DC cluster.

### Procedure

1. Create any users and groups from the NiFi Registry users.xml that do not already exist in Ranger.
2. Select Ranger as a dependency from NiFi Registry configuration.
3. Restart NiFi Registry.
4. Select Migrate File-based Authorizations to Ranger from the Actions drop-down. Confirm the action.
5. After a successful migration, verify that the policies are available in the NiFi Registry Ranger service.

## TLS Configuration

When you configure authentication and authorization for your flow management cluster, CFM sends sensitive information over the network to cluster hosts, such as Kerberos keytabs and configuration files that contain passwords. To secure this transfer, you must configure Transport Layer Security (TLS) encryption.

TLS is an industry standard set of cryptographic protocols for securing communications over a network.

Configuring TLS involves creating a private key and a public key for use by server and client processes to negotiate an encrypted connection at runtime. In addition, TLS can use certificates to verify the trustworthiness of keys presented during the negotiation to prevent spoofing and mitigate other potential security issues.

In CFM, you can configure TLS in one of the following ways:

### Enable Auto-TLS

Auto-TLS greatly simplifies the process of enabling and managing TLS encryption on your cluster.

Auto-TLS automates the creation of an internal certificate authority (CA) and deployment of certificates across all cluster hosts. It can also automate the distribution of existing certificates, such as those signed by a public CA.

Adding new cluster hosts or services to a cluster that is Auto-TLS enabled, automatically creates and deploys the required certificates.

In CDP, Auto-TLS is enabled by default.



**Note:** Wildcard certificates are not supported. For example, if two nodes, node1.nifi.apache.org and node2.nifi.apache.org, are assigned the same certificate with a CN or SAN entry of \*.nifi.apache.org, the certificates will not be supported.

Ensure that you do not generate wildcard certificates for the NiFi nodes.

For more information about Auto-TLS, see *Configuring TLS Encryption for Cloudera Manager Using Auto-TLS*.

### Related Information

[Configuring TLS Encryption for Cloudera Manager Using Auto-TLS](#)

## Manually configure TLS

If you use your own enterprise-generated certificates, you would need to manually configure TLS.

### TLS certificate requirements and recommendations

If you use your own enterprise-generated certificates, you would need to manually configure TLS.

Before you manually configure TLS, ensure that the certificate that you use meets the following requirements.

#### Certificate Requirements

Verify the following minimum requirements:

- The KeyStore must contain only one PrivateKeyEntry. Using multiple private keys in one KeyStore is not supported.
- The KeyStore password and key/certificate password must be the same or no password should be set on the certificate.
- The unique KeyStores used on each NiFi cluster node must use the same KeyStore password and key/certificate password. Ambari and Cloudera Manager do not support defining unique passwords per NiFi host.
- The X509v3 ExtendedKeyUsages section of the certificate must have the following attributes:
  - clientAuth - This attribute is for TLS web client authentication.
  - serverAuth - This attribute is for TLS web server authentication.
- The signature algorithm used for the certificate must be sha256WithRSAEncryption (SHA-256).
- The certificates must not use wildcards. Each cluster node must have its own certificate. If NiFi or NiFi Registry is behind Knox, do not use wildcard certificates for Knox.
- Subject Alternate Names (SANs) are mandatory and should at least include the FQDN of the host.
- Additional names for the certificate/host can be added to the certificate as SANs.
  - Add the FQDN used for the CN as a DNS SAN entry.
  - If you are planning to use a load balancer for the NiFi service, include the FQDN for the load balancer as a DNS SAN entry.
- The X509v3 KeyUsage section of the certificate must include the following attributes:
  - DigitalSignature
  - Key\_Encipherment

#### Cloudera Recommendations

Cloudera recommends the following security protocols:

- Use certificates that are signed by a CA. Do not issue self-signed certificates.
- Generate a unique certificate per host.

## Configure TLS encryption manually for NiFi and NiFi Registry

If you do not want to enable Auto-TLS because for example, you need to use your own enterprise-generated certificates, you can manually enable TLS for NiFi and NiFi Registry.

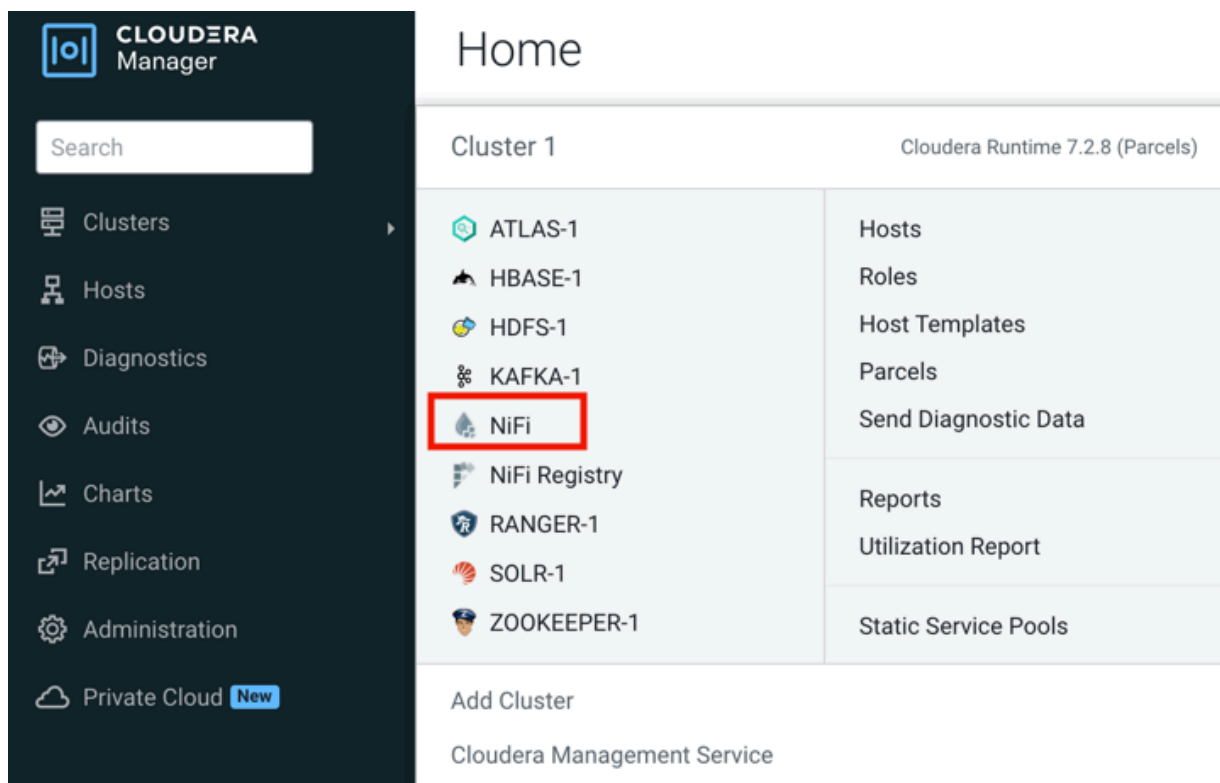
### Before you begin

Ensure you have set up TLS for Cloudera Manager:

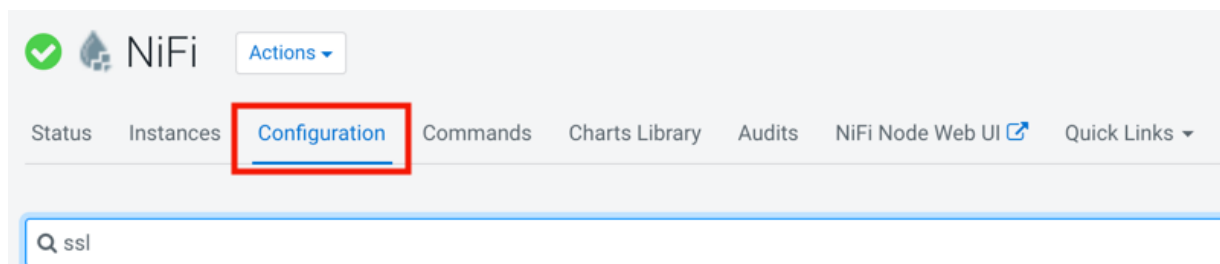
1. Review the requirements and recommendations for the certificates. See *TLS Certificate Requirements and Recommendations*.
2. Generate the TLS certificates and configure Cloudera Manager. See *Manually Configuring TLS Encryption for Cloudera Manager*.

### Procedure

1. From the Cloudera Manager UI, click Cluster NiFi .



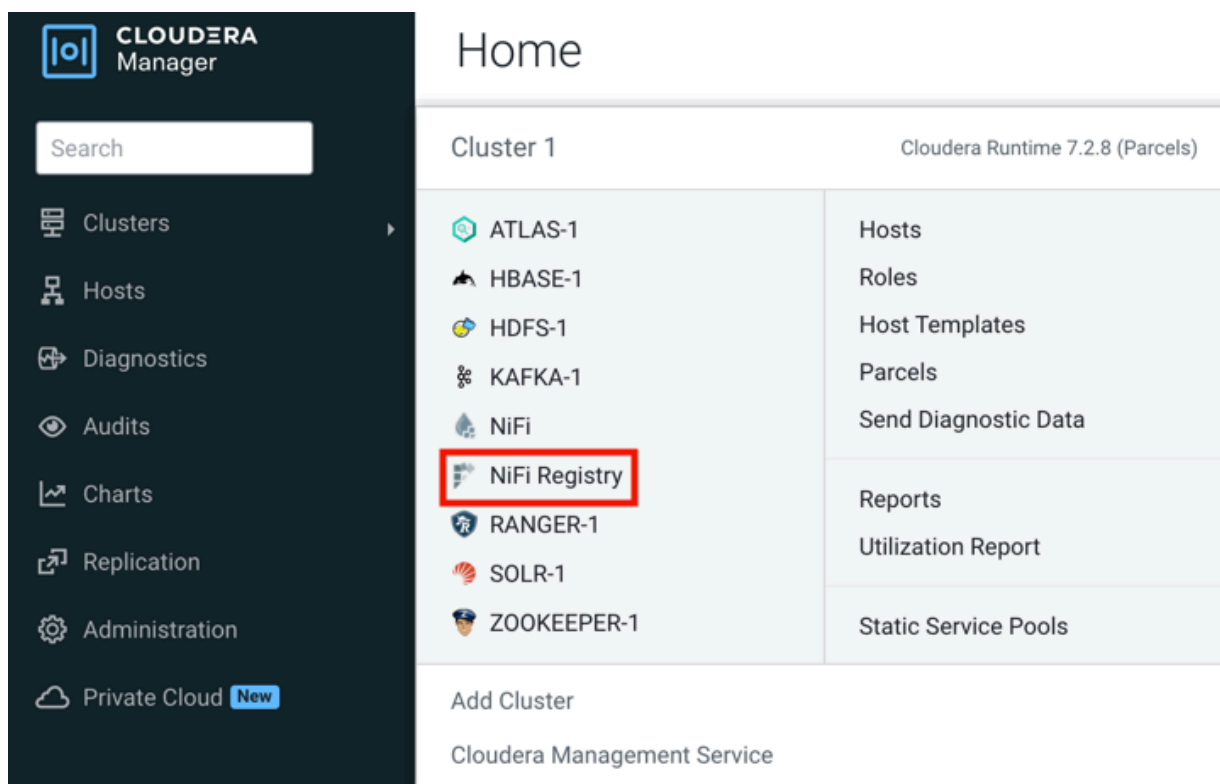
2. Click the **Configuration** tab.



3. Enter ssl in the Search field.  
The Security properties for NiFi appear.
4. Edit the Security properties.
5. Click Save Changes.



6. Restart the NiFi service.
7. Click **Cluster NiFi Registry** and repeat these steps to configure the Security properties for NiFi Registry.



### Related Information

[TLS certificate requirements and recommendations](#)

[Manually Configuring TLS Encryption for Cloudera Manager](#)

## NiFi TLS Properties

To enable and configure TLS manually for NiFi, edit the security properties according to the cluster configuration.

The following table lists the Security properties for NiFi:

Property	Description
NiFi Node TLS/SSL Server JKS Keystore File Location nifi.security.keystore	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when NiFi Node is acting as a TLS/SSL server. The keystore must be in JKS format.
NiFi Node TLS/SSL Server JKS Keystore File Password nifi.security.keystorePasswd	The password for the NiFi Node JKS keystore file.
NiFi Node TLS/SSL Server JKS Keystore Key Password nifi.security.keyPasswd	The password that protects the private key contained in the JKS keystore used when NiFi Node is acting as a TLS/SSL server.
NiFi Node TLS/SSL Client Trust Store File nifi.security.truststore	The location on disk of the trust store, in .jks format, used to confirm the authenticity of TLS/SSL servers that NiFi Node might connect to. This is used when NiFi Node is the client in a TLS/SSL connection. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
NiFi Node TLS/SSL Client Trust Store Password nifi.security.truststorePasswd	The password for the NiFi Node TLS/SSL Certificate Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.

Property	Description
xasecure.policymgr.clientssl.keystore	Path to keystore to use in policy manager.
xasecure.policymgr.clientssl.keystore.credential.file	Path to keystore credential file to use in policy manager.
xasecure.policymgr.clientssl.truststore	Path to truststore to use in policy manager.
xasecure.policymgr.clientssl.truststore.credential.file	Path to truststore credential file to use in policy manager.
Login Identity Provider: Default LDAP TLS - Keystore xml.loginIdentityProviders.provider ldap-provider.property.TLS - Keystore	Default LDAP TLS - Keystore
Login Identity Provider: Default LDAP TLS - Keystore Password xml.loginIdentityProviders.provider ldap-provider.property.TLS - Keystore Password	Default LDAP TLS - Keystore Password
Login Identity Provider: Default LDAP TLS - Keystore Type xml.loginIdentityProviders.provider ldap-provider.property.TLS - Keystore Type	Default LDAP TLS - Keystore Type
Login Identity Provider: Default LDAP TLS - Truststore xml.loginIdentityProviders.provider ldap-provider.property.TLS - Truststore	Default LDAP TLS - Truststore
Login Identity Provider: Default LDAP TLS - Truststore Password xml.loginIdentityProviders.provider ldap-provider.property.TLS - Truststore Password	Default LDAP TLS - Truststore Password
Login Identity Provider: Default LDAP TLS - Truststore Type xml.loginIdentityProviders.provider ldap-provider.property.TLS - Truststore Type	Default LDAP TLS - Truststore Type

## NiFi Registry TLS Properties

To enable and configure TLS manually for NiFi Registry, edit the security properties according to the cluster configuration.

The following table lists the Security properties for NiFi Registry:

Property	Description
NiFi Registry TLS/SSL Server JKS Keystore File Location nifi.registry.security.keystore	The path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when NiFi Registry is acting as a TLS/SSL server. The keystore must be in JKS format.
NiFi Registry TLS/SSL Server JKS Keystore File Password nifi.registry.security.keystorePasswd	The password for the NiFi Registry JKS keystore file.
NiFi Registry TLS/SSL Server JKS Keystore Key Password nifi.registry.security.keyPasswd	The password that protects the private key contained in the JKS keystore used when NiFi Registry is acting as a TLS/SSL server.
NiFi Registry TLS/SSL Client Trust Store File nifi.registry.security.truststore	The location on disk of the trust store, in .jks format, used to confirm the authenticity of TLS/SSL servers that NiFi Registry might connect to. This is used when NiFi Registry is the client in a TLS/SSL connection. This trust store must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
NiFi Registry TLS/SSL Client Trust Store Password nifi.registry.security.truststorePasswd	The password for the NiFi Registry TLS/SSL Certificate Trust Store File. This password is not required to access the trust store; this field can be left blank. This password provides optional integrity checking of the file. The contents of trust stores are certificates, and certificates are public information.
xasecure.policymgr.clientssl.keystore	Path to keystore to use in policy manager.

Property	Description
xasecure.policymgr.clientssl.keystore.credential.file	Path to keystore credential file to use in policy manager.
xasecure.policymgr.clientssl.truststore	Path to truststore to use in policy manager.
xasecure.policymgr.clientssl.truststore.credential.file	Path to truststore credential file to use in policy manager.

## Integrate NiFi and Atlas

You can integrate NiFi with Apache Atlas to take advantage of robust dataset and application lineage support.

### Manually Integrate with Atlas when Auto-TLS is not Enabled

If CFM or the cluster does not have Auto-TLS enabled and you want to Atlas, then you must manually integrate with Atlas by creating the `ReportLineageToAtlas` reporting task.

#### About this task

Perform this task if:

- CFM does not have TLS enabled; AND
- The cluster does not have auto-TLS enabled; AND
- You do not want to enable auto-TLS; AND
- You want Atlas as part of CFM on your deployment.

#### Procedure

1. From the Global Menu located in NiFi's upper right corner, select Controller Services and click the Reporting Tasks tab.
2. Click the Add (+) icon to launch the Add Reporting Task dialog.
3. Select `ReportLineageToAtlas` and click Add.
4. Click the Edit icon to launch the Configure Reporting Task dialog. The following properties are required:
  - Atlas URLs – a comma-separated list of Atlas Server URLs. Once you have started reporting, you cannot modify an existing Reporting Task to add a new Atlas Server. When you need to add a new Atlas Server, you must create a new reporting task.
  - Atlas Authentication Method – Specifies how to authenticate the Reporting Task to the Atlas Server. Basic authentication is the default.
  - NiFi URL for Atlas – Specifies the NiFi cluster URL
  - NiFi Lineage Strategy – Specifies the level of granularity for your NiFi dataflow reporting to Atlas. Once you have started reporting, you should not switch between simple and complete lineage reporting strategies.
  - Provenance Record Start Position – Specifies where in the Provenance Events stream the Reporting Task should start.
  - Provenance Record Batch Size – Specifies how many records you want to send in a single batch
  - Create Atlas Configuration File – If enabled, the `atlas-application-properties` file and the Atlas Configuration Directory are automatically created when the Reporting Task starts.
  - Kafka Security Protocol – Specifies the protocol used to communicate with Kafka brokers to send Atlas hook notification messages. This value should match Kafka's `security.protocol` property value.

### Manually Integrate with Atlas when Auto-TLS is Enabled

You must perform some manual steps to integrate with Atlas when auto-TLS is enabled on your cluster.

### About this task

You must perform these steps if:

- You want CFM to integrate with Atlas; AND
- The cluster has auto-TLS enabled

### Procedure

1. Select the Atlas integration checkbox.
2. Restart NiFi.
3. Click Create required NiFi object in the Cloudera Manager Actions menu.

## Integrate NiFi and NiFi Registry with Knox

Integrate NiFi and NiFi Registry with Knox to securely access NiFi and NiFi Registry nodes.

Apache Knox Gateway (Knox) provides the following benefits:

- Centralized access to all services in the cluster.
- Authentication with single sign-on.
- Service-level authorization to the cluster.
- Does not expose the service endpoints such as URLs, ports, IP addresses.

When you integrate NiFi and NiFi Registry with Knox, you can use the Knox URL as a single entry point to securely access all NiFi nodes and switch nodes if one fails.

For information more information on Knox, see *Apache Knox Overview*.

For information on how to select Knox during the NiFi and NiFi Registry installation, see *CFM Deployment*.

### Related Information

[Apache Knox Overview](#)

[CFM Deployment](#)

## Enhance or Overwrite Properties in Cloudera Manager

You can customize NiFi and NiFi Registry beyond what the customization page in Cloudera Manager allows. To customize, use the dot notation to represent the actual schema for a given property file.

### About this task

The following steps show how to enhance or overwrite xml based properties in Cloudera Manager using dot notation.

### Procedure

Use the following structure:

```
xml.<properties-type>.<entity>.<identifier>.class  
xml.<properties-type>.<entity>.<identifier>.property.<property-value>
```

Where:

- <properties-type> for NiFi can be authorizers and loginIdentityProviders
- <properties-type> for NiFi Registry can be authorizers and identityProviders.

The following property key/value example creates a user group provider entry into the authorizers file for NiFi:

```
Name: xml.authorizers.userGroupProvider.file-user-group-provider.class
Value: org.apache.nifi.authorization.FileUserGroupProvider

Name: xml.authorizers.userGroupProvider.file-user-group-provider.property
.Initial User Identity 2
Value: CN=localhost, OU=NIFI
```

This translates to the following entry in the generated authorizers.xml file:

```
<authorizers>
....
  <userGroupProvider>
    <identifier>file-user-group-provider</identifier>
    <class>org.apache.nifi.authorization.FileUserGroupProvider</class>
    <property name="Initial User Identity 2">CN=localhost, OU=NIFI</prop
erty>
  </userGroupProvider>
...
...
</authorizers>
```

Properties names that have spaces are supported and do not need to be escaped.

**Example**

For an example, see *Pairing LDAP with a Composite Group Provider*.

**Related Information**

[Pairing LDAP with a Composite Group Provider](#)

# Pairing LDAP with a Composite Group Provider

If you need to combine multiple user/group provider mechanisms into a composite provider, you can do so using the Cloudera Manager safety valves for the authorizers.xml file.

This example shows how File based users/group provider can be paired with an LDAP user group provider using a CompositeConfigurableUserGroupProvider.

Property Name	Description	Property Value (Default)
xml.authorizers.userGroupProvider.composite-user-group-provider.class		org.apache.nifi.authorization.CompositeConfigurableUserGroupP
xml.authorizers.userGroupProvider.composite-user-group-provider.property.Configurable User Group Provider		
xml.authorizers.userGroupProvider.composite-user-group-provider.property.User Group Provider 1		

<b>Name</b>	xml.authorizers.userGroupProvider.composite-u
<b>Value</b>	org.apache.nifi.authorization.CompositeConfigu
<b>Description</b>	Description
	<input type="checkbox"/> Final
<b>Name</b>	xml.authorizers.userGroupProvider.composite-u
<b>Value</b>	file-user-group-provider
<b>Description</b>	Description
	<input type="checkbox"/> Final
<b>Name</b>	xml.authorizers.userGroupProvider.composite-u
<b>Value</b>	ldap-user-group-provider
<b>Description</b>	Description
	<input type="checkbox"/> Final

## Default Ports for NiFi and NiFi Registry

Reference for the NiFi and NiFi Registry default ports.

## NiFi

The following table lists the default ports used by NiFi and the corresponding property in the `nifi.properties` file. You can change these values as required.



**Note:**

- The default values are set by Cloudera Manager.
- If you install NiFi-only binaries not managed by Cloudera Manager, the defaults will be different and can be found in the `nifi.properties` file.

Function	Property	Default Value set by Cloudera Manager
HTTP Port	<code>nifi.web.http.port</code>	8080
HTTPS Port	<code>nifi.web.https.port</code>	8443
Remote Input Socket Port	<code>nifi.remote.input.socket.port</code>	none
Cluster Node Protocol Port	<code>nifi.cluster.node.protocol.port</code>	9088
Cluster Node Load Balancing Port	<code>nifi.cluster.node.load.balance.port</code>	6342
Web HTTP Forwarding Port	<code>nifi.web.http.port.forwarding</code>	none

## NiFi Registry

The following table lists the default ports used by NiFi Registry and the corresponding property in the `nifi-registry.properties` file. You can change these values as required.



**Note:**

If you install NiFi-only binaries not managed by Cloudera Manager, then:

- When enabling HTTPS, unset the `nifi.registry.web.http.port` property.
- The default values will be different and can be found in the `nifi-registry.properties` file.

Function	Property	Default Value set by Cloudera Manager
HTTP Port	<code>nifi.registry.web.http.port</code>	18080
HTTPS Port	<code>nifi.registry.web.https.port</code>	18433