Cloudera Runtime 7.2.18

Configuring Apache Zeppelin Security

Date published: 2020-07-28
Date modified: 2024-03-11

https://docs.cloudera.com/

Legal Notice

Contents

Introduction

Zeppelin uses Apache Shiro to provide authentication and authorization (access control).

Clusters created in CDP have security enabled by default. Disabling security on a cluster in CDP is not supported.

Important: Zeppelin will be deprecated in Cloudera Runtime 7.2.18. For more information, see *Deprecation Notices in Cloudera Runtime 7.2.18*.

What to do next

If Ranger is enabled on your cluster, no additional configuration steps are required to have Ranger work with Zeppelin. Note, however, that a Ranger policy change takes about five to ten minutes to take effect.

Configuring Livy

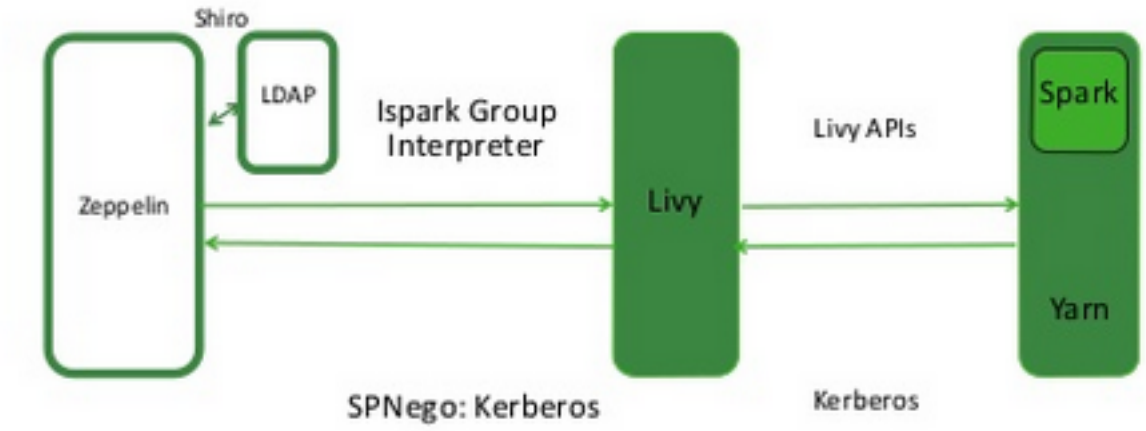
About this task

This section describes how to configure Livy in CDP.

Livy is a proxy service for Apache Spark; it offers the following capabilities:

- Zeppelin users can launch a Spark session on a cluster, submit code, and retrieve job results, all over a secure connection.
- When Zeppelin runs with authentication enabled, Livy propagates user information when a session is created. Livy user impersonation offers an extended multi-tenant experience, allowing users to share RDDs and cluster resources. Multiple users can access their own private data and session, and collaborate on a notebook.

The following graphic shows process communication among Zeppelin, Livy, and Spark:



The following sections describe several optional configuration steps.

Configure Livy user access control

You can use the livy.server.access-control.enabled property to configure Livy user access.

When this property is set to false, only the session owner and the superuser can access (both view and modify) a given session. Users cannot access sessions that belong to other users. ACLs are disabled, and any user can send any request to Livy.

When this property is set to true, ACLs are enabled, and the following properties are used to control user access:

- livy.server.access-control.allowed-users – A comma-separated list of users who are allowed to access Livy.
- livy.server.access-control.view-users – A comma-separated list of users with permission to view other users' infomation, such as submitted session state and statement results.
- livy.server.access-control.modify-users – A comma-separated list of users with permission to modify the sessions of other users, such as submitting statements and deleting the session.

Restart the Livy interpreter after changing settings

If you change any Livy interpreter settings, restart the Livy interpreter. Navigate to the Interpreter configuration page in the Zeppelin Web UI. Locate the Livy interpreter, then click restart.

Verify that the Livy server is running

To verify that the Livy server is running, access the Livy web UI in a browser window. The default port is 8998:

```
http://<livy-hostname>:8998/
```

Livy high availability support

Livy supports high availability. If there is more than one Livy server in the CDP cluster, high availability is automatically enabled by Cloudera Manager by setting the `livy.server.recovery.mode` property to `ha` and by including the list of Zookeeper servers in the Livy configuration.

The new high availability mode runs multiple instances of Livy in a cluster. Both the active and passive instances continuously run and the passive instances take the active role when required. There is only one active server at a time, and the remaining instances are passive. Passive instances only redirect requests to the active server using HTTP 307.

Limitations

- **JDBC connection**: The JDBC connections are not redirected. More details about this limitation are available below.
- **Load balancing**: The active-passive high availability model does not provide additional parallelism or capacity. Interactive sessions are only handled by the active server, adding more servers does not distribute load across servers.

Using Livy without Knox gateway

If you are not using Livy through Knox gateway, clients must follow HTTP redirects and resend authentication. The logic the clients can use is to make a list of the Livy servers by obtaining them from the Cloudera Manager API and if any of the servers do not respond, the clients should retry sending the request to another server in the list. For example, in the case of cURL, the `--location-trusted` flag has to be specified to follow redirects and resend authentication.

Livy high availability and JDBC connection

Livy provides high availability by active-passive setup. Only the active Livy server node can accept JDBC connections, and the passive nodes reject connection attempts. Therefore, if a client wants to connect using JDBC, it has to iterate through all servers and check which one accepts connections. If the active server goes down, the connection is broken and another node takes over the active role. This behavior is the same for both HTTP and binary mode connections.

Configure User Impersonation for Access to Hive

This section describes how to configure Apache Zeppelin user impersonation for Apache Hive.

About this task

User impersonation runs Hive queries under the user ID associated with the Zeppelin session.

Kerberos-enabled Cluster

If Kerberos is enabled on the cluster, enable user impersonation as follows:

To configure the %jdbc interpreter, complete the following steps:

1. In Hive configuration settings, set hive.server2.enable.doAs to true.
2. In the Zeppelin UI, navigate to the %jdbc section of the Interpreter page.
3. Enable authentication via the Shiro configuration: specify authorization type, keytab, and principal.
   a. Set zeppelin.jdbc.auth.type to KERBEROS.
   b. Set zeppelin.jdbc.principal to the value of the principal.
   c. Set zeppelin.jdbc.keytab.location to the keytab location.
4. Set hive.url to the URL for HiveServer2. Here is the general format:

   ```
   jdbc:hive2://HiveHost:10001/default;principal=hive/_HOST@HOST1.COM;hive.
   server2.proxy.user=testuser
   ```

   The JDBC interpreter adds the user ID as a proxy user, and sends the string to HiveServer2; for example:

   ```
   jdbc:hive2://dkhdp253.dk:2181,dkhdp252.dk:2181,dkhdp251.dk:2181/;service
   DiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2
   ```
5. Add a hive.proxy.user.property property and set its value tohive.server2.proxy.user.
6. Click Save, then click restart to restart the interpreter.

For information about authenticating Zeppelin users through Active Directory or LDAP, see "Configuring Zeppelin Security" in this guide.

Configure User Impersonation for Access to Phoenix

This section describes how to configure Apache Zeppelin user impersonation for Apache Phoenix.

About this task

User impersonation runs Phoenix queries under the user ID associated with the Zeppelin session.

To enable user impersonation for Phoenix, complete the following steps:

Procedure

1. In the HBase configuration settings, enable phoenix sql.
2. In Advanced HBase settings, set the following properties:

   ```
   hbase.thrift.support.proxyuser=true
   hbase.regionserver.thrift.http=true
   ```

3. In HDFS configuration settings, set the following properties:

```
hadoop.proxyuser.hbase.groups=*
hadoop.proxyuser.hbase.hosts=*
hadoop.proxyuser.zeppelin.groups=*
hadoop.proxyuser.zeppelin.hosts=*
```

4. Make sure that the user has access to HBase. You can verify this from the HBase shell with the user_permissions command.

Enabling Access Control for Zeppelin Elements

This section describes how to restrict access to specific Apache Zeppelin elements.

After configuring authentication, you may want to restrict access to Zeppelin notes and data, and also set restrictions on what users and groups can configure Zeppelin interpreters. You can authorize access at three levels within Zeppelin:

- UI authorization restricts access to Zeppelin Interpreter, Credential, and Configuration pages based on administrative privileges.
- Note-level authorization restricts access to notes based on permissions (owner, reader, or writer) granted to users and groups.
- Data-level authorization restricts access to specific data sets.

Enable Access Control for Interpreter, Configuration, and Credential Settings

This section describes how to restrict access to Apache Zeppelin interpreter, credential, and configuration settings.

About this task

By default, any authenticated account can access the Zeppelin interpreter, credential, and configuration settings. When access control is enabled, unauthorized users can see the page heading, but not the settings.

Before you begin

Users and groups must be defined on all Zeppelin nodes and in the associated identity store.

Procedure

1. Define a [roles] section in shiro.ini contents, and specify permissions for defined groups. The following example grants all permissions ("*") to users in group admin:

```
[roles]
admin = *
```

2. In the [urls] section of the shiro.ini contents, uncomment the interpreter, configurations, or credential line(s) to enable access to the interpreter, configuration, or credential page(s), respectively. (If the [urls] section is not defined, add the section. Include the three /api lines listed in the following example.)

   The following example specifies access to interpreter, configurations, and credential settings for role "admin":

```
[urls]
/api/version = anon
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
#/** = anon
```

```
/** = authc
```

To add more roles, separate role identifiers with commas inside the square brackets.

Note: The sequence of lines in the [urls] section is important. The /api/version line must be the first line in the [urls] section:

```
/api/version = anon
```

Next, specify the three /api lines in any order:

```
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
```

The authc line must be last in the [urls] section:

```
/** = authc
```

3. Map the roles to LDAP or Active Directory (AD) groups. The following is an example of the shiro.ini settings for Active Directory (before pasting this configuration in your Zeppelin configuration, update the Active Directory details to match your actual configuration settings).

```
# Sample LDAP configuration, for Active Directory user Authentication, c
urrently tested for single Realm
[main]
ldapRealm=org.apache.zeppelin.realm.LdapRealm
ldapRealm.contextFactory.systemUsername=cn=ldap-reader,ou=ServiceUsers,dc=
lab,dc=hortonworks,dc=net
ldapRealm.contextFactory.systemPassword=SomePassw0rd
ldapRealm.contextFactory.authenticationMechanism=simple
ldapRealm.contextFactory.url=ldap://ad.somedomain.net:389
# Ability to set ldap paging Size if needed; default is 100
ldapRealm.pagingSize=200
ldapRealm.authorizationEnabled=true
ldapRealm.searchBase=OU=CorpUsers,DC=lab,DC=hortonworks,DC=net
ldapRealm.userSearchBase=OU=CorpUsers,DC=lab,DC=hortonworks,DC=net
ldapRealm.groupSearchBase=OU=CorpUsers,DC=lab,DC=hortonworks,DC=net
ldapRealm.userObjectClass=person
ldapRealm.groupObjectClass=group
ldapRealm.userSearchAttributeName = sAMAccountName
# Set search scopes for user and group. Values: subtree (default), onel
evel, object
ldapRealm.userSearchScope = subtree
ldapRealm.groupSearchScope = subtree
ldapRealm.userSearchFilter=(&(objectclass=person)(sAMAccountName={0}))
ldapRealm.memberAttribute=member
# Format to parse & search group member values in 'memberAttribute'
ldapRealm.memberAttributeValueTemplate=CN={0},OU=CorpUsers,DC=lab,DC=horto
nworks,DC=net
# No need to give userDnTemplate if memberAttributeValueTemplate is provid
ed
#ldapRealm.userDnTemplate=
# Map from physical AD groups to logical application roles
ldapRealm.rolesByGroup = "hadoop-admins":admin_role,"hadoop-users":hado
op_users_role
# Force usernames returned from ldap to lowercase, useful for AD
ldapRealm.userLowerCase = true

# Enable support for nested groups using the LDAP_MATCHING_RULE_IN_CHAIN
 operator
ldapRealm.groupSearchEnableMatchingRuleInChain = true
```

```
sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
### If caching of user is required then uncomment below lines
cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
securityManager.cacheManager = $cacheManager

securityManager.sessionManager = $sessionManager
securityManager.realms = $ldapRealm
# 86,400,000 milliseconds = 24 hour
securityManager.sessionManager.globalSessionTimeout = 86400000
shiro.loginUrl = /api/login

[urls]
# This section is used for url-based security.
# You can secure interpreter, configuration and credential information by
urls. Comment or uncomment the below urls that you want to hide.
# anon means the access is anonymous.
# authc means Form based Auth Security
# To enfore security, comment the line below and uncomment the next one
#/api/version = anon
/api/interpreter/** = authc, roles[admin_role,hadoop_users_role]
/api/configurations/** = authc, roles[admin_role]
/api/credential/** = authc, roles[admin_role,hadoop_users_role]
#/** = anon
/** = authc
```

Additional information:

- ldapRealm.rolesByGroup = "hadoop-admins":admin_role,"hadoop-users":hadoop_users_role

  This line maps the AD groups "hadoop-admins" and "hadoop-users" to custom roles which can be used in the [urls] section to control access to various Zeppelin users. Note that the short group names are to be used rather than fully qualified names such as "cn=hadoop-admins,OU=CorpUsers,DC=lab,DC=hortonworks,DC=net". The role names can be set to any name but the names should match those used in the [urls] section.
- ldapRealm.groupSearchEnableMatchingRuleInChain = true

  A very powerful option to search all of the groups that a given user is member of in a single query. An LDAP search query with this option traverses the LDAP group hierarchy and finds all of the groups. This is especially useful for nested groups. More information can be found here. Caution: this option can cause performance overhead (slow to log in, etc.) if the LDAP hierarchy is not optimally configured.
- ldapRealm.userSearchFilter=(&(objectclass=person)(sAMAccountName={0}))

  Use this search filter to limit the scope of user results when looking for a user's Distinguished Name (DN). This is used only if userSearchBase and userSearchAttributeName are defined. If these two are not defined, userDnTemplate is used to look for a user's DN.

4. When unauthorized users attempt to access the interpreter, configurations, or credential page, they can see the page heading, but not the settings.

Enable Access Control for Notebooks

This section describes how to restrict access to Apache Zeppelin notebooks by granting permissions to specific users and groups.

About this task

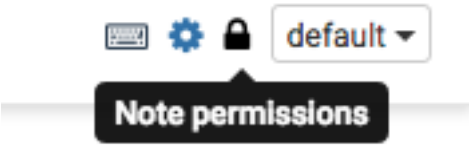
There are two main steps in this process: defining the searchBase property in the Zeppelin Shiro configuration, and then specifying permissions.

Procedure

1. In Zeppelin configuration settings, the Zeppelin administrator should specify activeDirectoryRealm.searchBase or ldapRealm.searchBase, depending on whether Zeppelin uses AD or LDAP for authentication. The value of sear chBase controls where Zeppelin looks for users and groups.

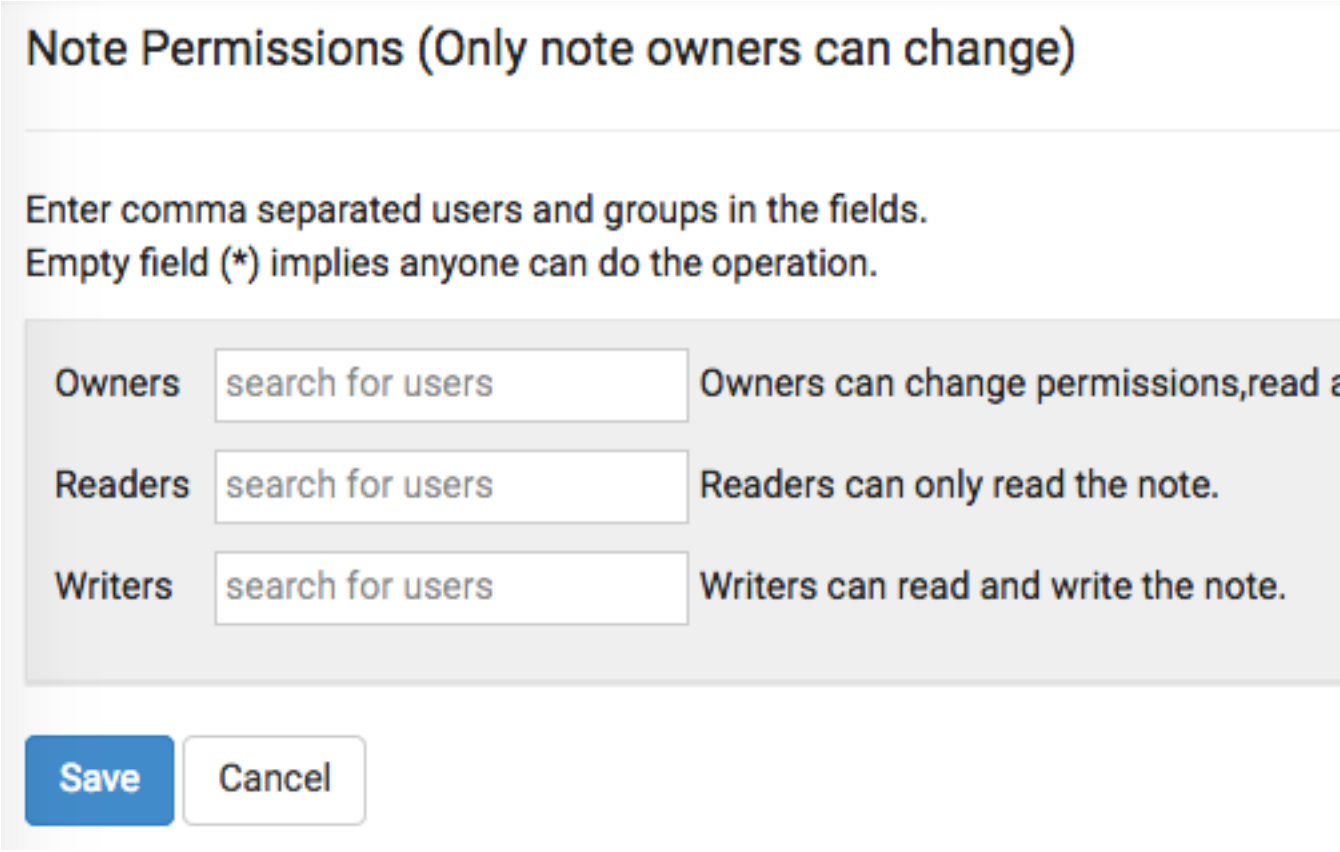
   For more information, refer to "Shiro Settings: Reference" in this guide. For an example, see "Configure Zeppelin for Authentication: LDAP and Active Directory" in this guide.
2. The owner of the notebook should navigate to the note and complete the following steps:

   a. Click the lock icon on the notebook:



   b. Zeppelin presents a popup menu. Enter the user and groups that should have access to the note. To search for an account, start typing the name.

   Note: If you are using Shiro as the identity store, users should be listed in the [user]section. If you are using AD or LDAP users and groups should be stored in the realm associated with your Shiro configuration.



Enable Access Control for Data

This section describes how to restrict access to Apache Zeppelin data.

Access control for data brought into Zeppelin depends on the underlying data source:

- To configure access control for Spark data, Zeppelin must be running as an end user ("identity propagation"). Zeppelin implements access control using Livy. When identity propagation is enabled via Livy, data access is controlled by the type of data source being accessed. For example, when you access HDFS data, access is controlled by HDFS permissions.
- To configure access control for Hive data, use the JDBC interpreter.
- To configure access control for the Spark shell, define permissions for end users running the shell.

Shiro Settings: Reference

This section provides additional information about the Shiro settings used to configure Apache Zeppelin security.

Active Directory Settings

This section provides additional information about Shiro Active Directory settings.

Active Directory (AD) stores users and groups in a hierarchical tree structure, built from containers including the organizational unit (ou), organization (o), and domain controller (dc). The path to each entry is a Distinguished Name (DN) that uniquely identifies a user or group.

User and group names typically have attributes such as a common name (cn) or unique ID (uid).

Specify the DN as a string, for example cn=admin,dc=example,dc=com. White space is ignored.

activeDirectoryRealm

specifies the class name to use for AD authentication. You should set this to org.apache.zeppelin.realm.ActiveDirectoryGroupRealm.

activeDirectoryRealm.url

specifies the host and port where Active Directory is set up. .

If the protocol element is specified as ldap, SSL is not used. If the protocol is specified as ldaps, access is over SSL.

Note: If Active Directory uses a self-signed certificate, import the certificate into the truststore of the JVM running Zeppelin; for example:

```
echo -n | openssl s_client –connect ldap.example.com:389 | \
    sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/
examplecert.crt

keytool –import \
    -keystore $JAVA_HOME/jre/lib/security/cacerts \
    -storepass changeit \
    -noprompt \
    -alias mycert \
    -file /tmp/examplecert.crt
```

activeDirectoryRealm.principalSuffix

simplifies the logon information that users must use to log in. Otherwise, AD requires a username fully qualified with domain information. For example, if a fully-qualified user account is user@hdpqa.example.com, you can specify a shorter suffix such as user@hdpqa.

```
activeDirectoryRealm.principalSuffix = @<user-org-level-domain>
```

activeDirectoryRealm.searchBase

defines the base distinguished name from which the directory search starts. A distinguished name defines each entry; "dc" entries define a hierarchical directory tree.

activeDirectoryRealm.systemUsername
activeDirectoryRealm.systemPassword

defines the username and password that Zeppelin uses to connect to Active Directory when it searches for users and groups. These two settings are used for controlling access to UI features, not for authentication. The Bind method does not require a valid user password.

Example: activeDirectoryRealm.systemPassword = passwordA

activeDirectoryRealm.groupRolesMap

a comma-separated list that maps groups to roles. These settings are used by Zeppelin to restrict UI features to specific AD groups. The following example maps group hdpdv_admin at hdp3.example .com to the "admin" role:

```
CN=hdpdv_admin,DC=hdp3,DC=example,DC=com:admin
```

activeDirectoryRealm.authorizationCachingEnabled

specifies whether to use caching to improve performance. To enable caching, set this property to true.

LDAP Settings

This section provides additional information about Shiro LDAP settings.

LDAP stores users and groups in a hierarchical tree structure, built from containers including the organizational unit (ou), organization (o), and domain controller (dc). The path to each entry is a Distinguished Name (DN) that uniquely identifies a user or group.

User and group names typically have attributes such as a common name (cn) or unique ID (uid).

Specify the DN as a string, for example cn=admin,dc=example,dc=com. White space is ignored.

Zeppelin LDAP authentication uses templates for user DNs.

ldapRealm

specifies the class name to use for LDAP authentication. You should set this to org.apache.zeppelin. realm.LdapRealm unless you are familiar with LDAP and prefer to use org.apache.shiro.realm.ldap. JndiLdapRealm. ..

ldapRealm.contextFactory.environment[ldap.searchBase]

defines the base distinguished name from which the LDAP search starts. Shiro searches for user DnTemplate at this address.

If the protocol is specified as ldap, SSL is not used. If the protocol is specified as ldaps, access is over SSL.

ldapRealm.userDnTemplate

specifies the search location where the user is to be found. Shiro replaces {0} with the username acquired from the Zeppelin UI. Zeppelin uses User DN templates to configure associated realms.

ldapRealm.contextFactory.url

specifies the host and port on which LDAP is running.

If the protocol element is specified as ldap, SSL will not be used. If the protocol is specified as ldaps, access will be over SSL.

Note: If LDAP is using a self-signed certificate, import the certificate into the truststore of JVM running Zeppelin; for example:

```
echo -n | openssl s_client –connect ldap.example.com:389 | \
    sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/
examplecert.crt

keytool –import \
    -keystore $JAVA_HOME/jre/lib/security/cacerts \
    -storepass changeit \
    -noprompt \
    -alias mycert \
    -file /tmp/examplecert.crt
```

ldapRealm.contextFactory.systemUsername
ldapRealm.contextFactory.systemPassword

define the username and password that Zeppelin uses to connect to LDAP, to search for users and groups. These two settings are used for controlling access to UI features, not for authentication. The Bind method does not require a valid user password.

Examples:

```
ldapRealm.contextFactory.systemUsername=uid=guest,ou=people,dc=h
adoop,dc=apache,dc=org
```

```
ldapRealm.contextFactory.systemPassword=somePassword
```

ldapRealm.authorizationCachingEnabled

specifies whether to use caching to improve performance. To enable caching, set this property to true.

General Settings

This section provides additional information about Shiro general settings.

securityManager.sessionManager.globalSessionTimeout

specifies how long to wait (in milliseconds) before logging out a user, if they are logged in and are not moving the cursor.

The default is 86,400,000 milliseconds, which equals 24 hours.

shiro.ini Example

The following example shows a minimum set of shiro.ini settings for authentication and access control for a Zeppelin deployment that uses Active Directory.

Before you begin

In this example, the corresponding account information is configured in Active Directory (at adhost.field.hortonworks .com) and on Zeppelin nodes.

```
[main]
# AD authentication settings
activeDirectoryRealm = org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
activeDirectoryRealm.url = ldap://adhost.org.hortonworks.com:389
activeDirectoryRealm.searchBase = DC=org,DC=hortonworks,DC=com
```

```
activeDirectoryRealm.systemUsername
activeDirectoryRealm.systemPassword

# general settings
sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
securityManager.cacheManager = $cacheManager
securityManager.sessionManager = $sessionManager
securityManager.sessionManager.globalSessionTimeout = 86400000
shiro.loginUrl = /api/login

[roles]
admin = *

[urls]
# authentication method and access control filters
/api/version = anon
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
#/** = anon
/** = authc
```