CFM 2.0.1

CFM Security

Date published: 2020-06-30
Date modified: 2020-10-02

https://docs.cloudera.com/

Legal Notice

Contents

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.

Authentication

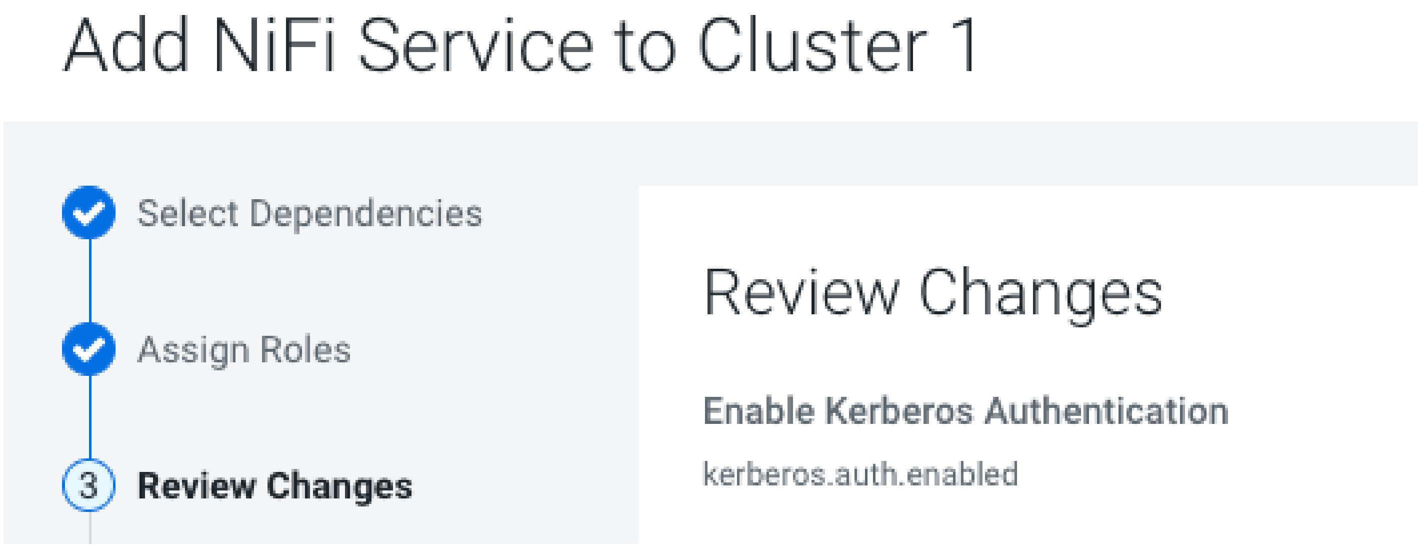
TLS/SSL must be enabled before NiFi can support any form of user authentication. The primary mechanisms of authenticating to NiFi and NiFi Registry in a CDP Private Cloud Base cluster are Kerberos and LDAP.

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.



Enable Kerberos Authentication
kerberos.auth.enabled

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
Authorizers: LDAP User Search Filter	(uid=*)
Authorizers: LDAP User Identity Attribute	uid

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
Authorizers: LDAP User Search Filter	(uid=*)
Authorizers: LDAP User Identity Attribute	uid
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 DNs 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 loca lhost@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 identies 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:

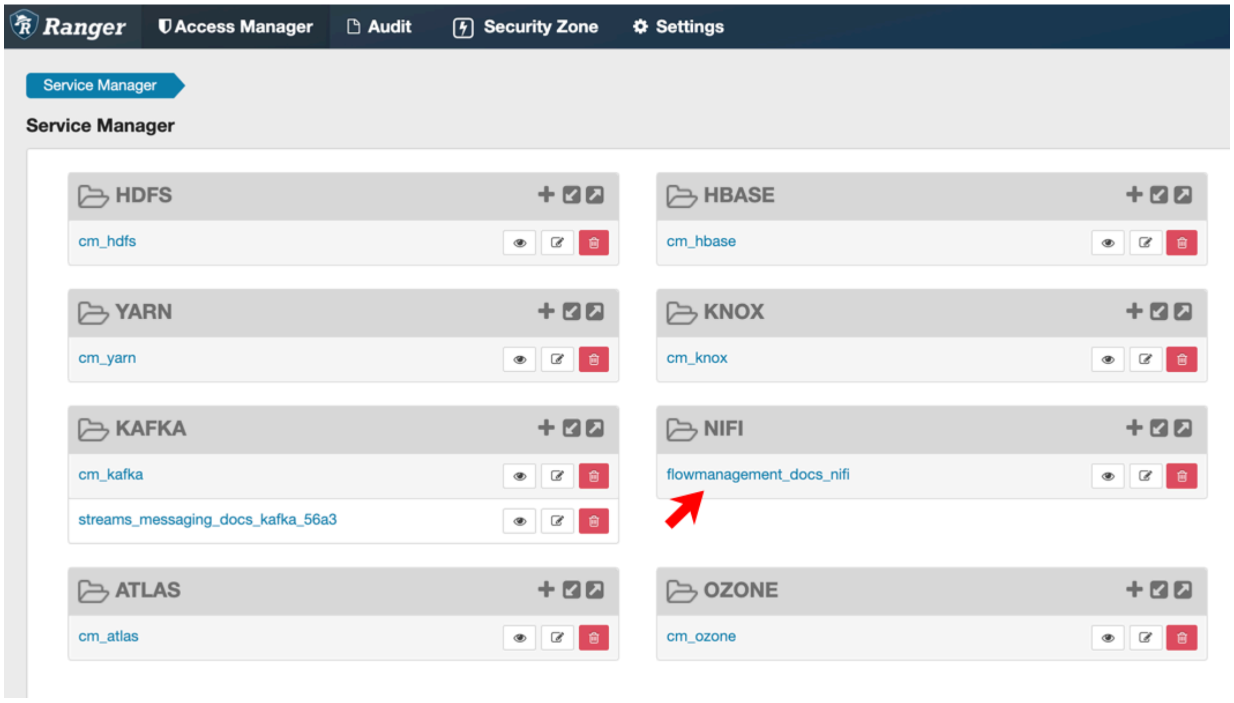
- *Pre-defined Ranger access policies for NiFi resources*
- *Pre-defined Ranger access policies for NiFi Registry resources*

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.

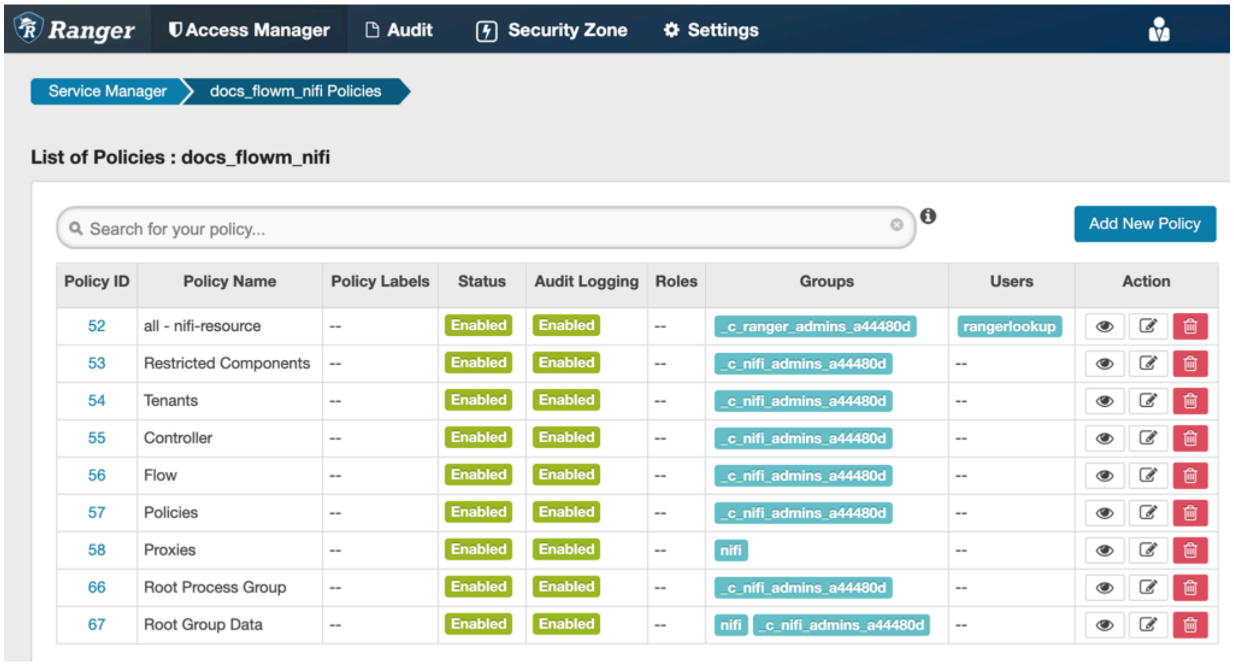
   The following image shows the list of pre-defined policies for NiFi:



   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:



List of Policies : docs_flowm_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.

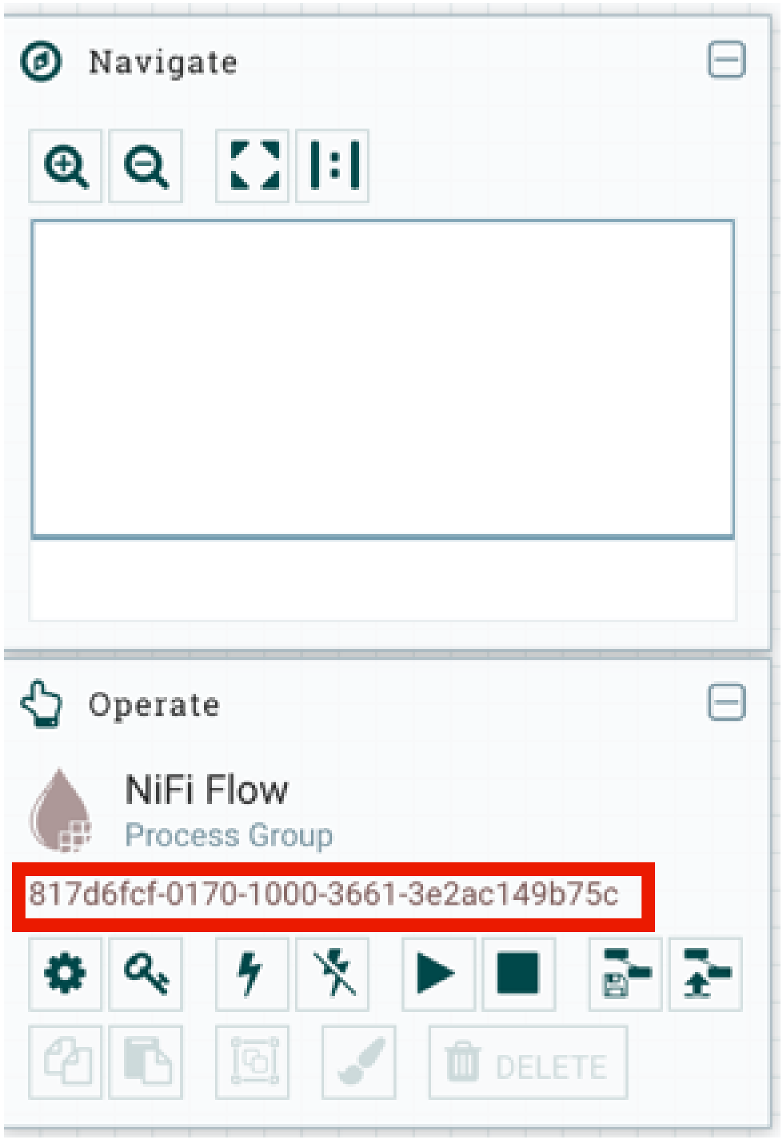
Note:

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

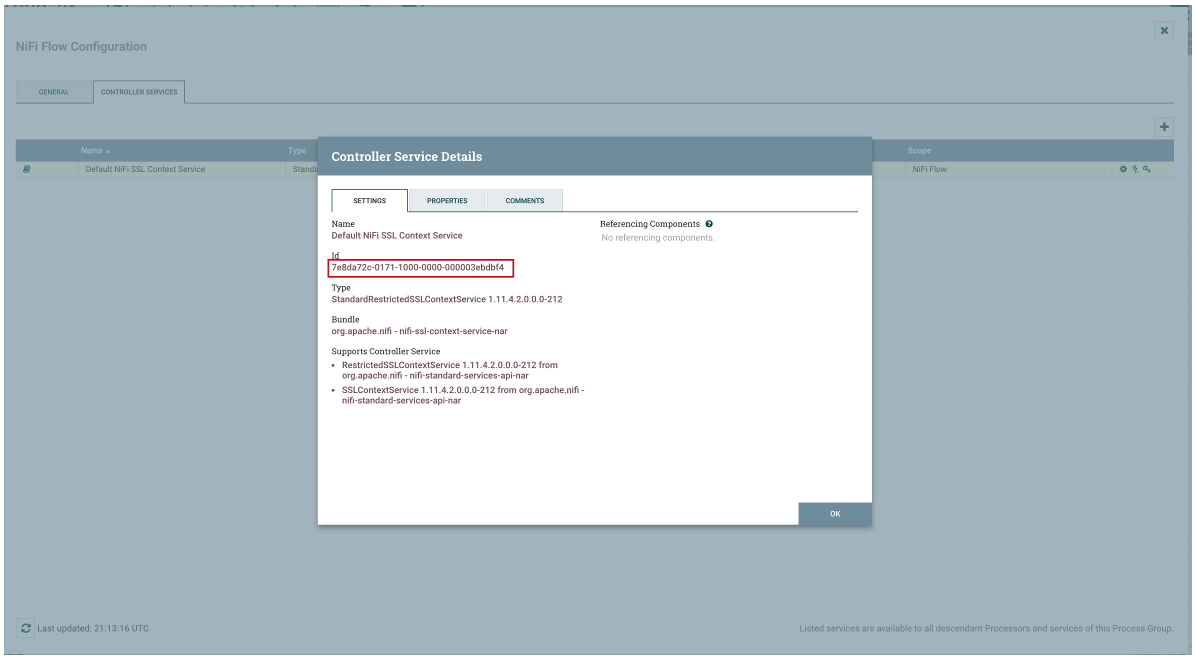
1. From the NiFi canvas, copy the ID of the process group, SSL Context Service, or controller service for reporting tasks that the user needs access to.
2. To locate the ID for a process group:
   a) Click the process group.
   The ID appears in the **Operate** pane.



817d6fcf-0170-1000-3661-3e2ac149b75c

   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.



Name
Default NiFi SSL Context Service

Id
7e8da72c-0171-1000-0000-000003ebdbf4

Type
StandardRestrictedSSLContextService 1.11.4.2.0.0.0-212

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

Supports Controller Service
- RestrictedSSLContextService 1.11.4.2.0.0.0-212 from org.apache.nifi - nifi-standard-services-api-nar
- SSLContextService 1.11.4.2.0.0.0-212 from org.apache.nifi - nifi-standard-services-api-nar

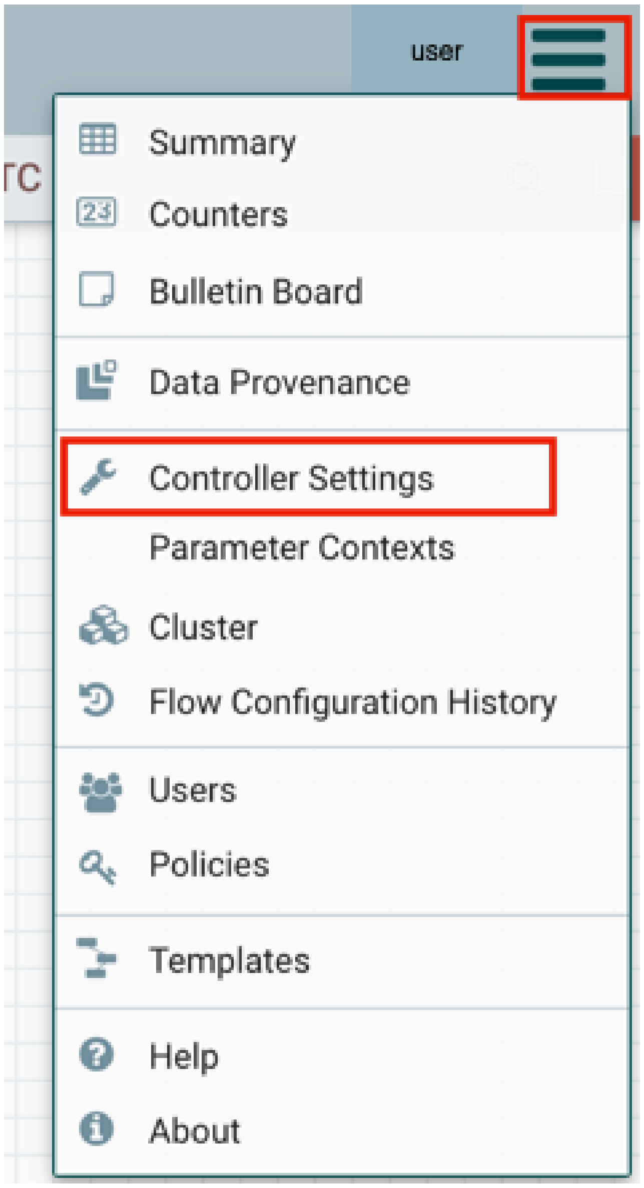
Referencing Components
No referencing components.

Listed services are available to all descendant Processors and services of this Process Group.

4. To locate the ID of a controller service for reporting tasks:

a) Click the process group.

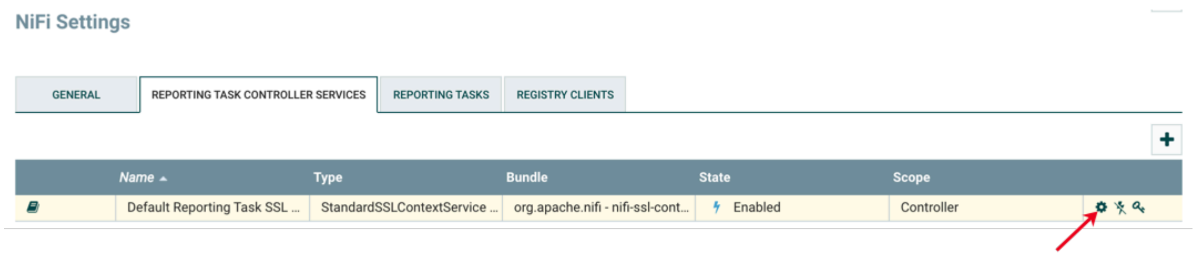
b) Click the menu on the top right of the UI and select Controller Settings.



The **NiFi Settings** page appears.

c) Click the **Reporting Tasks Controller Services** tab.

d) Click the Settings icon for the controller service.



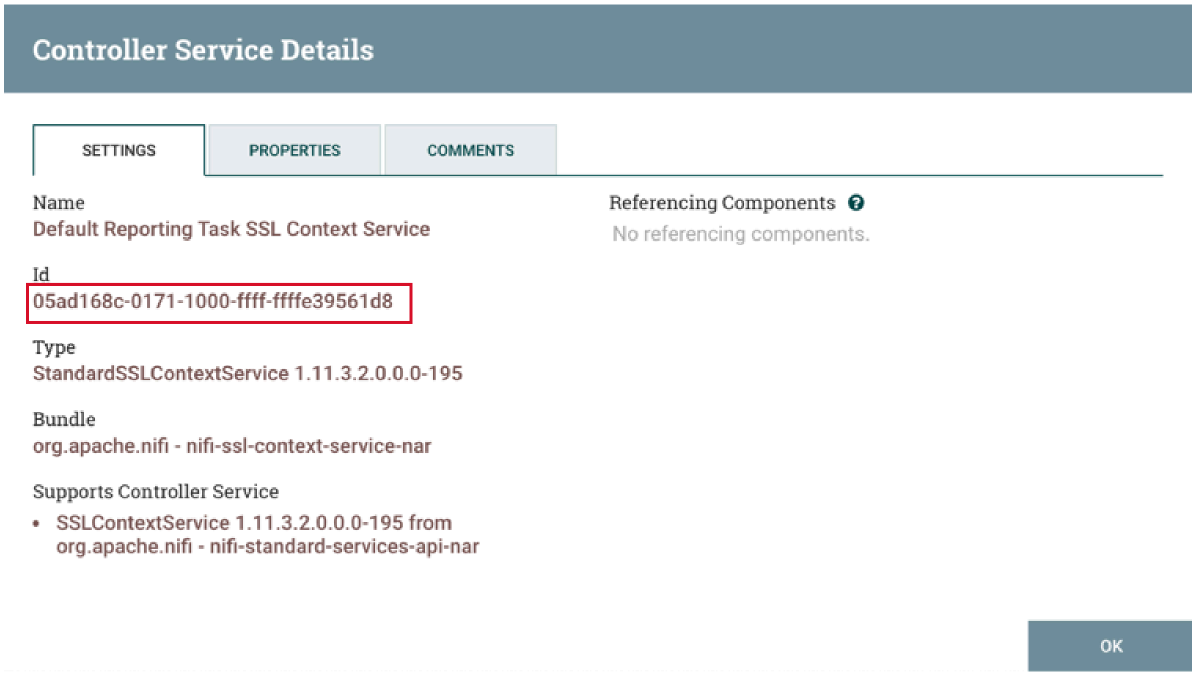
NiFi Settings

GENERAL | REPORTING TASK CONTROLLER SERVICES | REPORTING TASKS | REGISTRY CLIENTS

Name ▲	Type	Bundle	State	Scope
Default Reporting Task SSL ...	StandardSSLContextService ...	org.apache.nifi - nifi-ssl-cont...	Enabled	Controller

The **Controller Service Details** page appears.

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



Controller Service Details

Name
Default Reporting Task SSL Context Service

Id
05ad168c-0171-1000-ffff-ffffe39561d8

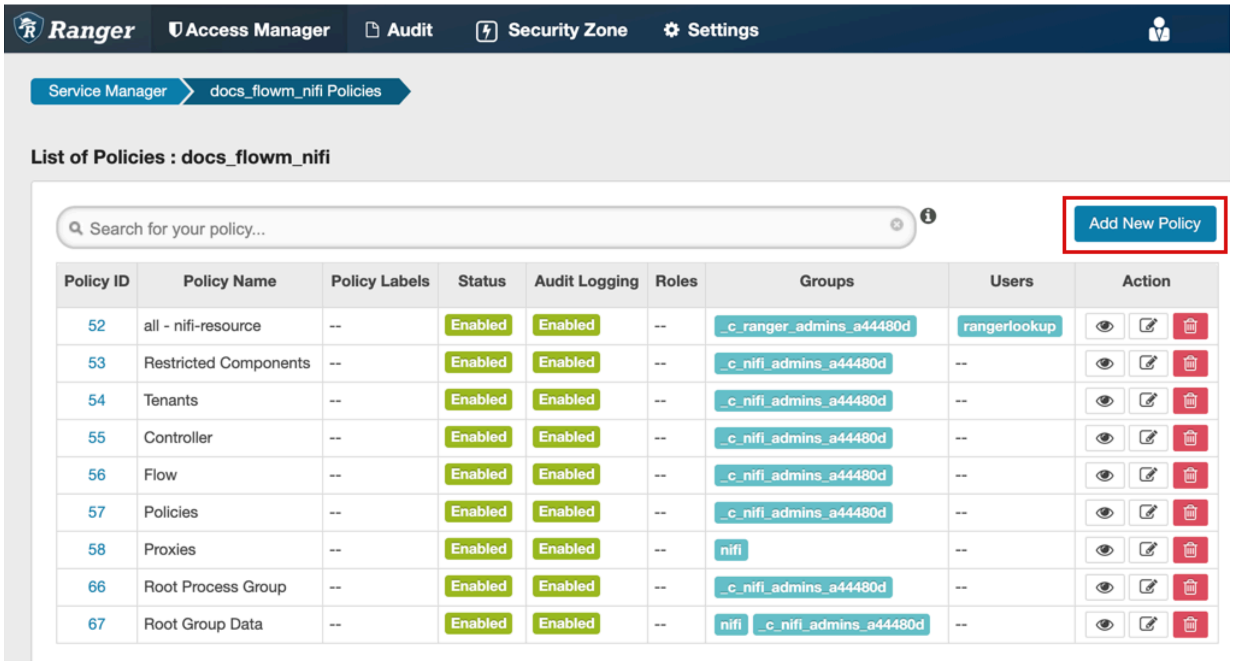
Type
StandardSSLContextService 1.11.3.2.0.0.0-195

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

Supports Controller Service
- SSLContextService 1.11.3.2.0.0.0-195 from org.apache.nifi - nifi-standard-services-api-nar

Referencing Components
No referencing components.

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



List of Policies : docs_flowm_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 **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.

Example

In the following scenario a user requires access to specific NiFi and NiFi resources. You must add the user to the appropriate access 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 pre-defined 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.

Pre-defined Ranger access policies for Apache NiFi

Based on a user’s responsibilities, you can add users to one or more of the following Ranger access policies. When you create a custom policy, use the resource descriptor in the NiFi Resource Identifier field.

The following table lists the pre-defined Ranger access policies for NiFi.

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

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	Allows users to create/modify restricted components assuming other permissions are sufficient. The restricted components may indicate the specific permissions that are required. 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. Some examples of restricted components are ExecuteScript, List/FetchHDFS, and TailFile.	/restricted-components
Root Group Data	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. **Note:** The nifi group is a dynamically managed list of Knox and NiFi node identities. The group exists on all Data Hub Flow Management hosts.	/data/process-groups
Root Group Provenance Data	Allows users to view provenance data.	/provenance-data/process-groups/
Root Process Group	Allows users to view and modify the root process group including adding/removing processors to the canvas. This policy is inherited down the hierarchy unless there is a more specific policy on a component.	/process-groups
Tenants	Allows users to view and modify user accounts and user groups.	/tenants

Pre-defined Ranger access policies for Apache NiFi Registry

Based on a user’s responsibilities, you can add users to one or more of the following Ranger access policies. When you create a custom policy, use the resource descriptor in the NiFi Registry Resource Identifier field.

The following table lists the pre-defined Ranger access policies for NiFi Registry.

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

Ranger Policy	Description	Resource Descriptor
Actuator	Allows users to access the Spring Boot Actuator end-points.	/actuator

Ranger Policy	Description	Resource Descriptor
Buckets	Allows users to view and modify all buckets.	/buckets
Policies	Allows users to view the policies for all components.	/policies
Proxies	Allows proxy machines to send requests on behalf of others.	/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/nifiregistry:
  - 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

Procedure

1. Uncheck Authorizers: Enable File User Group Provider to disable the file-user-group-provider.
2. Uncheck Authorizers: Enable Composite Configurable User Group Provider to disable the composite-configurable-user-group-provider.
3. 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.
4. Check LDAP Enabled to enable ldap-user-group-provider.
   a) Configure all ldap-user-group-provider parameters.
5. Update Authorizers: Ranger Authorizer Property - User Group Provider to use the composite-user-group-provider instead of the configurable one.

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

Procedure

1. Uncheck Authorizers: Enable File User Group Provider to disable the file-user-group-provider.
2. Uncheck Authorizers: Enable Composite Configurable User Group Provider to disable the composite-configurable-user-group-provider.
3. 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.
4. Check LDAP Enabled to enable ldap-user-group-provider.
   a) Configure all ldap-user-group-provider parameters.
5. Update Authorizers: Default File Access Policy Property - User Group Provider to use the composite-user-group-provider instead of the configurable one.

Migrating 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.

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 balance 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.

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 CDP Private Cloud Base 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 CDP Private Cloud Base 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 CDP Private Cloud Base 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 CDP Private Cloud Base cluster.

About this task

You must perform these steps if:

- You want CFM to integrate with Atlas; AND
- The CDP Private Cloud Base 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.

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

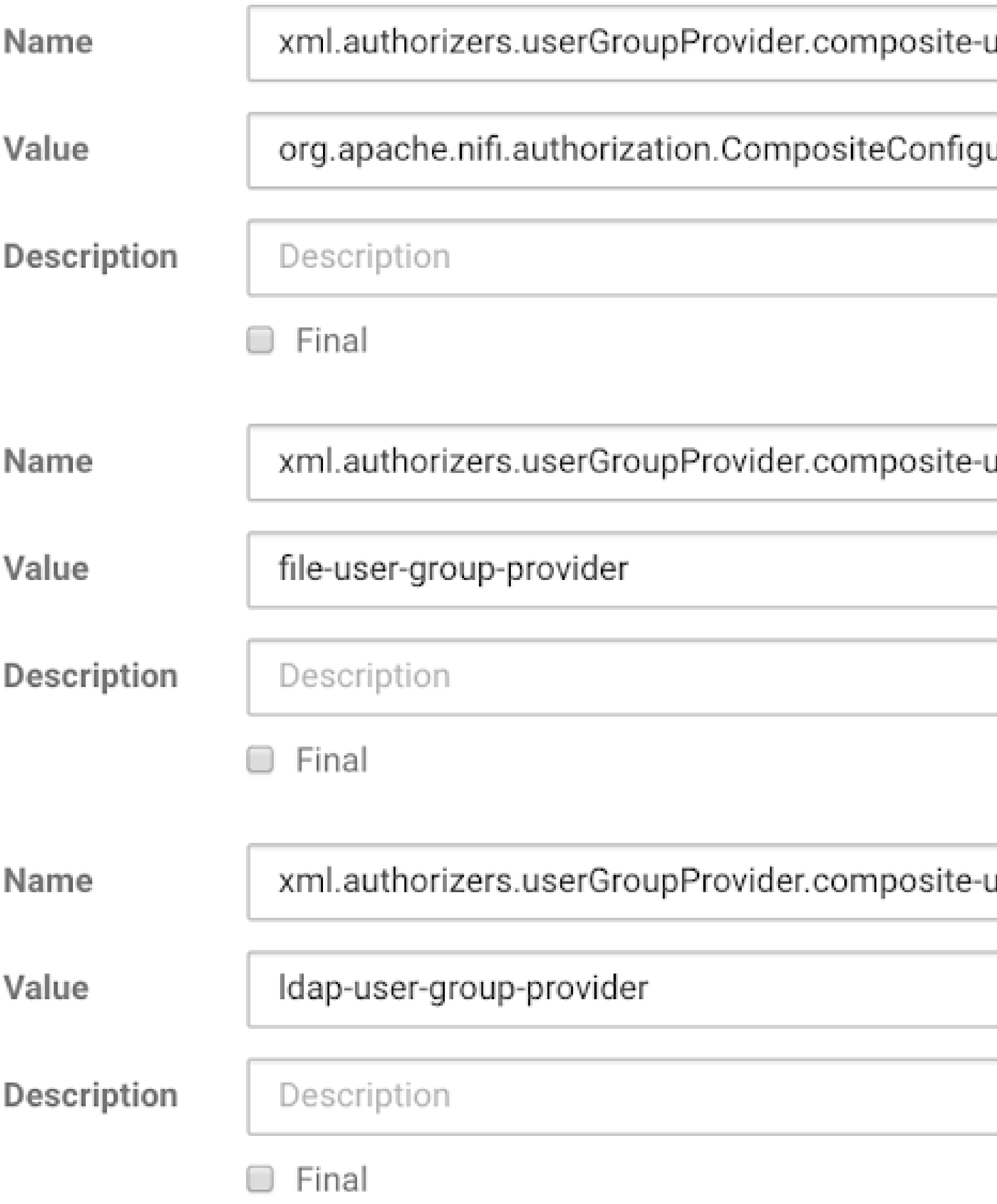
Pairing LDAP with a Composite Group Provider

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		



Name	xml.authorizers.userGroupProvider.composite-u
Value	org.apache.nifi.authorization.CompositeConfigu
Description	Description
	☐ Final
Name	xml.authorizers.userGroupProvider.composite-u
Value	file-user-group-provider
Description	Description
	☐ Final
Name	xml.authorizers.userGroupProvider.composite-u
Value	ldap-user-group-provider
Description	Description
	☐ 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	nifi.web.http.port	8080
HTTPS Port	nifi.web.https.port	8443
Remote Input Socket Port	nifi.remote.input.socket.port	none
Cluster Node Protocol Port	nifi.cluster.node.protocol.port	9088
Cluster Node Load Balancing Port	nifi.cluster.node.load.balance.port	6342
Web HTTP Forwarding Port	nifi.web.http.port.forwarding	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	nifi.registry.web.http.port	18080
HTTPS Port	nifi.registry.web.https.port	18433