ISILON ONEFS WITH HADOOP KERBEROS AND IDENTITY MANAGEMENT APPROACHES

Technical Solution Guide

Hadoop and OneFS cluster configurations for secure access and file permissions management

ABSTRACT

This technical solution guide provides an overview of the approaches to security and identity management when integrating Isilon OneFS with Hadoop clusters.

28 February 2018

WHITE PAPER

TABLE OF CONTENTS

Executive Summary

It is well documented how to directly integrate and Kerberize a Hadoop cluster and an Isilon OneFS cluster against a KDC. This paper will highlight the considerations and potential different approaches to integrating Kerberos authentication and identity management when using Hadoop distributions with OneFS.

Audience

This guide is intended for Hadoop systems administrators, storage administrators, IT architects, and IT managers who will be running OneFS with Hadoop.

Introduction

It is important to understand the implications when transitioning a Hadoop cluster with the Isilon integration from a simple security model to a Kerberized authentication model. This is especially true with integrated KDC and LDAP servers, for example Active Directory, as not only does it bring Kerberos authentication in through a KDC, it also brings identity management into the solution using LDAP. The basis of this paper addresses the added options and implications of different identity management approaches when using an Isilon cluster with Hadoop.

It's important to recognize the differences between authentication (I am who I say I am) and identity management (this is who I am), and why this becomes critical when looking at strategies to integrate with the Hadoop and Isilon clusters. Ultimately, access to files is determined by the file permissions, but the initial access is governed by authentication and identity management. Historically, security and file access were very loosely controlled in Hadoop clusters, with only the username usually being used. However, authentication and identity management have become more integrated and critical as Hadoop has become more enterprise-focused.

Solution Purpose

The solutions described in this document provide:

- An overview of the implication of Kerberos and identity management in Hadoop and Isilon OneFS clusters.
- Potential configuration deployments and approaches with Isilon-integrated Hadoop clusters.
- Not all potential configurations will be discussed; many others will exist and will function, but they are out of scope for this document.

In order to provide the potential deployment scenarios and configurations, we need to review the basic components of how authentication and identity management work in Hadoop and Isilon clusters.

Hadoop Authentication and Identity Management

Traditional Hadoop clusters implement a very simple model to manage users and file access in which ultimately the cluster only uses local user accounts and local group membership for identity and access management. In this model, the cluster accepts that you are who you say you are, based on your username and group membership; no authentication is actually occurring. If you make a request, and you are the HDFS or Yarn user, the Hadoop cluster accepts this and allows you to do whatever the HDFS or Yarn user have the permissions to do. In this model, it is also important to recognize that the underlying UID and GID of a user or group are not considered important, as only the username or group name matter. This reliance on user and group memberships only made the deployment of Hadoop clusters straightforward initially.

As enterprise security requirements become more important, securing the Hadoop cluster and its data has become more common. The primary method of achieving this is through Kerberization of the Hadoop cluster. *Kerberos* is an industry standard authentication protocol used to secure systems and has been widely deployed for a long time. It is important to recognize that Kerberos only provides *authentication*—it provides a method for an entity, a system, or a user to prove only who they say they are. Having proved who they say they are, secure file access can be enforced through appropriate methods of *authorization*. Now, when limiting specific systems, users, or groups to resources, we can be confident that this is enforced correctly, as we can rely on Kerberos to provide a valid mechanism for ensuring who a user is. With Kerberos, we can be confident that only the specific users who have been granted access actually have it, and all other users are denied access.

Ultimately, authorization will rely on identity management and access control to determine who has access to the system, but without Kerberos to ensure that legitimate authentication is occurring, access control cannot be enforced successfully. We cannot have confidence in secure access control when locking access down to a single user if anyone can impersonate that user without issue. Kerberos ensures that a principal in the system or *Realm* (in Kerberos terms) is who they say they are.

This separation of authentication and authorization is critical to understanding how secure access control is implemented. When implementing a Hadoop cluster with Isilon, you must address both the authorization and identity management components of the integration.

Accounts

Within a Hadoop cluster, two distinct types of accounts will exist. Before looking at deployment scenarios, let's review the type of accounts, and the primary purpose for using those accounts.

Service Accounts

These accounts are used to run all of the underlying Hadoop cluster services: HDFS, Yarn, Mapred, Hbase, Hive, and so on. There is usually one account for every service deployed. Typically these are created (on Hadoop cluster installations) on every deployed host by the Hadoop administrator. These accounts can be customized, pre-created, or exist in a directory service depending on the deployment model. However, in most cases, they are created as local users and local groups with the same identity on all hosts at the Hadoop service install time. This provides a common set of service accounts across the entire cluster to run and manage services.

User Accounts

A user account is typically a user who will run jobs or tasks on data in the Hadoop cluster. A user account can be created as a local account on a single host, multiple hosts, and even on all hosts, depending on the deployment model. A user account can also be a common Directory Service-based account that is available to all hosts.

The implementation of using local accounts or Directory Service accounts is beyond the scope of this document and will depend on the requirements of the cluster and infrastructure available.

For the purpose of the design patterns coming up we need to recognize that any account, either service or user can exist as a local or Directory Service based account, as this will have implications on the deployment scenarios and configuration required to implement secure Hadoop clusters with Isilon.

Authentication

Kerberos

In simple Hadoop security models, any user could impersonate any user just by becoming that user and accessing the Hadoop data as that user. No access check was performed if that user could legitimately impersonate that user. After the cluster is Kerberized, a user or service has to be able to prove who they say they are. This is achieved by providing additional information when attempting to impersonate an entity. In the case of a user, this is usually a password, and for a service, this is achieved by providing access to an encrypted password file known as a *keytab*. Having authenticated, the entity now has a valid Kerberos ticket, which is used to prove who they say they are to any other services or users.

When running a Kerberos-secured environment, it is important to recognize the types of principals (users or services) that can exist and that are required to provide proof of their identity. The design, implementation, and configuration of how Kerberos works are beyond the scope of this paper but many detailed documents are available online on these topics.

User Principal Names – UPNs

User Principal Names (UPNs) are primarily users who access the system, run jobs, and need access to data. A UPN is created in the Key Distribution Center (KDC) where all Kerberos accounts or principals exist, and it is usually secured with a password known only to the user that will use the account. Therefore a UPN can confirm its identity by providing the password when challenged -- ensuring that only users who know the password can prove they are, in fact, that user. UPNs take the form of `username@KERBEROS_REALM` and usually have an associated complex password. In a secure Hadoop cluster, a user wishing to run a job and access specific data now has to ensure that they are who they say they are by providing the username and password of the account they wish to access the system as. Typically a UPN is linked to a user identity of the same name, which provides the primary and supplemental identities of the user, either as a UID, GIDs or SIDs. In many installs, these UPNs are a directory service-based user accounts that are deployed across all hosts in the Hadoop environment.

Service Principal Names – SPNs

Service Principal Names (SPNs) are accounts for services that operate inside a secure Kerberos realm. Since we wish to enforce secure access between systems, we also require a mechanism for services that run on different systems to be able to authenticate themselves and prove their identity in a similar way to users. When a service, for example: HDFS, LDAP, Yarn, or HBase, wants to operate inside a secure Kerberos environment, we need a mechanism for the service to prove who it is as well. This is where SPNs come into play. An SPN for the service is created in the KDC similar to a user's UPN, but instead of a password an encrypted password file is generated and copied to the host running the service. It is secured, so only that service account has access. This keytab file takes the place of an interactive password. Every time a service interacts, it would be unrealistic to type a password, so the service uses the encrypted keytab file to provide the password when authenticating itself. SPNs take the form of `service/hostname_fqdn@REALM` and, along with the keytab, this SPN allows services to make and receive secure Kerberos connections by proving the service is who they say they are, in a similar way to an authenticated user. An SPN is also associated with an identity, so that following authentication it can access resources using its associated identity. In most Hadoop and Isilon clusters, these service accounts begin as local accounts created on installation; it is common to continue using these local identities that are now associated with a KDC-based SPN.

The combination of UPNs and SPNs allow users and system services to interact and prove their identities and therefore ensure valid authentication within a Kerberos system. Having guaranteed authentication, we can next look to authorization and how we can enforce access to data through valid access control.

Authorization

Access Control

Access Control is the method of enforcing access restrictions to a resource based on an access control list that specifies who can access and what they can do with the resource. The scope of the access control model and permissions is beyond this whitepaper, but what is critical is "the who" or a list of users and groups who can appear in these access lists. Since we have implemented Kerberos to enforce "I am who I say I am" authentication, we can leverage identity management to create valid "who am I" for these principals and then add these users and groups to the access control list and be confident that authentication ensures validity.

Identity Management

All users and groups must have an identity. Simply put, they need a name and some form of an alphanumeric identifier, for example: a UID or SID, as well as many other attributes. Identity Management is the allocation of attributes and groupings of these entities into collections that provide some extended value. A number of users are grouped into a defined collection to provide some common behavior or meaning. For example, all the system administrator user accounts are members of the sysadmin group. The advantage of using groups is to abstract the individual into an identifiable group that can be used with access control to enforce the capability of the principal within the system. Two primary forms of Identity exist:

- Local Accounts – These are host-specific users and groups only available to a specific host
- Directory Service Accounts – These users and groups exist in a central repository available to many hosts, for example LDAP, AD, and NIS

The use of local or directory accounts is important to understand when deploying distributed systems, as the access control will be based on these identities post-authentication. A principal having proved its identity through Kerberos is now dependent on identity management to provide the information on "who they are." Successful access control can now be enforced, as we know that a principal is who they say they are, and that they are this identity and have other additional identities.

It is important to review the common configurations, as the upcoming deployed security models will depend on how not only the authentication is deployed but also the identity management. When setting up or deploying accounts, principals, and identities, the following options are available and can be used for Hadoop and Isilon integration for both Hadoop Service and User accounts:

Common Account Locations and Basic Properties

Local Account: No authentication with per host identity management

- Local account with ID – Name, ID
- Local account is a member of Local Groups – Name, ID, Members

Kerberos Principal: Will provide authentication

- Kerberos principal in KDC; UPN or SPN – Name, Password, Keytab

Local Account + Kerberos Principal: Authentication with per host identity management

- Local account with ID – Name, ID
- Local account is a member of Local Groups – Name, ID, Members
- Kerberos principal in KDC; UPN, or SPN

Directory Service Account* + Kerberos Principal: Authentication with central identity management

- Directory Based Account ID – Name, ID
- Directory Based Account is a member of Directory Based Groups
- Kerberos principal in KDC; UPN, or SPN

* Implementation of directory-based Hadoop service accounts is not covered in this document.

The different implementations have different use cases and provide different capabilities and will be used to support different security and access control models, as we will discuss later in this white paper. As was mentioned in the previous section, it is often common to see a mixed configuration of accounts in an integrated Hadoop and Isilon cluster. This will simplify deployment but provide a secure but scalable and flexible deployment.

- Local Service Accounts for Hadoop Services – Deployed by the Hadoop distribution at install time
- Local Service Account on Isilon for Hadoop Services – Deployed at Access Zone setup time for the Hadoop cluster*
- Directory-based User Account for end-user job submission and management

*See the following blog post: Isilon and Hadoop Local User UID Parity

Kerberos Principals

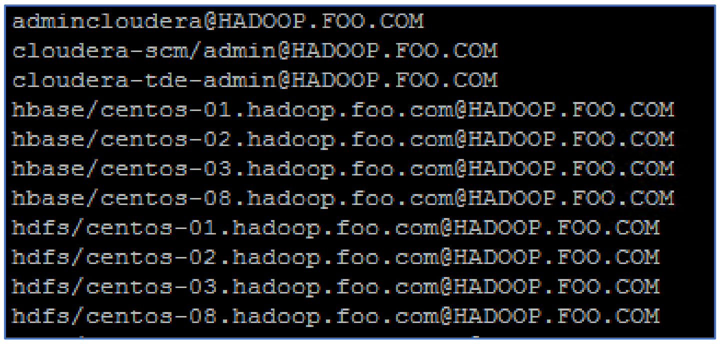
During the Kerberization process, the wizards in either Cloudera CDH or Hortonworks Ambari-based HDP create a number of Kerberos principals and keytabs in the KDC. It is important to review the differences between these different principal types and the distributions, and to understand the multitenancy scenarios and implications for deployment with OneFS.

KDC and Active Directory Principals

In order to fully understand the implications of Kerberized Hadoop with Isilon, we have to contrast the Kerberos principals that are created, along with how different distributions deal with the requirements of the KDCs and Hadoop cluster to support fully Kerberized services.

MIT KDC Kerberos Principals

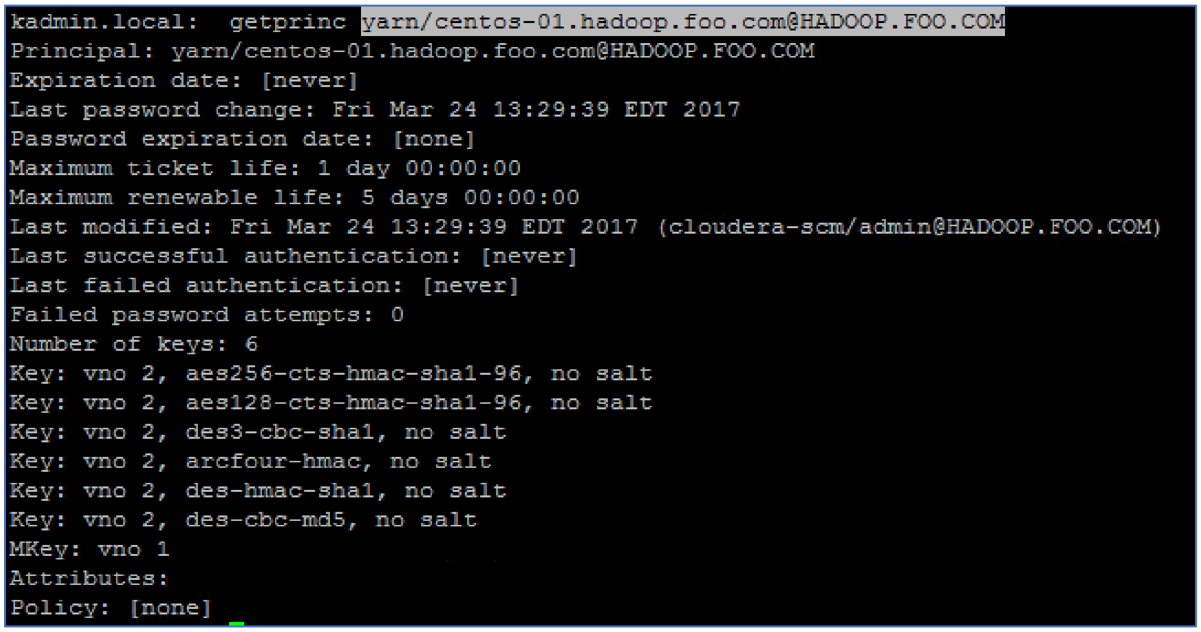
The Kerberization process on the Hadoop cluster generates the required SPN Principal in the MIT-based KDC as follows:



```
admincloudera@HADOOP.FOO.COM
cloudera-scm/admin@HADOOP.FOO.COM
cloudera-tde-admin@HADOOP.FOO.COM
hbase/centos-01.hadoop.foo.com@HADOOP.FOO.COM
hbase/centos-02.hadoop.foo.com@HADOOP.FOO.COM
hbase/centos-03.hadoop.foo.com@HADOOP.FOO.COM
hbase/centos-08.hadoop.foo.com@HADOOP.FOO.COM
hdfs/centos-01.hadoop.foo.com@HADOOP.FOO.COM
hdfs/centos-02.hadoop.foo.com@HADOOP.FOO.COM
hdfs/centos-03.hadoop.foo.com@HADOOP.FOO.COM
hdfs/centos-08.hadoop.foo.com@HADOOP.FOO.COM
```

Figure 1. MIT KDC UPN and SPN Listings

Looking into greater detail at an SPN below, we see additional details on the SPN and the Keytab as follows:



```
kadmin.local:  getprinc yarn/centos-01.hadoop.foo.com@HADOOP.FOO.COM
Principal: yarn/centos-01.hadoop.foo.com@HADOOP.FOO.COM
Expiration date: [never]
Last password change: Fri Mar 24 13:29:39 EDT 2017
Password expiration date: [none]
Maximum ticket life: 1 day 00:00:00
Maximum renewable life: 5 days 00:00:00
Last modified: Fri Mar 24 13:29:39 EDT 2017 (cloudera-scm/admin@HADOOP.FOO.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 6
Key: vno 2, aes256-cts-hmac-sha1-96, no salt
Key: vno 2, aes128-cts-hmac-sha1-96, no salt
Key: vno 2, des3-cbc-sha1, no salt
Key: vno 2, arcfour-hmac, no salt
Key: vno 2, des-hmac-sha1, no salt
Key: vno 2, des-cbc-md5, no salt
MKey: vno 1
Attributes:
Policy: [none]
```

Figure 2. MIT KDC SPN details

This is a standard MIT-based SPN as created by Hadoop, and all SPN principals will be of this form: `service/host@REALM`. This MIT SPN is used by the local Yarn service account on the `centos-01.hadoop.foo.com` host where the keytab for this SPN has been copied.

Active Directory Kerberos Principals

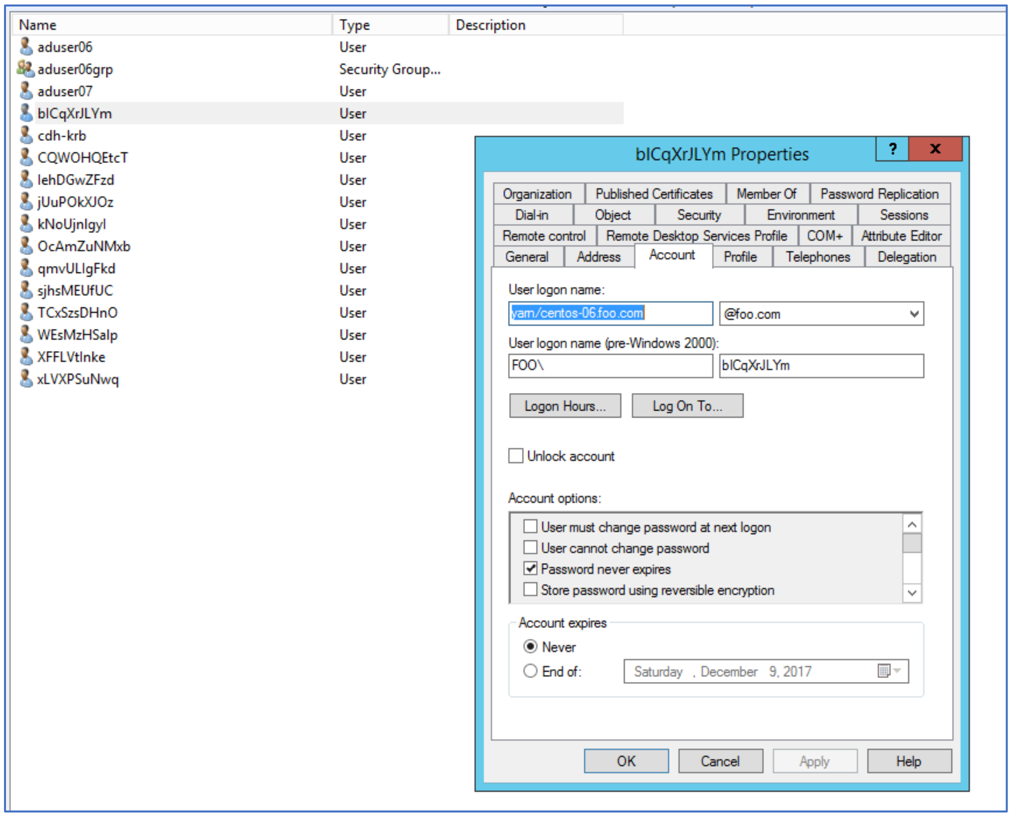
When we Kerberize the Hadoop cluster against an Active Directory the Principals initially can look slightly different at first but it is important to review what is going on here and why. It is these principals that are critical to understanding when it comes integrating Active Directory Kerberos with Hadoop and Isilon deployments.

We can see the following Cloudera-based deployment in Figure 5; the SPN for the Yarn service on a `centos-05.foo.com` host is as follows. What we need to understand here is that the:

- SPN is equal to: yarn/centos-05.foo.com@FOO.COM

  However…

- The name and pre-Windows2000 or sAMAccountName is set to a random text string, in the following example: bICqXrJLYm



Name	Type	Description
aduser06	User	
aduser06grp	Security Group...	
aduser07	User	
bICqXrJLYm	User	
cdh-krb	User	
CQWOHQEtcT	User	
lehDGwZFzd	User	
jUuPOkXJOz	User	
kNoUjnlgyl	User	
OcAmZuNMxb	User	
qmvULIgFkd	User	
sjhsMEUfUC	User	
TCxSzsDHnO	User	
WEsMzHSalp	User	
XFFLVtlnke	User	
xLVXPSuNwq	User	

bICqXrJLYm Properties

User logon name:
yarn/centos-06.foo.com @foo.com

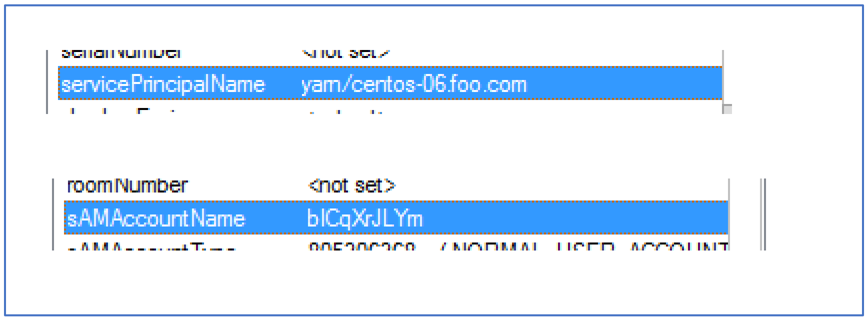
User logon name (pre-Windows 2000):
FOO\ bICqXrJLYm

☐ Unlock account

Account options:
☐ User must change password at next logon
☐ User cannot change password
☑ Password never expires
☐ Store password using reversible encryption

Account expires
● Never
○ End of: Saturday , December 9, 2017

Figure 3. Active Directory SPN Cloudera generated



servicePrincipalName	yarn/centos-06.foo.com
roomNumber	<not set>
sAMAccountName	bICqXrJLYm

Figure 4. Active Directory SPN details

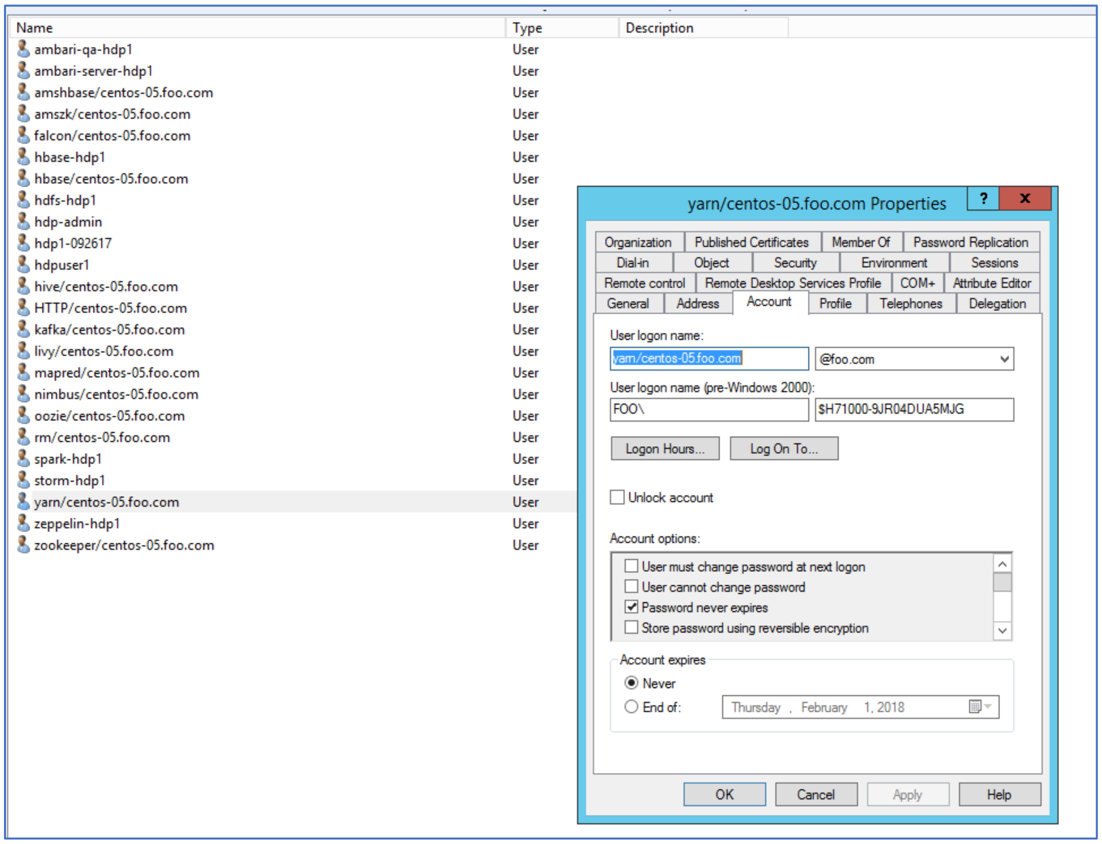
The reason this occurs is fairly straightforward: The SPN is unique, as it contains qualifiers to be created as a unique entity (hostname):

- Service Name: yarn
- Hostname: centos-05.foo.com
- Kerberos Realm: FOO.COM

But within Microsoft Active Directory, the pre-Windows2000 or sAMAccountName attribute is a required attribute and must be populated in order to ensure uniqueness. The Kerberization wizards automatically generate a random name ensuring compliance and uniqueness. Since this attribute is never used by the Hadoop service, this random string is an elegant solution.

As we will see later in this document, this attribute (sAMAccountName) is critical to Isilon OneFS and its multitenant integration.

The Principals look slightly different when Hortonworks generates the Kerberos Principals as follows:



Name	Type	Description
ambari-qa-hdp1	User	
ambari-server-hdp1	User	
amshbase/centos-05.foo.com	User	
amszk/centos-05.foo.com	User	
falcon/centos-05.foo.com	User	
hbase-hdp1	User	
hbase/centos-05.foo.com	User	
hdfs-hdp1	User	
hdp-admin	User	
hdp1-092617	User	
hdpuser1	User	
hive/centos-05.foo.com	User	
HTTP/centos-05.foo.com	User	
kafka/centos-05.foo.com	User	
livy/centos-05.foo.com	User	
mapred/centos-05.foo.com	User	
nimbus/centos-05.foo.com	User	
oozie/centos-05.foo.com	User	
rm/centos-05.foo.com	User	
spark-hdp1	User	
storm-hdp1	User	
yarn/centos-05.foo.com	User	
zeppelin-hdp1	User	
zookeeper/centos-05.foo.com	User	

yarn/centos-05.foo.com Properties

User logon name: yarn/centos-05.foo.com @foo.com

User logon name (pre-Windows 2000): FOO\ $H71000-9JR04DUA5MJG

☐ Unlock account

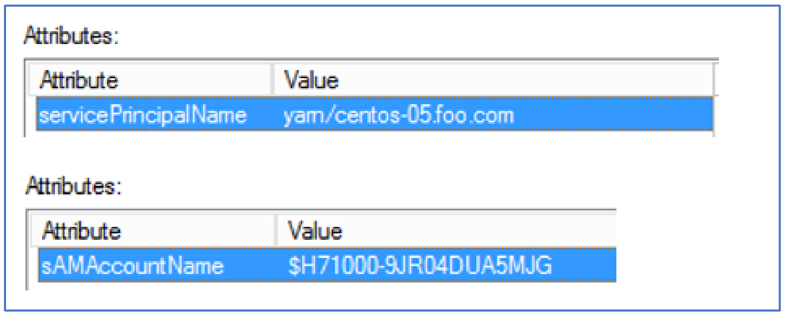
Account options:

- ☐ User must change password at next logon
- ☐ User cannot change password
- ☑ Password never expires
- ☐ Store password using reversible encryption

Account expires

- ● Never
- ○ End of: Thursday , February 1, 2018

Figure 5. Active Directory SPN Hortonworks generated



Attributes:

Attribute	Value
servicePrincipalName	yarn/centos-05.foo.com

Attributes:

Attribute	Value
sAMAccountName	$H71000-9JR04DUA5MJG

Figure 6. Active Directory SPN details

As you can see, the Hortonworks SPNs are slightly different in how they are represented, but the SPN and the sAMAccountName still exhibit the same structure and behavior.

Isilon SPNs

It is also important to review the required SPNs for the Isilon cluster when deploying Kerberos. Here again, we have some specific differences with the SPNs required, where they exist in the KDC, and how the different distributions manage them.

KDC versus Active Directory SPNs

Since Isilon is providing Kerberized access to the HDFS and WebHDFS services, it requires SPNs for these services only. Then, depending on the type of KDC used and the Hadoop cluster management tool in use, the following section outlines the requirements for the required SPNs:

KDC

- hdfs/smartconnectzonename – where *smartconnectzonename* is the FQDN that the Hadoop cluster is connecting with
- HTTP/smartconnectzonename – where *smartconnectzonename* is the FQDN that the Hadoop cluster is connecting with
- SPNs created in the KDC
- Need to be managed by Isilon OneFS to obtain the correct keytab to each cluster node; if the KDC is created by Ambari, it must be recreated by Isilon OneFS (see the Ambari section below)

Active Directory

- hdfs/clustername – where *clustername* is the FQDN of the machine account name that the cluster is joined to Active Directory as
- hdfs/smartconnectzonename – where *smartconnectzonename* is the FQDN that the Hadoop cluster is connecting with
- HTTP/smartconnectzonename – where *smartconnectzonename* is the FQDN that the Hadoop cluster is connecting with
- SPNs for Isilon OneFS should be created on the Isilon Cluster Machine object in Active Directory* and not in the managed OU*
- Some SPNs are auto-created by Isilon on AD join, but HDFS and HTTP may need to be manually created
- SPNs may be attempted to be created by the Kerberos Wizards. This can have additional consequences, for example duplicates** or the wrong location in AD

* See the Ambari section below

Cloudera Manager versus Ambari

- Cloudera Manager does not create Isilon SPNs
- *Ambari attempts to create the hdfs/smartconnectzonename and HTTP/smartconnectzonename
    - With AD, it will create these in the wrong places, for example in the Ambari-managed OU, not on the Isilon machine object
    - If they already exist in the KDC or AD, Kerberization will fail, as no duplicates can exist
    - If Ambari created the SPNs and keytab in KDC, Isilon does not have these keytabs – they must be deleted and recreated from the Isilon Kerberos provider

**For more information, refer to the Active Directory duplicate SPN issue in the duplicate-spns-with-isilon-ad-kerberos-and-hortonworks blog post.

For additional information on implementing Isilon OneFS with Hadoop distributions and KDCs, see the Isilon Hadoop Install guides.

Cloudera CDH and Hortonworks Ambari and HDP

It is also important to review the behaviors of the two primary Hadoop distributions that are commonly seen with Isilon and Kerberos. In general, the Kerberization requirements and process are very similar, but a few differences in the creation and behavior of the two distributions is worth highlighting, as again it has implications for OneFS integration and especially multitenancy.

Cloudera

The Cloudera Manager Kerberization wizards will create the following configuration in the KDC or Active Directory:

- Cloudera Manager only creates SPNs and Keytabs for the Service Accounts: For example, Yarn, Hive, Impala, and HBase
- No HDFS or HTTP SPN is created in the KDC. (Since no HDFS service is installed, these are not created)
- Name is a random string in Active Directory
- sAMAccountName is the same random string in Active Directory

Since Cloudera does not create any UPNs or the HDFS and HTTP SPNs in the KDC or Active Directory, no additional consideration is needed for Cloudera Kerberization.

An example of a service SPN is presented below. SPNs behave the same for both Cloudera and Hortonworks-based Kerberizations. The required SPNs for the Isilon cluster are created and managed by OneFS providing a valid Kerberos configuration for the Hadoop services.

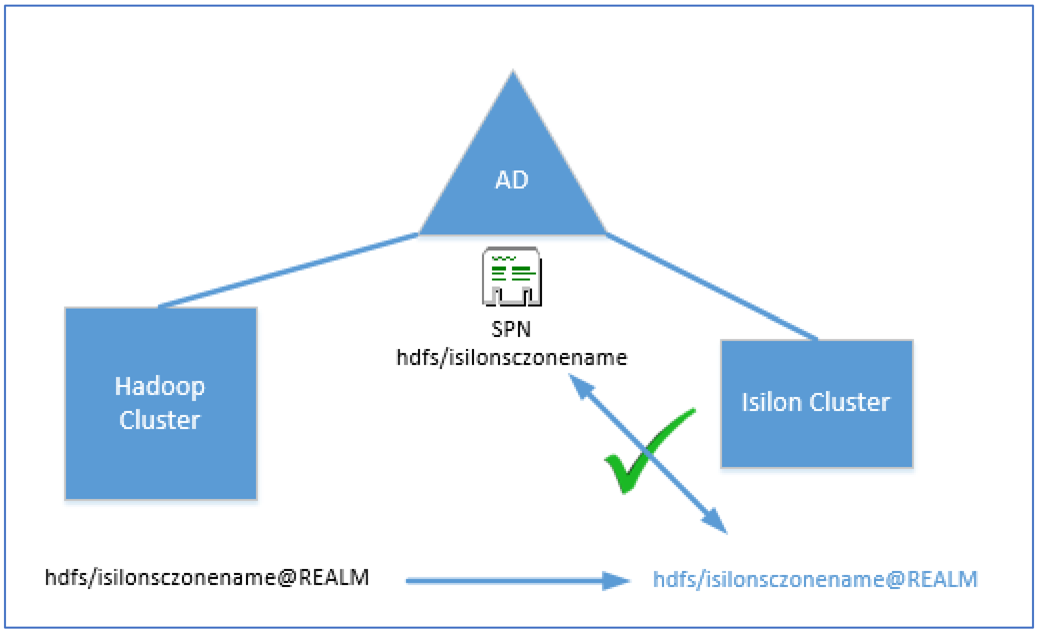


Figure 7. SPN usage with Isilon and Hadoop Cluster

Figure 7 illustrates how a Kerberized service is implemented within Active Directory. The Isilon cluster exposes a Kerberized HDFS service through a defined SPN of hdfs/isilonsczone name for the Kerberos realm of which it is a member (note that the HTTP SPN is not shown, but it is still required).

Hortonworks and Ambari

The Ambari Kerberization wizards will create the following configuration in the KDC or Active Directory:

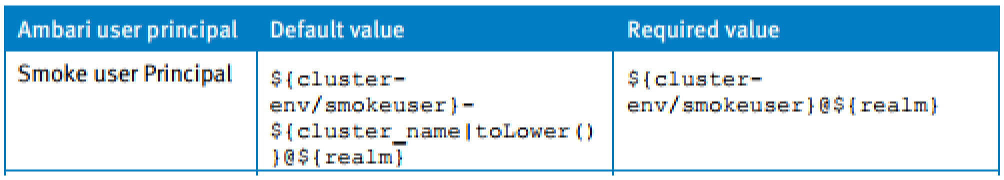
- Ambari creates SPNs for the Service Accounts and Keytabs for the Service Accounts: For example, Yarn, Hive, Impala, and HBase
- *HDFS and HTTP SPNs for the Isilon cluster are created either in the KDC or in the designated OU in Active Directory. These Ambari-generated SPNs are not correct and will need to be removed, as Isilon does not have the appropriate keytabs associated with them.
- **Ambari creates UPNs for a number of smoke test accounts: For example, ambari-qa, Spark, and HBase
- Name is the same as the account; By default, it appends –*clustername* to the end, however a basic Isilon Kerberos installation is recommended to remove this. Refer to the EMC Isilon OneFS with Hadoop and Hortonworks for Kerberos Installation Guide for further details.
- sAMAccountName is a random string

The critical points to be aware of here are the following:

1. *The SPNs created by Ambari: HDFS and HTTP are incorrect and need to be removed, allowing the Isilon cluster to create and manage its own SPNs for these services on Isilon OneFS. This is covered in the Isilon and Hortonworks Kerberization Guide and also addressed here in this blog post.

2. **The smoke test UPNs created need to be updated to support Isilon integration when Active Directory is used. This is detailed in the installation guide.

Looking at this configuration in detail assumes that you have followed the Isilon and Hortonworks Kerberos Installation Guide to deploy Kerberos. This document makes the following assumption: When configuring the Principals, we recommend that you remove the `cluster_name` and `cluster_name|toLower()` parameters from the Principal Suffix being generated. For example:



Ambari user principal	Default value	Required value
Smoke user Principal	`${cluster-env/smokeuser}-${cluster_name|toLower()}@${realm}`	`${cluster-env/smokeuser}@${realm}`

Figure 8. A Sample Ambari UPN change in the Kerberization Wizard

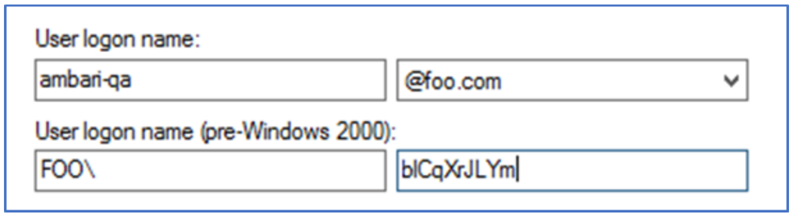
When the Ambari UPNs are created, this would ultimately lead to:

- `ambari-qa-hdp-clustername@REALM` being created

But by removing `cluster_name|toLower()`

- `ambari-qa@REALM` is created

However to ensure that the sAMAccountName requirement is fullfilled in Active Directory, a random string is generated for the created UPN as shown below:



User logon name: ambari-qa @foo.com

User logon name (pre-Windows 2000): FOO\ bICqXrJLYm

Figure 9. An Active Directory UPN created by the Ambari Kerberization Wizard

Note that this initially creates an issue for Isilon OneFS, as it looks up these Ambari-created users using the sAMAccountName. The Kerberized UPN connects to the cluster using the full UPN, but it is then deconstructed down to just the username. This username is then looked up, and it will fail in this case because of a sAMAccountName mismatch. This creates a lookup issue and ultimately access is denied. This issue is discussed in detail as part of a service troubleshooting blog post.

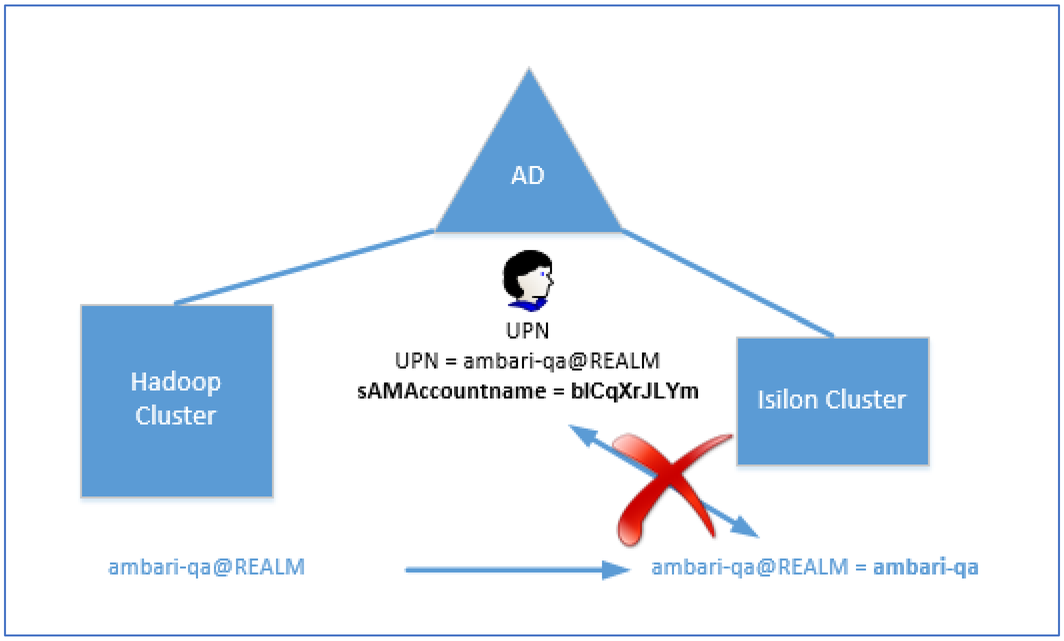
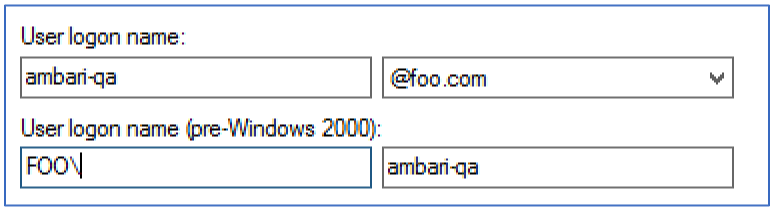


Figure 10. Mismatch exists between Isilon user and Ambari UPN sAMAccountName

If we modify the sAMAccountName to be equal to the UPN short name, OneFS can look up this user and no issues are observed.



User logon name:

ambari-qa	@foo.com

User logon name (pre-Windows 2000):

FOO\	ambari-qa

Figure 11. Modification required to an Ambari UPN

Having made a modification to the UPN in Active Directory, the lookup succeeds and the smoke test Ambari UPN user can interact with Isilon OneFS correctly as follows:

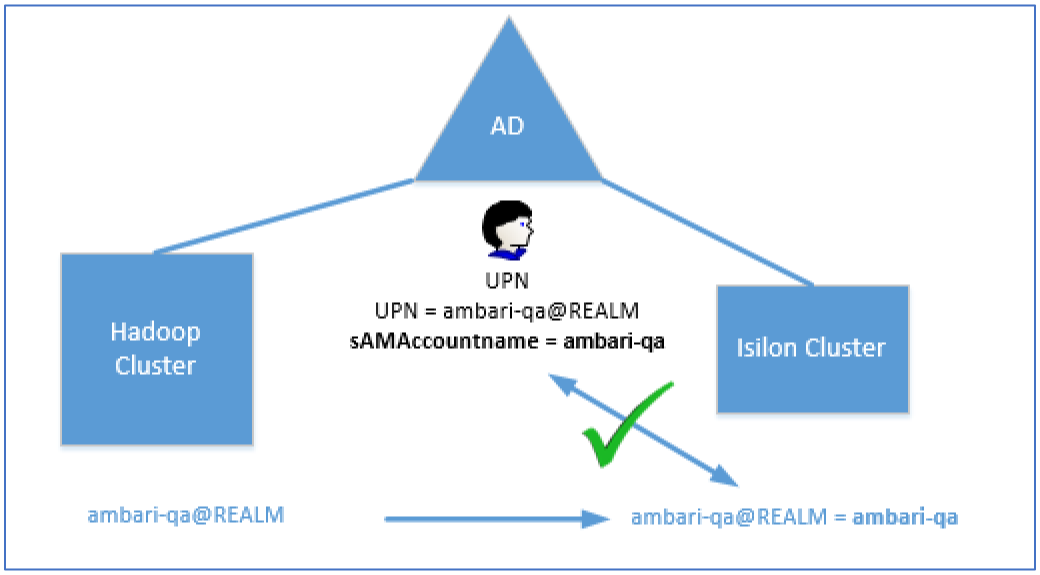


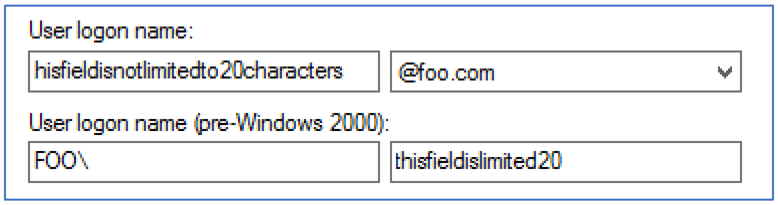
Figure 12. Successful lookup of an Ambari based UPN

This modification to the sAMAccountName is only required on Ambari-generated UPNs: For example, HDFS, ambari-qa, Spark, HBase, Storm, Zeppelin, and any other UPNs created by Ambari that are in use. No SPNs need any modification.

This modification is not required on Cloudera-based implementations, because no UPNs are created as we noted above.

Note:

The username or the UPN name attribute has a limit of 256 characters, but the sAMAccountName name (also known as the User logon name pre–Windows 2000) is limited to 20 characters in the Active Directory schema. (You cannot set the sAMAccountName name to longer than 20 chars using AD tools.) This should be taken into account when adding any suffixes like the clustername, as we will see later in this whitepaper.



User logon name:

hisfieldisnotlimitedto20characters	@foo.com

User logon name (pre-Windows 2000):

FOO\	thisfieldislimited20

Figure 13. Character limits with the Ambari UPNs sAMAccountName

Isilon Considerations

Isilon OneFS uses the concept of an Access Zone to create a data and authentication boundary within OneFS. Each Access Zone is associated with a data path root and can be attached to a number of different Authentication and Identity providers. As an example, in a simple cluster along with the default administrative system zone, a second Access Zone is defined and associated with an Active Directory provider. This provides separation of data and authentication providers at the simplest level to create a secure path to data in this zone.

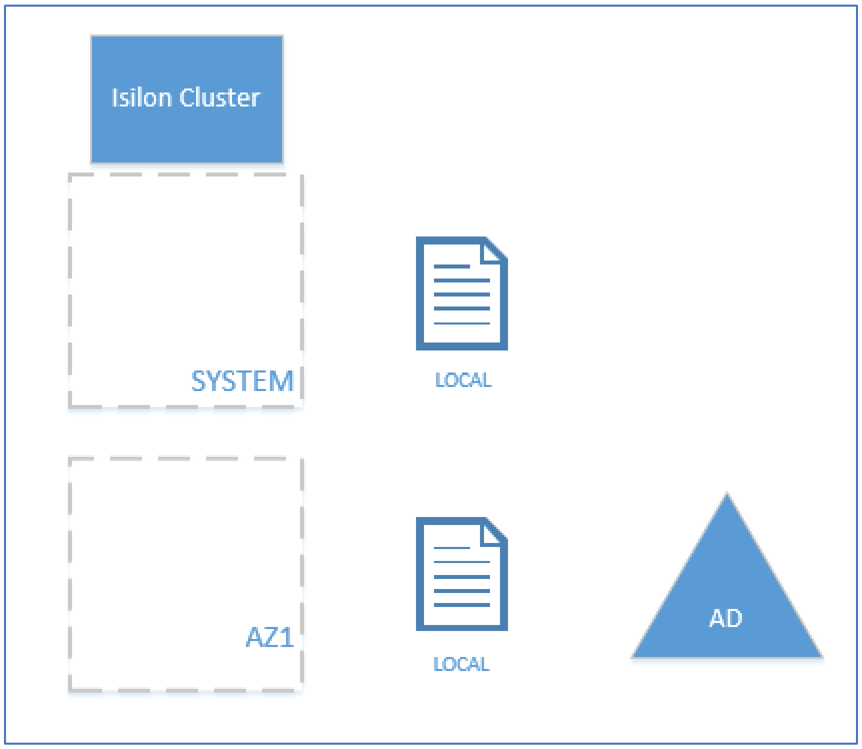


Figure 14. Single Zone plus System Zone

Additional Access Zones can be added to provide separation of data and to associate different authentication providers with these data sets. This is the foundation of multitenant architecture.

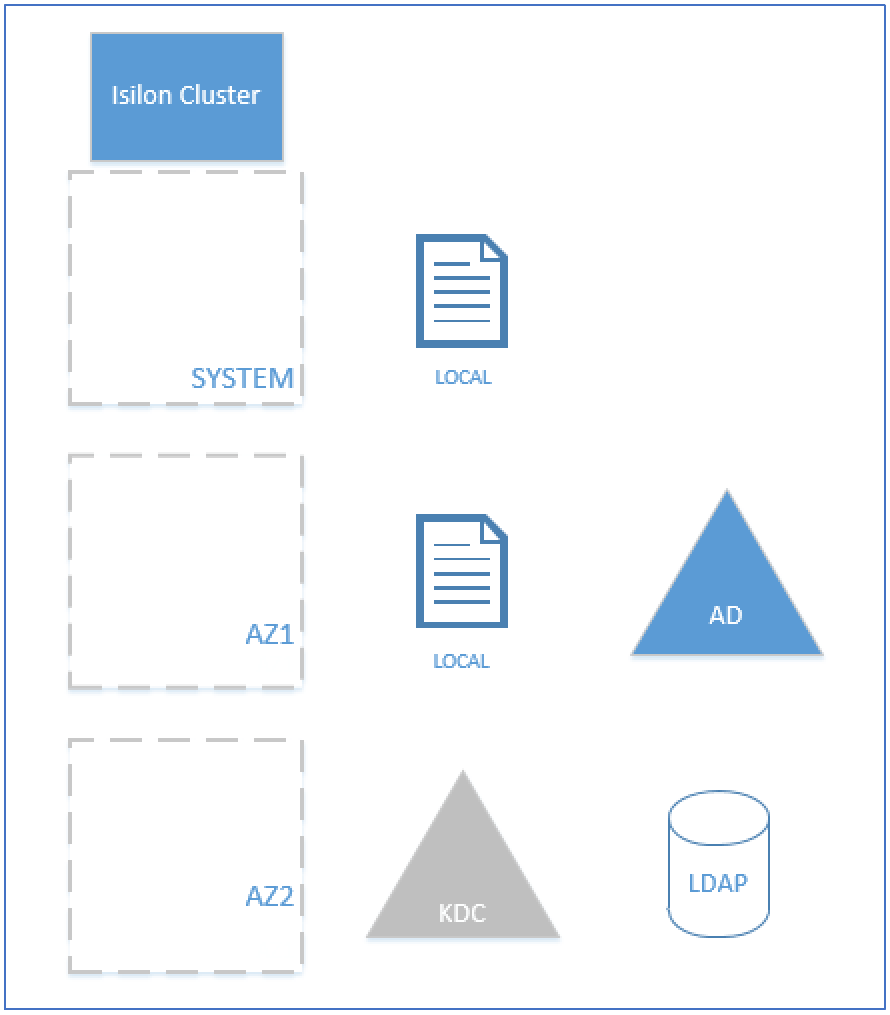


Figure 15. Multi-Access Zones

This capability gives us the ability to create multiple HDFS roots within a OneFS cluster and deploy multiple Hadoop clusters against an Isilon cluster. When deploying multiple Hadoop clusters against an Isilon cluster, it is critical to plan ahead and understand the deployment requirements for setting up and securing these Hadoop clusters prior to deployment of the Hadoop clusters.

Isilon supports many different forms of access management through authentication and identity management depending on how a specific Access Zone is configured. Access Zones within OneFS provide a separation of data and an authentication boundary. Similar to Hadoop clusters, each zone within an Isilon cluster will need matching service and user accounts to provide access control, valid authentication, and data access.

Isilon OneFS supports the ability for:

- Local Accounts
- Kerberos Authentication: KDC – MIT KDC
- Identity Management with Directory Services: LDAP and NIS
- Integrated KDC + LDAP with Microsoft Active Directory

Isilon identity management is a large subject, and a solid understanding of how it works and integrates alongside authentication providers and identity providers should be fully investigated before integrating Isilon into any Hadoop infrastructure. The following series of posts is a useful primer to Isilon Authentication and Access Management behaviors and will provide additional background to compliment the following whitepaper: EMC Isilon Multiprotocol Concepts Series.

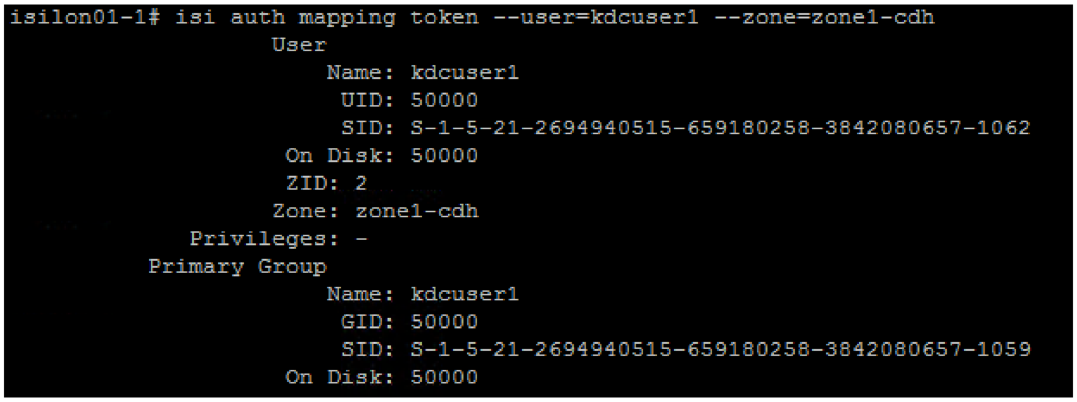
Isilon Identity Requirements

One extremely important consideration to be aware of with Isilon identity management is how Isilon stores identities on-disk and where those identities come from. A primary design tenant of OneFS identity management is that every user or group known to Isilon is composed of a SID and a POSIX ID (a UID or GID). The origins of these identities is based on where the identity is created and managed: Is the user a local user, an Active Directory user or group, an LDAP user, and so on. We can classify these identities as 'real' or 'generated' at the highest level. Since a user or group must always have a SID and a POSIX identifier, each identifier can be composed of different parts, depending on where it came from and what it constitutes. The core concept is that when an identity is looked up, if it contains or is mapped to other identities, and it can get the required SID component and the posix ID, then the identity meets the requirement and can be used as is Whereas, if only one of the identities is found, the Isilon cluster assigns and tracks an internally assigned identity to meet the requirement, hence the term 'generated' identity.

These identities are critical to a user or group and dictate the on-disk identity used for a file permission or ownership. Even though each identity always contains both, only one is stored as the identity on disk. Which one is used is dependent on the nature of both IDs. The primary goal is to use an authoritative ID as the on-disk identify, so that both of the IDs are evaluated, and OneFS selects the appropriate ID as the on-disk identity. When multiple identity providers are in use, a user could be composed of internal identities or external identities. It is critical to maintain consistency in this case and get the correct ID on-disk.

Since many different identity scenarios can exist, we can illustrate a few.

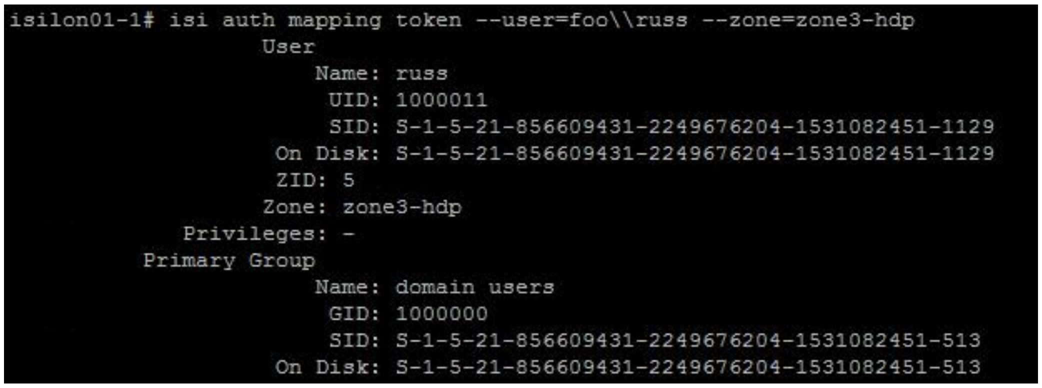
When a local user account is created; both the SID and POSIX UID are internally assigned and managed IDs. In this situation, the user's POSIX UID is used as the on-disk identity, as seen below.



```
isilon01-1# isi auth mapping token --user=kdcuser1 --zone=zone1-cdh
                   User
                       Name: kdcuser1
                        UID: 50000
                        SID: S-1-5-21-2694940515-659180258-3842080657-1062
                    On Disk: 50000
                    ZID: 2
                   Zone: zone1-cdh
             Privileges: -
          Primary Group
                       Name: kdcuser1
                        GID: 50000
                        SID: S-1-5-21-2694940515-659180258-3842080657-1059
                    On Disk: 50000
```

Figure 16. Local users with an internal UID and SID: UID on-disk

When looking at an Active Directory user account, we see the SID is assigned in the AD. However, to meet the OneFS requirement, an internal UID is assigned. In this case, since the SID is more authoritative, it is used as the on-disk ID.



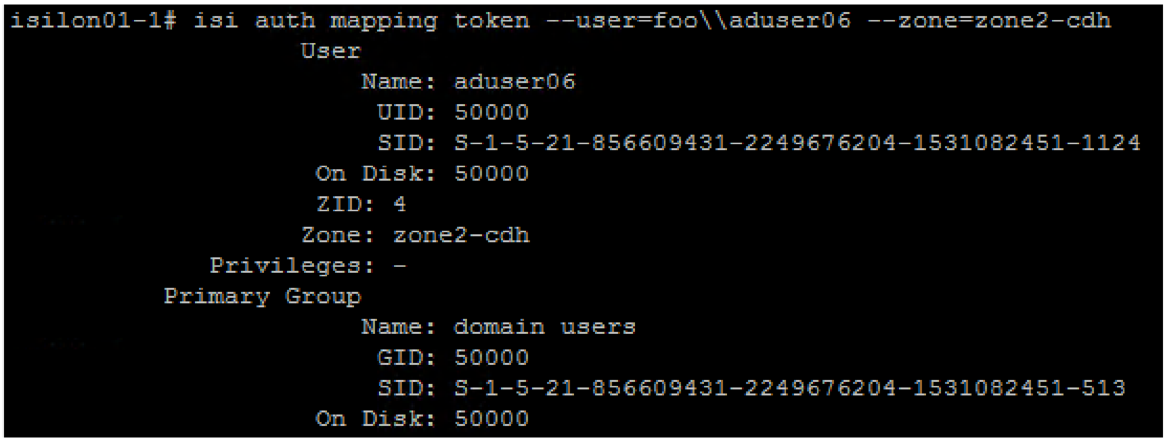
```
isilon01-1# isi auth mapping token --user=foo\\russ --zone=zone3-hdp
                   User
                       Name: russ
                        UID: 1000011
                        SID: S-1-5-21-856609431-2249676204-1531082451-1129
                    On Disk: S-1-5-21-856609431-2249676204-1531082451-1129
                    ZID: 5
                   Zone: zone3-hdp
             Privileges: -
          Primary Group
                       Name: domain users
                        GID: 1000000
                        SID: S-1-5-21-856609431-2249676204-1531082451-513
                    On Disk: S-1-5-21-856609431-2249676204-1531082451-513
```

Figure 17. Active Directory user with an AD SID and an internal UID: SID on-disk

UNIX Attribute Enable Active Directory IDs

It is possible to use the Active Directory schema to also assign UIDs and GIDs to AD-based users and groups. This means that when a user is looked up in the AD, we get an assigned SID and POSIX identity, thus meeting the OneFS requirement with no need to generate an internal one. In this situation, both IDs are from an external source and valid, but we still only select one of them for the on-disk ID; in this case, it's the UID, as illustrated below.

By enabling RFC2307 connectivity on the Active Directory Provider on OneFS, we enable AD to provide both SID and UID, meeting the requirements of OneFS without internal assignment of either identity. This is very useful when using Hadoop clusters, as typically most user and accounts are based on POSIX identities, and most Hadoop clusters use POSIX IDs and POSIX-based permissions on HDFS data. When using Active Directory for Hadoop users and IDs, we strongly recommend that you use the RFC2307 capabilities to maintain a consistent access model. Anytime the POSIX attributes in the Active Directory schema are used, you should consult the following knowledgebase article and implement the required changes: https://support.emc.com/kb/335338



```
isilon01-1# isi auth mapping token --user=foo\\aduser06 --zone=zone2-cdh
                   User
                       Name: aduser06
                        UID: 50000
                        SID: S-1-5-21-856609431-2249676204-1531082451-1124
                    On Disk: 50000
                    ZID: 4
                   Zone: zone2-cdh
             Privileges: -
          Primary Group
                       Name: domain users
                        GID: 50000
                        SID: S-1-5-21-856609431-2249676204-1531082451-513
                    On Disk: 50000
```

Figure 18. Active Directory user with an AD SID and an AD UID: AD+RFC2307 UID is used on-disk

Having reviewed the fundamental requirements needed to deploy authentication and access management with Hadoop and OneFS, we now can review potential deployment scenarios.

Deployment Methodologies

As we look at the different models for deploying Hadoop and Isilon clusters from a user and account perspective, we have to recognize the following design tenants:

- The location of the identity of the account; Hadoop local, Isilon Local or Directory-based
- Is Simple or Kerberos authentication in use
- Any specific consideration in regards to that configuration

The actual implementation and configuration of specific approaches are beyond the scope of this document, however you can find individual configuration documents on the Isilon Hadoop Info Hub to support them. It is very common to deploy with local accounts and simple authentication to get the clusters operational and then deploy a Kerberos configuration and central user directory solution, as this simplifies the initial installation. Let's review at a high level how this approach is executed:

All Authentication and Identity Management approaches should be determined prior to the installation where possible to avoid issue post-Hadoop deployment.

1. Create an Isilon HDFS Access Zone and create all Hadoop Local service accounts per the Isilon Hadoop Installation guides
2. Create all the Hadoop service accounts on all hosts to be deployed in the Hadoop cluster (UID/GID parity per the Isilon Hadoop Installation guides)
3. Deploy the Hadoop cluster
4. Test and validate the Hadoop cluster with the silon Integration, using local accounts
5. Kerberize the Hadoop cluster per the Isilon Hadoop Kerberos Installation guides
6. Kerberize Isilon per the Isilon Hadoop Kerberos Installation guides
7. Implement Identity Management

8. Test and validate the Hadoop cluster with the Isilon Integration, using Directory Service-based user accounts*

* Note, the service accounts are still local IDs, while the user accounts are based on a Directory Service User IDs, but all accounts are secured with Kerberos.

Overview of Deployment Diagrams

The design pattern we will discuss will illustrate the different options available for using different authentication and identity management providers. The following table illustrates the entities we will use to describe possible deployment methodologies:

USER ID	A local USER account has an associated account and identity, runs jobs and tasks
SVS ID	A local SERVICE account; HDFS, Yarn, Mapred HBase
ID	An ID in a Directory Service – LDAP or Active Directory
UPN	User Principal Name in the KDC
SPN	Service Principal Name in the KDC
KDC	KDC provides Kerberos Principals: SPNs and UPNs
AD OU	Active Directory, Provides a Directory-based account principals; SPNs and UPNs – Kerberos-based with an LDAP Identity Provider
LDAP	LDAP Server, Provides account identities

Figure 19. Symbols used in the deployment methodologies

Hadoop Cluster Deployments

Let's review the basic methods of implementing standalone Hadoop clusters, where the accounts and principals are created, and how they operate to better understand the more complex integration models coming up.

1. Simple Standalone Cluster – Local Service Accounts, Local User Accounts

The simplest deployment model is using local service and user accounts for all Hadoop operations and functions. This model contains no authentication of data access control with per host user accounts. Local users are created and added to hosts for job submission, while all the services run with service accounts created at cluster install time. This is the simplest Hadoop cluster deployment available.

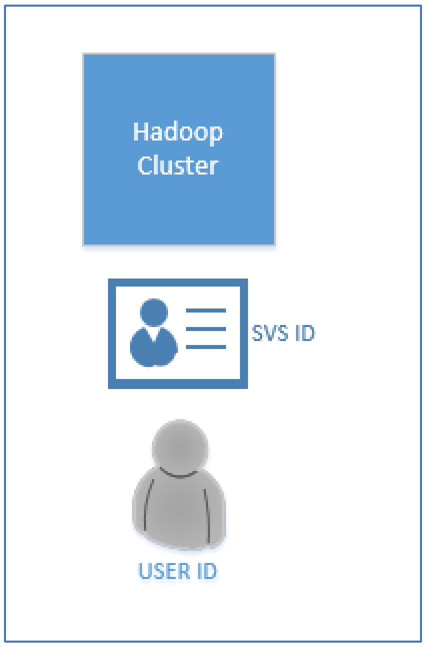


Figure 20. Simple – Local Accounts

Account Type	Identity	KDC Authenticated
Hadoop Service Account	Local ID	No
User Accounts	Local ID	No

- Simple to deploy and manage; Service and User Accounts and Identities are per host
- No security
- Likely the initially deployed state prior to Kerberization

2. Simple Cluster – Local Accounts with Kerberos Authentication

Adding Kerberos to the cluster enforces authentication of users submitting jobs to remove the ability of users to impersonate any user within the cluster. Service account SPNs with associated keytabs are created and distributed, while user UPNs are secured with a password known only to that specific user. In these simple initial deployments, only Kerberos authentication is in use; identity management is still based on local identities.

Cloudera and Hortonworks Differences

As we introduce Kerberos into the Hadoop and Isilon environment, we need to briefly recap a minor difference between the two distributions and how the built-in cluster Kerberos wizards create different principals in the KDC as we discussed earlier.

3. Cloudera CDH - Kerberos

Cloudera's CDH Kerberization wizard creates only the Service Account SPNs in the KDC.

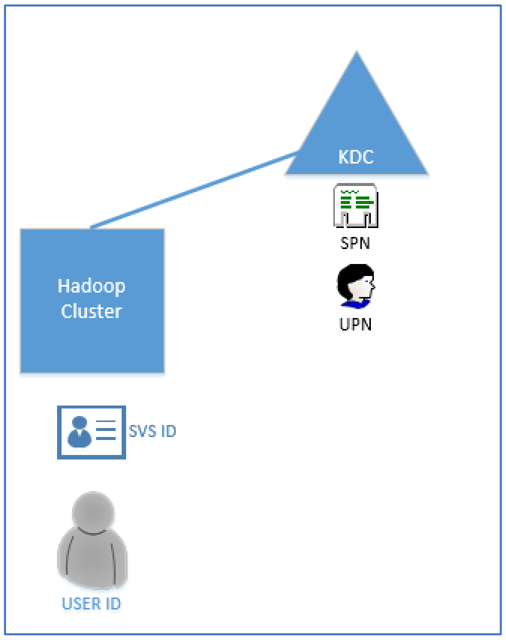


Figure 21. Secure – Local Accounts with Kerberos Authentication – Cloudera CDH

Account Type	Identity	KDC Authenticated
Hadoop Service Account	Local ID	Yes - SPN
User Accounts	Local ID	Yes – UPN

- Simple to deploy and manage
- Cluster secured with Kerberos
- Identity management is still based on local accounts
- Hadoop services leverage SPNs
- All users submitting jobs will require a UPN and password in the KDC

4. Hortonworks HDP and Ambari - Kerberos

The Ambari Kerberization wizard within Hortonworks creates all the required service account SPNs, but it also creates a number of Ambari user account UPNs in the KDC as shown:

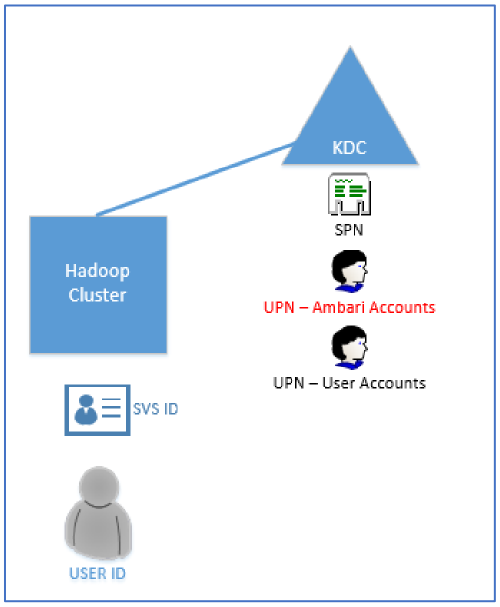


Figure 22. Secure – Local Accounts with Kerberos Authentication – Ambari Hortonworks HDP

Account Type	Identity	KDC Authenticated
Hadoop Service Account	Local ID	Yes – SPN, Ambari-UPN
User Accounts	Local ID	Yes – UPN

- Simple to deploy and manage
- Cluster secured with Kerberos
- Identity management is still based on local accounts
- Hadoop services leverage SPNs
- Ambari creates a number of UPNs in the KDC for smoke tests and service checks
- All users submitting jobs will require a UPN and password in the KDC

From this point in the document moving forward, we will generalize Cloudera and Ambari-based deployments from an SPN and UPN perspective to simplify the deployment models, any deviation will be called out as needed.

5. Integrated Cluster – Local Service Accounts, Directory Service User Accounts and Kerberos Authentication

In this model, service accounts are still local accounts created at cluster installation with corresponding KDC principals created upon Kerberization. However, normal users who run jobs are Directory Service-based accounts residing in an LDAP or Active Directory Server. The setup and integration of these accounts to the local hosts is not covered here. Many methods exist on how to do this; SSSD would be one approach as an example.

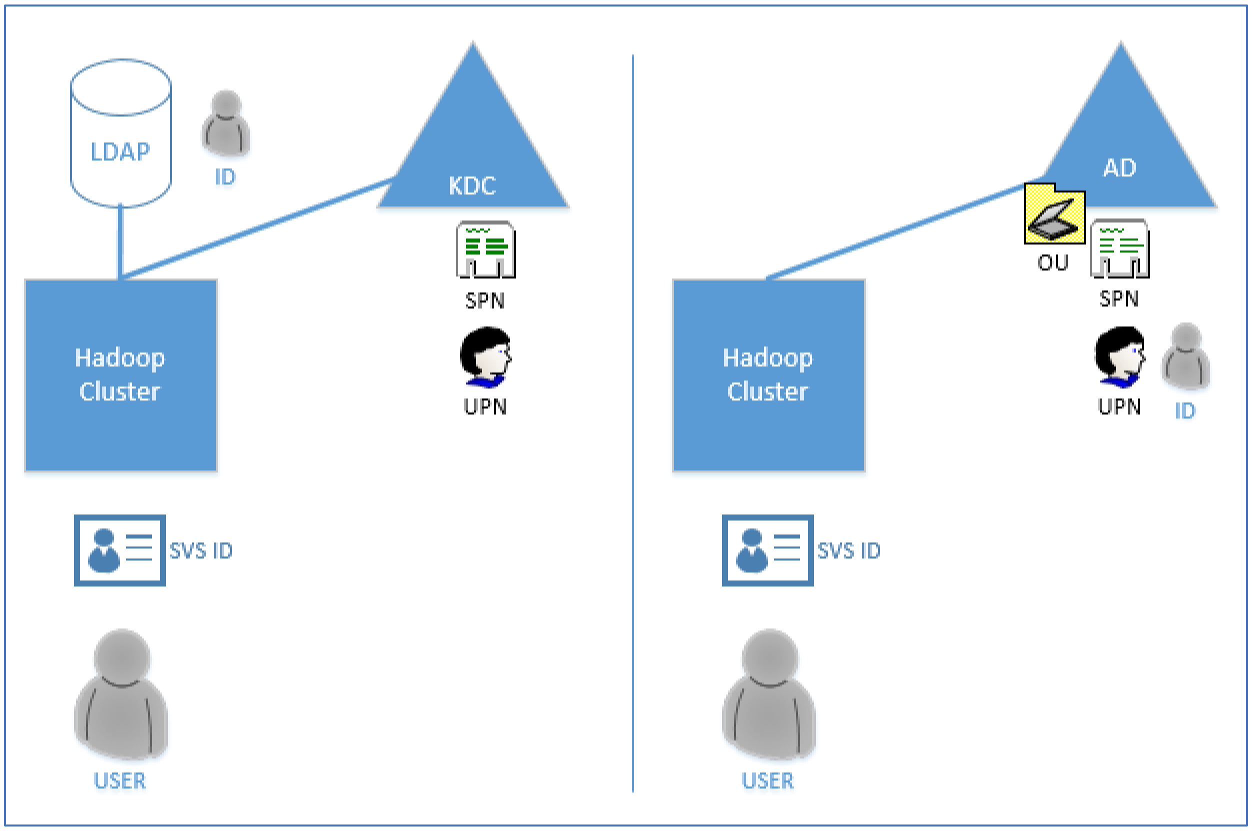


Figure 23. Secure – Directory-based Accounts with Kerberos Authentication

Account Type	Identity	KDC Authenticated
Hadoop Service Accounts	Local ID	Yes – SPN, Ambari-UPN
User Accounts	LDAP or AD	Yes – UPN

- Simple to deploy and manage
- Cluster secured with Kerberos
- Service Accounts are local identities
- Hadoop services leverage SPNs
- Ambari creates a number of UPNs in the KDC
- All users submitting jobs are based on a UPN and password in the KDC; user identity is based on Directory Service Account

These initial deployment models illustrate the potential location of identities and Kerberos principals when deploying standalone Hadoop clusters and the options available. As Isilon is brought into the cluster, similar deployment methodologies exist for the management and deployment of these accounts to operate an integrated environment.

Isilon – Hadoop Cluster Integration Deployments

Ultimately, it is the combination of authentication through Kerberos and authorization through Identity Management that permits a fully secured Hadoop and Isilon cluster to be deployed and operated. The following sections in this whitepaper will review the potential deployment models available for authentication and identity management when deploying Isilon OneFS with Hadoop clusters. We will look to highlight common patterns, best practices, and additional configuration information required to deploy different models.

1. Simple Authentication Hadoop and Isilon Integration

Let's review simple authentication and identity management before Kerberos and LDAP services are integrated. In this model, no authentication is occurring, and identity management is entirely dependent on local identities.

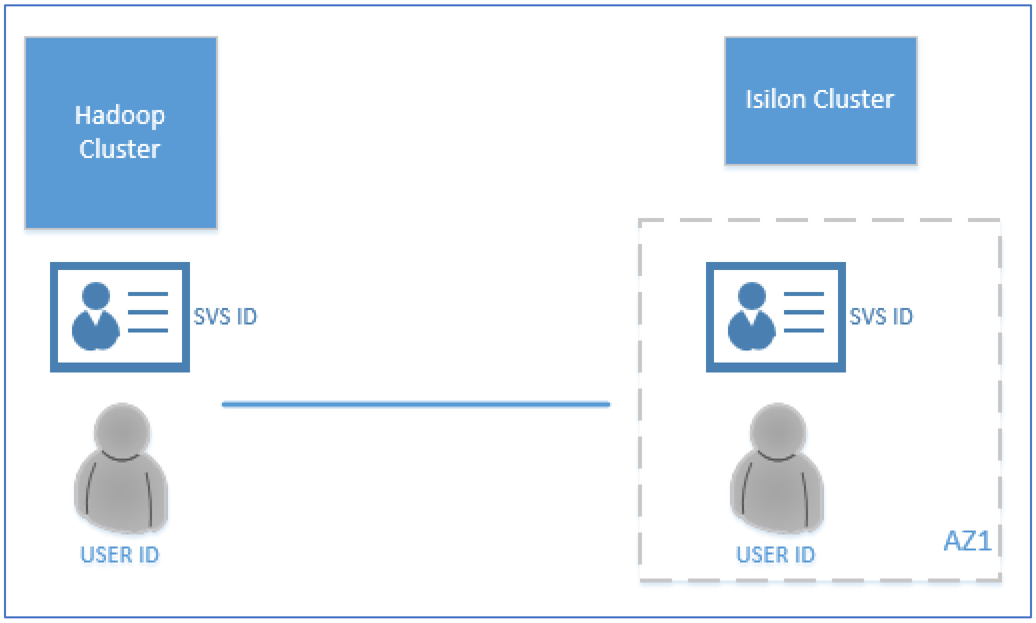


Figure 24. Isilon – Hadoop Cluster Simple Authentication Integration

Account Type	Identity	KDC Authenticated	Isilon Providers Used
Hadoop Service Accounts	Local ID	No	-
User Accounts	Local ID	No	-
Service Accounts - Isilon	Local Isilon ID	No	Local Provider Isilon
Users Account - Isilon	Local Isilon ID	No	Local Provider Isilon

- Simple to deploy and manage; Service and User Accounts and Identities are per host
- No security
- Likely the initially deployed state prior to Kerberization
- Ensure UID/GID parity
- Per Access Zone Isilon Local Service and User accounts

2. Local Accounts with Kerberos Authentication Hadoop and Isilon Integration

Introducing Kerberos into the Hadoop – Isilon integration creates a secure cluster in which all users are required to authenticate before executing jobs.

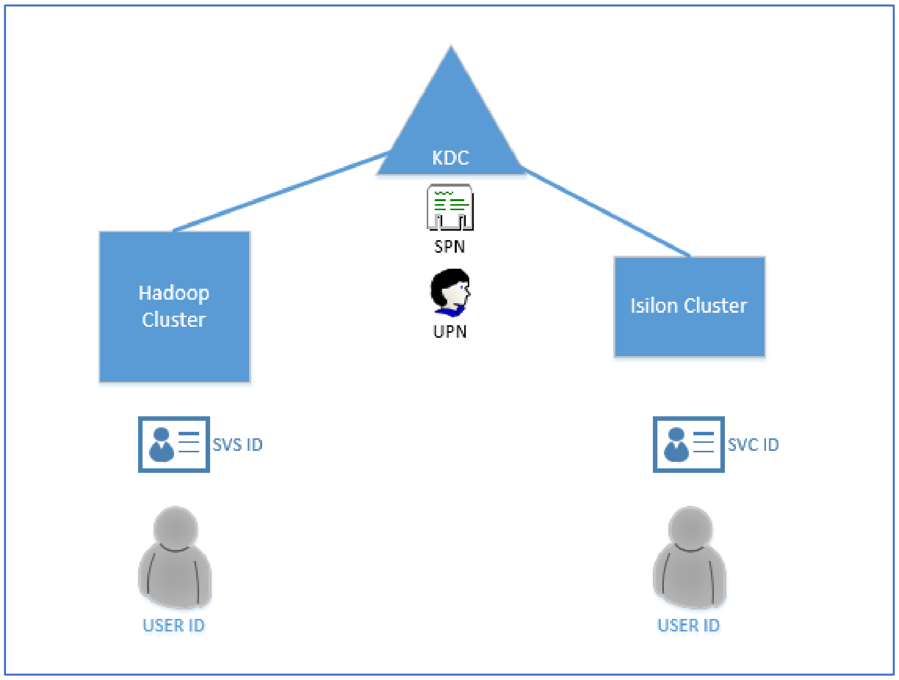


Figure 25. Isilon – Hadoop Cluster KDC Based Authentication Integration

Account Type	Identity	KDC Authenticated	Isilon Providers Used
Hadoop Service Accounts	Local ID	Yes	-
User Accounts	Local ID	Yes	-
Service Accounts - Isilon	Local Isilon ID	Yes	Local Provider Isilon Kerberos Provider
Isilon Users Account - Isilon	Local Isilon ID	Yes	Local Provider Isilon Kerberos Provider

- Simple to deploy and manage; likely cluster was deployed with Simple Authentication and Kerberos added
- Cluster secured with Kerberos
- Identity management is still based on local accounts
- Hadoop services leverage SPNs
- Ambari creates a number of UPNs in the KDC for smoke tests and service checks
- All users submitting jobs will require a Local Identity with an associated UPN and password in the KDC

In order to complete the Kerberization, the following SPNs are needed for Isilon:

- hdfs/smartconnectname
- HTTP/smartconnectname

3. Local Service Accounts with Kerberos Authentication and Directory User Accounts

A very common deployment is to leverage Active Directory as the Kerberos provider to the Hadoop and Isilon clusters. In this deployment scenario, we continue to leverage local accounts for all Service account identities. User identities are managed in the central AD. This model simplifies User management and local accounts on Local hosts and Isilon do not need to be created and managed. This model assumes Hadoop hosts are integrated into Active Directory to provide this functionality. A common method of achieving this is through SSSD integration, but many others exist.

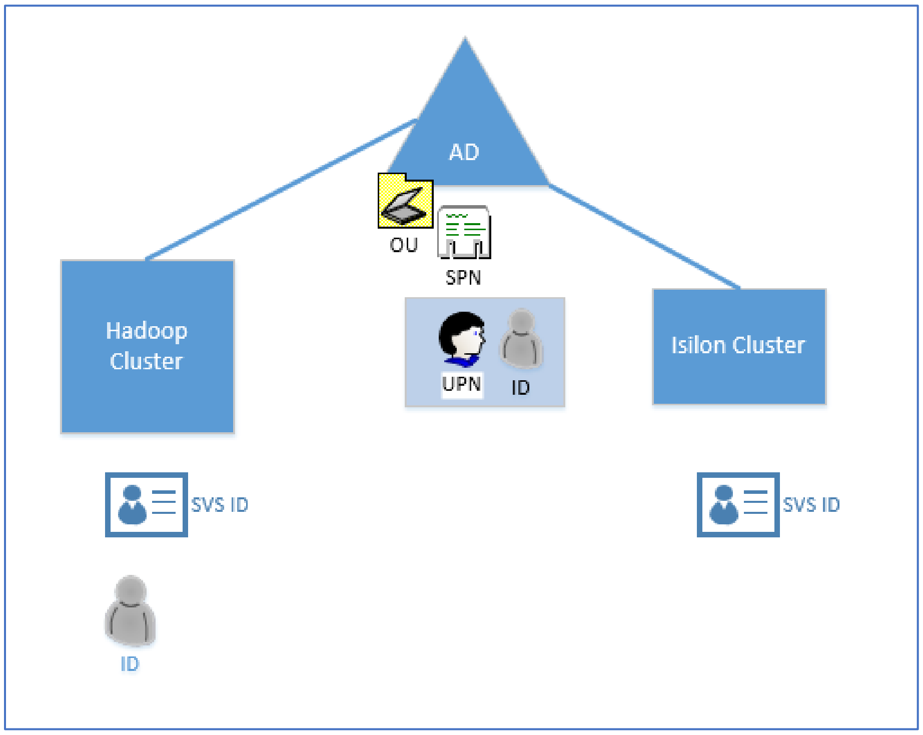


Figure 26. Isilon – Hadoop Cluster Active Directory Based Authentication Integration

Account Type	Identity	KDC Authenticated	Isilon Providers Used
Hadoop Service Accounts	Local ID	Yes	-
User Accounts	Local ID	Yes	-
Isilon Service Accounts	Local Isilon ID	Yes	Local Provider Isilon
Users Account	AD	Yes	Active Directory Provider

In order to complete the Kerberization, the following SPNs are needed for Isilon:

- hdfs/clustername
- hdfs/smartconnectname
- HTTP/smartconnectname

It is possible to use a Directory Service LDAP or AD for all Accounts (service and users) with a Hadoop deployment, but the configuration and integration of using non-local service accounts will require modification to the Hadoop cluster installation per the vendor's installation guide. Isilon would also need to be configured to leverage these Directory Service accounts.

Isilon – Hadoop Kerberos Trust-based Deployments

An extremely useful capability of Kerberos authentication is the ability to leverage trusts, where one Kerberos realm can be enabled to trust principals from another realm. This architecture can be very useful for delegation of administrative management, where individual realms can be managed by different administrators while maintaining a consistent security model across the environment. Kerberos Trusts are standard to most Kerberos distributions, and implementations are easy to deploy. The management of trusts is not outlined here, but all distributions provide thorough documentation. Some additional configuration will be required on the Hadoop cluster and on the Isilon cluster to support Kerberos trusts, but this is well documented as well.

Many forms of trusts exist and can be implemented. When looking at trusts, it is important to recognize that Kerberos will handle the underlying authentication through the trust relationship, but we still require identity management. Many different methods of implementing identity management exist for integration into a trusted Kerberos configuration to meet this requirement. Using the different approaches and providers available to OneFS and the Hadoop cluster (for example, local, LDAP, AD, and so on), we can use trusts to support different realms with a consistent identity model in place.

1. Local Hadoop Dedicated KDC with a Trusted Active Directory for Users

In this deployment, we leverage a local KDC for Kerberization of the Hadoop and Isilon Principals, with a trusted Active Directory Realm for User Principals. The trust can be either a one-way or two-way trust from the local KDC to the Active Directory as follows:

- Local KDC for all Hadoop and Isilon SPNs, Ambari UPNs
- Hadoop Service accounts are local to Hadoop hosts and Isilon
- User UPNs and identities exist in Active Directory; users have POSIX attributed in schema
- Users authenticate to the Hadoop cluster through the Kerberos Trust

The advantage of this model is that all the Hadoop and Isilon cluster Kerberos management is on a local KDC managed by the Hadoop and Isilon Administrators without having to make administrative modifications to the central authentication infrastructure. It isolates the Kerberos configuration—allowing for simpler management—and provides potential flexibility in deployment.

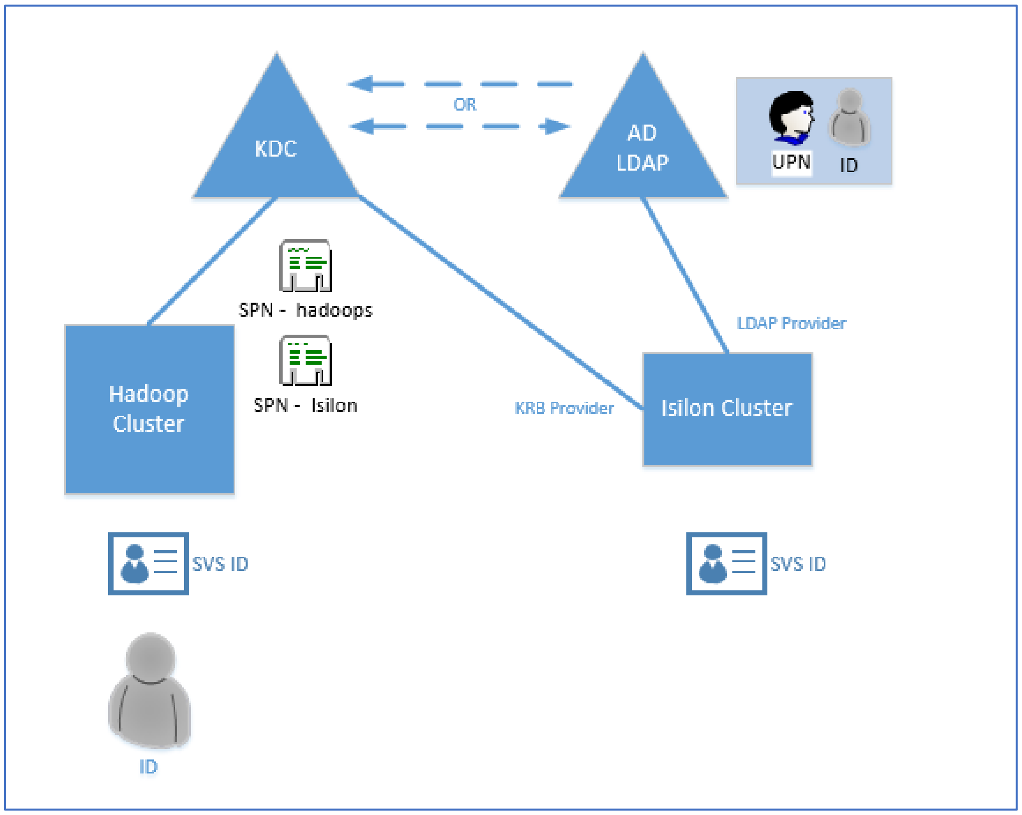


Figure 27. Isilon – Hadoop Cluster Local KDC with a Trusted Active Directory Based Authentication Integration

Account Type	Identity	KDC Authenticated	Isilon Providers Used
Hadoop Service Accounts	Local ID	Yes	-
User Accounts	Local ID	Yes	-
Isilon Service Accounts	Local Isilon ID	Yes	Local Provider Isilon Kerberos Provider*
Users Account	Active Directory	Yes-via Trust	LDAP Provider**

*Isilon is Kerberized against the Kerberos Provider, not Active Directory

**Isilon is connected to Active Directory as an LDAP provider only. Since Kerberos is managed through the Kerberos provider, the zone cannot contain an Active Directory provider also. Instead, we leverage the LDAP component of the AD to provide the identity of the AD user to Isilon only; all service accounts continue to be local. In order to leverage an Isilon LDAP provider against an Active Directory, some additional modifications to the OneFS LDAP provider must be made. See the Appendix for details.

- Local KDC for all Hadoop SPNs and Ambari UPNs
- Cluster secured with Kerberos
- Service Account Identity management is still based on local accounts
- Hadoop services leverage Local KDC SPNs
- Ambari creates a number of UPNs in the KDC for smoke tests, service checks
- All Hadoop management and Kerberos integration can be done on an isolated local KDC
- User Principals can be maintained and managed in a central corporate directory, use AD as an LDAP-only provider from Isilon

In order to complete the Kerberization, the following SPNs are needed for Isilon:

- hdfs/smartconnectname
- HTTP/smartconnectname

Since Isilon is Kerberized against the KDC we do not need a cluster name SPN for HDFS.

Many types of Kerberos trust exist and can be implemented. If the requirements for the trust can be met, then authentication across realms can succeed. Having completed successful authentication, it is then critical to implement an identity model that supports these cross-realm Principals successfully.

Isilon – DAS Hadoop Tiered Storage Integration Deployments

Hadoop Tiered Storage (HTS) is the ability to integrate a full independent DAS cluster with an Isilon cluster to create two tiers of storage that are fully accessible to the Hadoop cluster. Data can reside on the DAS Hot tier or the Isilon cold tier with seamless access to the data between the two storage tiers. For additional information, see the following Hadoop Tiered Storage with Dell EMC Isilon and Dell EMC ECS Clusters white paper: https://community.emc.com/docs/DOC-61992

Ultimately, for HTS to function correctly, any account wishing to read or write data to Isilon OneFS must have access and file permissions on the data. Like all deployments, a number of different configurations can exist to meet these requirements with different approaches and levels of security and integration as follows:

1. Simple Authentication

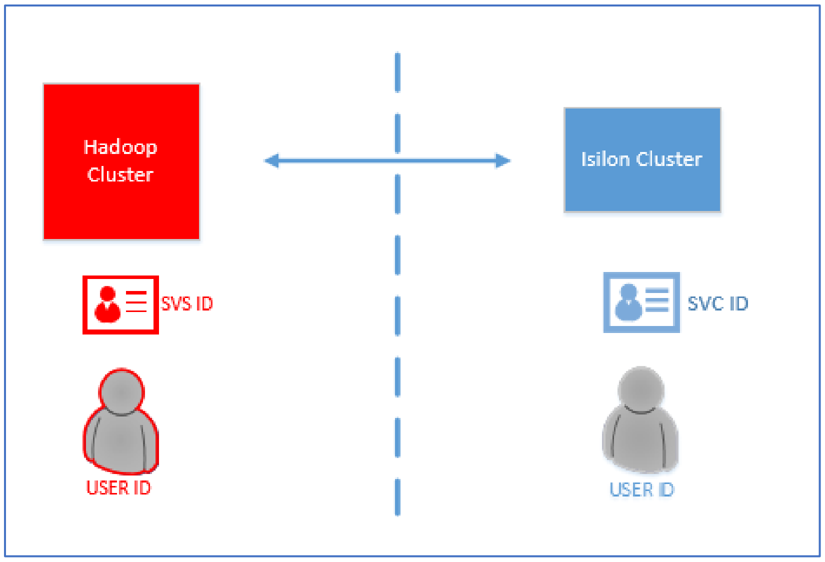


Figure 28. Simple Authentication HTS

Account Type	Hadoop Identity	KDC Authenticated	Isilon Identity
Hadoop Service Account	Local ID	No	Local ID
User Accounts	Local ID	No	Local ID

This is the simplest form of HTS. The Hadoop and Isilon clusters are considered entirely separate entities with no Kerberos security in place. In order for the Hadoop user to access the HTS Isilon namespace, a valid username and permissions need to exist on the target Isilon OneFS Access Zone and path.

2. Kerberized DAS cluster and Simple Isilon Authentication - Not recommended

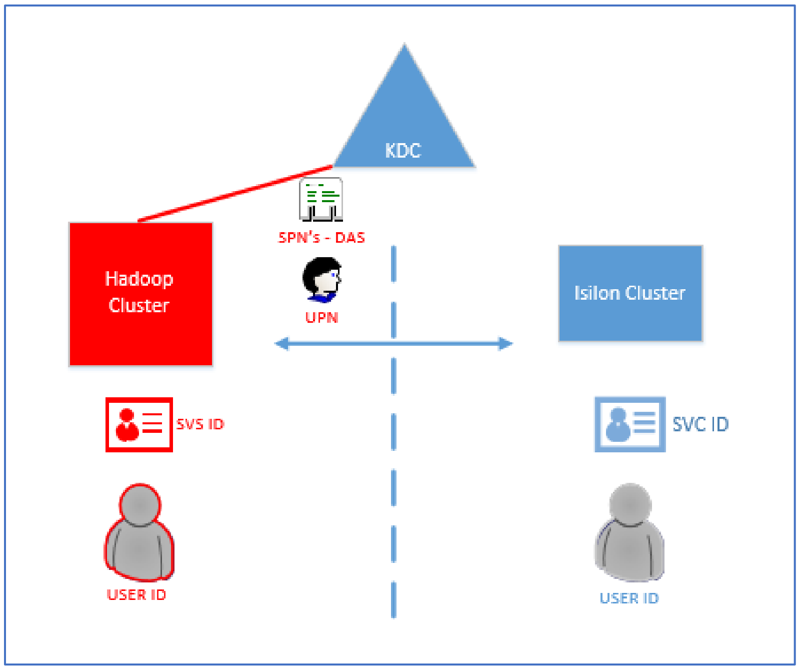


Figure 29. Hadoop Tiered Storage with Kerberized DAS and Simple Isilon security with local accounts

In this model, the Hadoop cluster is Kerberized, but the Isilon is not. This is not a recommended deployment model, as no security is in place on the Isilon cluster to support the Kerberized Hadoop cluster. It is also common to see the following error when a Kerberized cluster attempts to access a non-Kerberized cluster:

```
hadoop fs -ls hdfs://isilon-tier.hadoop.foo.com:8020/ls: Failed on local exception: java.io.IOException: Server
asks us to fall back to SIMPLE auth, but this client is configured to only allow secure connections.; Host
Details : local host is: "centos-02.hadoop.foo.com/10.246.156.7"; destination host is: "isilon-
tier.hadoop.foo.com":8020;
```

3. Kerberized DAS and Isilon Clusters with Local User Accounts

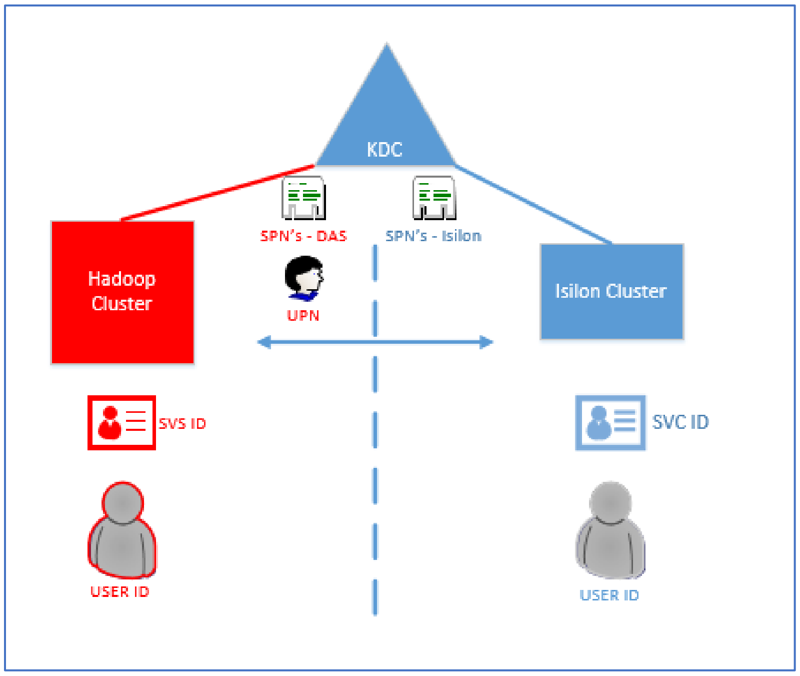


Figure 30. Hadoop Tiered storage leveraging Kerberos authentication with local accounts

Account Type	Hadoop Identity	KDC Authenticated	Isilon Identity
Hadoop Service Account	Local ID	Yes	Local ID
User Accounts	Local ID	Yes	Local ID

A secure HTS model is for the Hadoop and Isilon clusters to be Kerberized against the same KDC while leveraging local accounts only. This provides secure access to both the Hadoop data and any HTS data on the Isilon tier with any Hadoop-based Identity needing to access data on the Isilon HTS to have a corresponding local user identity. This is a simple secure solution where user parity is maintained for any users or services accessing data on both tiers. It is still recommended to build out the Isilon HTS Access Zone with the standard HDFS directory structure and all the required local Hadoop service accounts.

In order to complete the Kerberization, the following SPNs are needed for Isilon:

- hdfs/smartconnectname
- HTTP/smartconnectname

4. Kerberized DAS and Isilon Clusters with Directory Service User Accounts

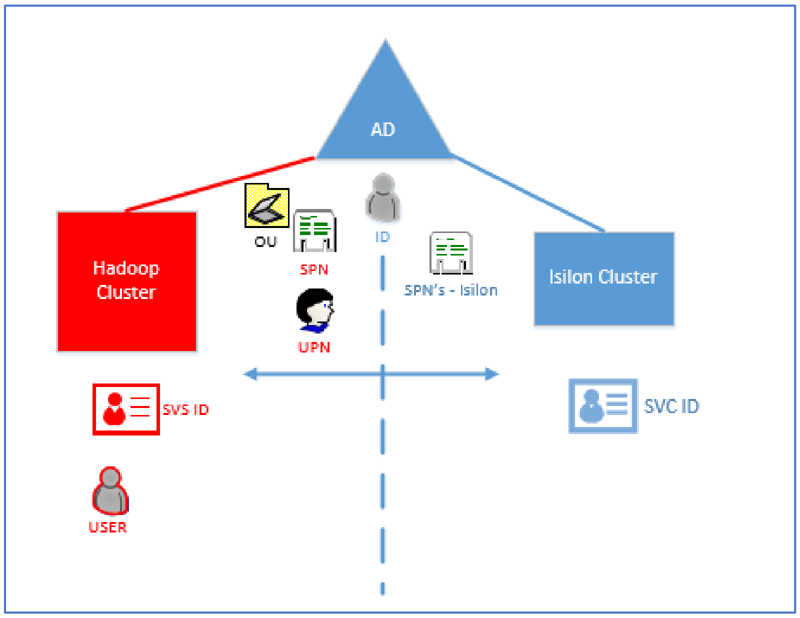


Figure 31. Hadoop Tiered storage leveraging Kerberos authentication with Directory Service user accounts

Account Type	Hadoop Identity	KDC Authenticated	Isilon Identity
Hadoop Service Account	Local ID	Yes	Local ID
User Accounts	Directory Service	Yes	Directory Service

Another secure HTS model is for the Hadoop and Isilon clusters to be Kerberized against the same Active Directory. This continues to use local service account identities while user accounts are in the Directory Service. This provides secure Kerberized access to both the Hadoop data and any HTS data on the Isilon tier with the benefit of any Hadoop user Identity needing to access data on the Isilon cluster. HTS is already provisioned by the Active Directory provider. This limits administrative overhead and can simplify deployment of HTS to Isilon. This is a secure solution where user parity is maintained for any users or services accessing data on both tiers. It is still recommended to build out the Isilon HTS Access Zone with the standard HDFS directory structure and all the required local Hadoop service accounts. However, we no longer need to manage the local account on the Isilon cluster for standard user access.

In order to complete the Kerberization, the following SPNs are needed for Isilon:

- hdfs/clustername
- hdfs/smartconnectname
- HTTP/smartconnectname

The above deployment models illustrate how to integrate security and identity management into Isilon Hadoop Tiered Storage to provide secure access to data across tiers using Kerberos and Directory services. These deployment models are really no different than the integrated models and provide the same level of flexibility and secure access.

Best Practices and Approaches

As we have demonstrated, many different methods of deployment exist for integrating security into Isilon Hadoop cluster configurations. The approaches and best practices will differ depending on the specifics of the deployment selected. The following points outline some general best practices that should be followed prior to deployment:

- Evaluate and select the Authentication and Identity Management model before deployment of Hadoop and Isilon integration begins
- Implement and validate the Simple security model before Kerberos is enabled
- Validate all Kerberos requirements to ensure Kerberos is operational, for example, DNS, Reverse DNS, and NTP
- Follow all best practices in the Hadoop distribution Kerberos deployment guides
- Implement Kerberos in all deployments to ensure valid authentication and authorization
- UID/GID parity between all Isilon and Hadoop Local Service accounts
- If using Active Directory for User accounts, always use the RFC2307 schema extensions to provide all POSIX IDs
- Consult the following knowledgebase article: https://support.emc.com/kb/335338 if using AD for POSIX attributes with RFC2307
- All Isilon SPNs must be managed by Isilon to support valid keytabs
- Review and follow the Isilon Hadoop Kerberos Deployment guides before deployments are started: https://community.emc.com/docs/DOC-61379
- The Isilon for Hadoop Best practice Guides provides additional deployment information: http://www.emc.com/collateral/white-paper/h12877-wp-emc-isilon-hadoop-best-practices.pdf

Summary

This paper has outlined the considerations and implementations of different security models in Hadoop and Isilon-integrated deployments. It has discussed the underlying requirements needed to deploy secure data access to HDFS data and many of the options available. Additional deployment models not illustrated here do exist and are supported by OneFS and Hadoop assuming they can meet the core authentication and access management requirements.

Kerberos and Identity Management can be a complex solution to integrate. Hopefully, the design patterns outlined here will simplify the approach and facilitate easier deployment and integrations. The flexibility and power of OneFS's unified permission model are clearly demonstrated here by offering many options to complete a secure Hadoop – Isilon integration.

Resources

- Using Hadoop with Isilon - Isilon Info Hub
- https://www.cloudera.com/documentation/enterprise/latest/topics/cdh_sg_hadoop_security_active_directory_integrate.html

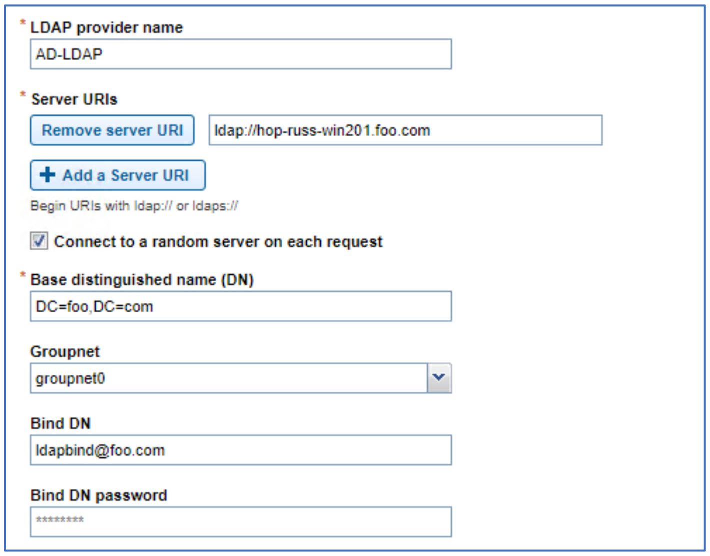
Appendix

Configuring an LDAP Provider against Active Directory

To use an LDAP provider with OneFS against an Active Directory to pull identity but not use the Kerberos authentication, the LDAP provider needs to be modified to query the schema correctly. This configuration is used when the AD is used just for IDs, common in a Trusted KDC - AD configuration, as follows:

1. Add an LDAP Provider.

- Provider the URI of one or multiple LDAP Servers, the DC in this case, as we are using AD
- Provide the Distinguished Name of the AD Domain
- Provide a user account: BIND DN and password which can bind to the AD Domain to query it (a normal user account works)



* LDAP provider name: AD-LDAP

* Server URIs: ldap://hop-russ-win201.foo.com

✓ Connect to a random server on each request

* Base distinguished name (DN): DC=foo,DC=com

Groupnet: groupnet0

Bind DN: ldapbind@foo.com

Bind DN password: ********

Figure 32. LDAP provider connecting to an Active Directory Domain

2. Modify the following properties of the LDAP provider to allow it to correctly query the AD schema. Without this modification, the LDAP query to the AD will not return results correctly and lookups will fail.

- Name Attribute: uid -- > sAMAccountname
- User Filter: (objectClass=posixAccount) -- > User
- Group Filter: (objectClass=posixGroup) -- > Group
- Homedir Attribute: homeDirectory -- > UNIXhomeDirectory

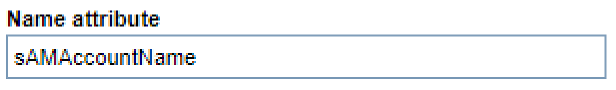
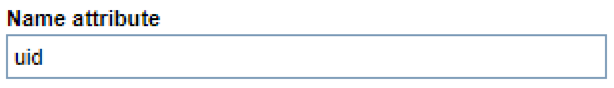
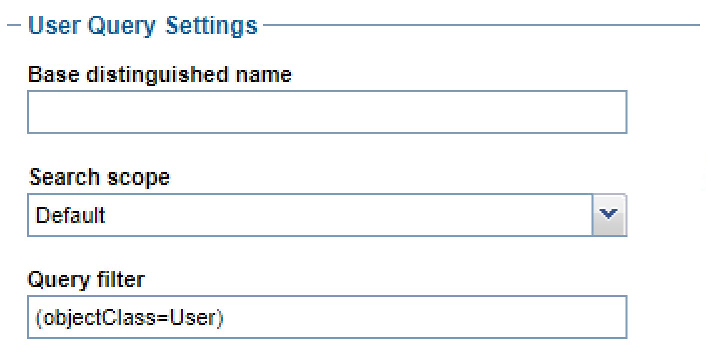
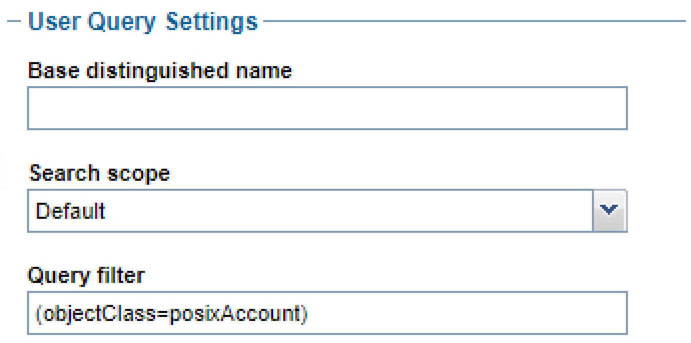
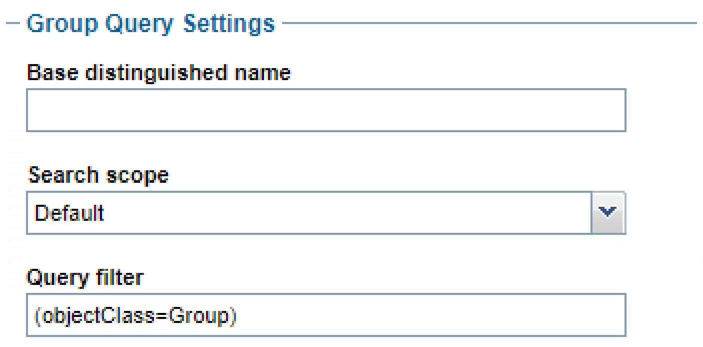
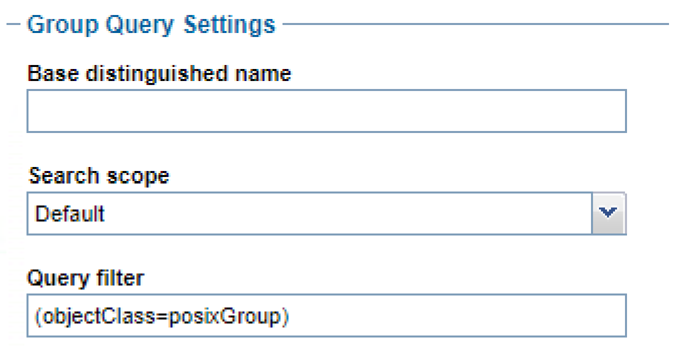
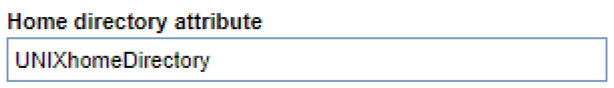
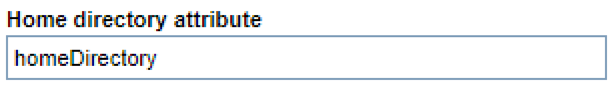
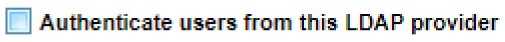
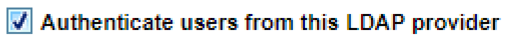
MODIFIED ATTRIBUTE	ORIGINAL ATTRIBUTE
 Name attribute: sAMAccountName	 Name attribute: uid
 User Query Settings Base distinguished name: Search scope: Default Query filter: (objectClass=User)	 User Query Settings Base distinguished name: Search scope: Default Query filter: (objectClass=posixAccount)
 Group Query Settings Base distinguished name: Search scope: Default Query filter: (objectClass=Group)	 Group Query Settings Base distinguished name: Search scope: Default Query filter: (objectClass=posixGroup)
 Home directory attribute: UNIXhomeDirectory	 Home directory attribute: homeDirectory
 ☐ Authenticate users from this LDAP provider	 ☑ Authenticate users from this LDAP provider

Figure 33. LDAP fields requiring modifications

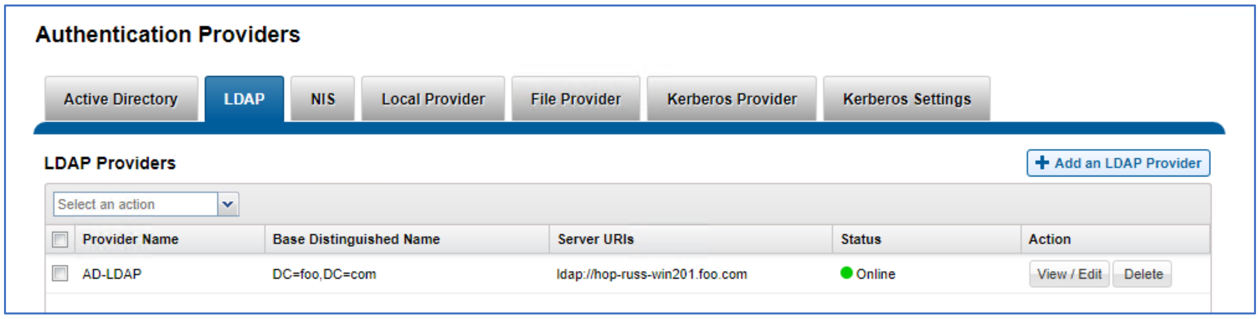
3. Review the LDAP provider.

MODIFIED CONFIGURATION OF LDAP PROVIDER	ORIGINAL DEFAULT CONFIGURATION OF LDAP PROVIDER
isilon01-1# isi auth ldap view --provider-name=AD-LDAP Name: AD-LDAP Base DN: DC=foo,DC=com Server Uris: ldap://hop-russ-win201.foo.com Status: online Alternate Security Identities Attribute: - Authentication: No Balance Servers: Yes Bind DN: ldapbind@foo.com Bind Timeout: 10 Certificate Authority File: - Check Online Interval: 3m CN Attribute: cn Create Home Directory: No	isilon01-1# isi auth ldap view --provider-name= AD-LDAP Name: AD-LDAP Base DN: DC=foo,DC=com Server Uris: ldap://hop-russ-win201.foo.com Status: online Alternate Security Identities Attribute: - Authentication: Yes Balance Servers: Yes Bind DN: ldapbind@foo.com Bind Timeout: 10 Certificate Authority File: - Check Online Interval: 3m CN Attribute: cn Create Home Directory: No

Crypt Password Attribute: -	Crypt Password Attribute: -
Email Attribute: mail	Email Attribute: mail
Enabled: Yes	Enabled: Yes
Enumerate Groups: Yes	Enumerate Groups: Yes
Enumerate Users: Yes	Enumerate Users: Yes
Findable Groups: -	Findable Groups: -
Findable Users: -	Findable Users: -
GECOS Attribute: gecos	GECOS Attribute: gecos
GID Attribute: gidNumber	GID Attribute: gidNumber
Group Base DN: -	Group Base DN: -
Group Domain: LDAP_GROUPS	Group Domain: LDAP_GROUPS
Group Filter: (objectClass=Group)	Group Filter: (objectClass=posixGroup)
Group Members Attribute: memberUid	Group Members Attribute: memberUid
Group Search Scope: default	Group Search Scope: default
Groupnet: groupnet0	Groupnet: groupnet0
Home Directory Template: -	Home Directory Template: -
Homedir Attribute: UNIXhomeDirectory	Homedir Attribute: homeDirectory
Ignore TLS Errors: No	Ignore TLS Errors: No
Listable Groups: -	Listable Groups: -
Listable Users: -	Listable Users: -
Login Shell: -	Login Shell: -
Member Of Attribute: -	Member Of Attribute: -
Name Attribute: sAMAccountName	Name Attribute: uid
Netgroup Base DN: -	Netgroup Base DN: -
Netgroup Filter: (objectClass=nisNetgroup)	Netgroup Filter: (objectClass=nisNetgroup)
Netgroup Members Attribute: memberNisNetgroup	Netgroup Members Attribute: memberNisNetgroup
Netgroup Search Scope: default	Netgroup Search Scope: default
Netgroup Triple Attribute: nisNetgroupTriple	Netgroup Triple Attribute: nisNetgroupTriple
Normalize Groups: No	Normalize Groups: No
Normalize Users: No	Normalize Users: No
Nt Password Attribute: -	Nt Password Attribute: -
Ntlm Support: all	Ntlm Support: all
Provider Domain: -	Provider Domain: -
Require Secure Connection: No	Require Secure Connection: No
Restrict Findable: Yes	Restrict Findable: Yes
Restrict Listable: No	Restrict Listable: No
Search Scope: subtree	Search Scope: subtree
Search Timeout: 100	Search Timeout: 100
Shadow User Filter: (objectClass=shadowAccount)	Shadow User Filter: (objectClass=shadowAccount)
Shadow Expire Attribute: shadowExpire	Shadow Expire Attribute: shadowExpire
Shadow Flag Attribute: shadowFlag	Shadow Flag Attribute: shadowFlag
Shadow Inactive Attribute: shadowInactive	Shadow Inactive Attribute: shadowInactive
Shadow Last Change Attribute: shadowLastChange	Shadow Last Change Attribute: shadowLastChange
Shadow Max Attribute: shadowMax	Shadow Max Attribute: shadowMax
Shadow Min Attribute: shadowMin	Shadow Min Attribute: shadowMin
Shadow Warning Attribute: shadowWarning	Shadow Warning Attribute: shadowWarning
Shell Attribute: loginShell	Shell Attribute: loginShell
UID Attribute: uidNumber	UID Attribute: uidNumber
Unfindable Groups: wheel, 0, insightiq, 15, isdmgmt, 16	Unfindable Groups: wheel, 0, insightiq, 15, isdmgmt, 16
Unfindable Users: root, 0, insightiq, 15, isdmgmt, 16	Unfindable Users: root, 0, insightiq, 15, isdmgmt, 16
Unique Group Members Attribute: -	Unique Group Members Attribute: -
Unlistable Groups: -	Unlistable Groups: -
Unlistable Users: -	Unlistable Users: -
User Base DN: -	User Base DN: -
User Domain: LDAP_USERS	User Domain: LDAP_USERS
User Filter: (objectClass=User)	User Filter: (objectClass=posixAccount)
User Search Scope: default	User Search Scope: default

Table 1. LDAP Provider configuration

4. Validate the LDAP provider is online against the Active Directory.

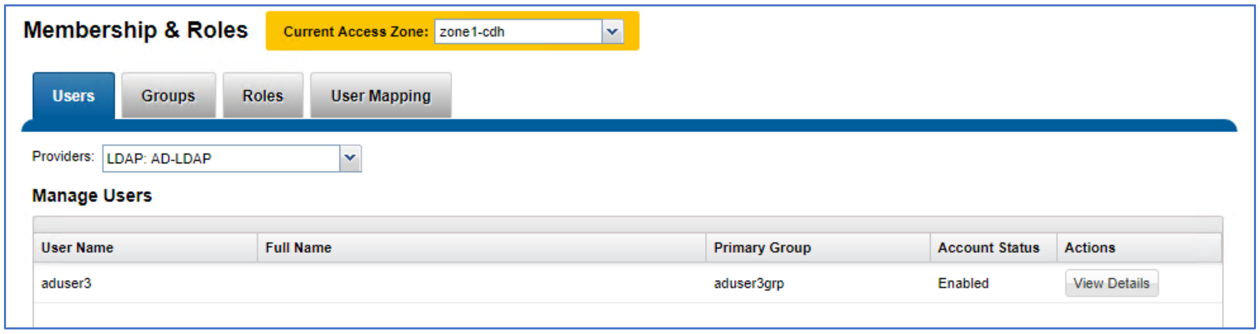

Authentication Providers

LDAP Providers

	Provider Name	Base Distinguished Name	Server URIs	Status	Action
☐	AD-LDAP	DC=foo,DC=com	ldap://hop-russ-win201.foo.com	● Online	View / Edit Delete

Figure 34. LDAP Provider

5. Query the LDAP provider for an AD-based user account information.

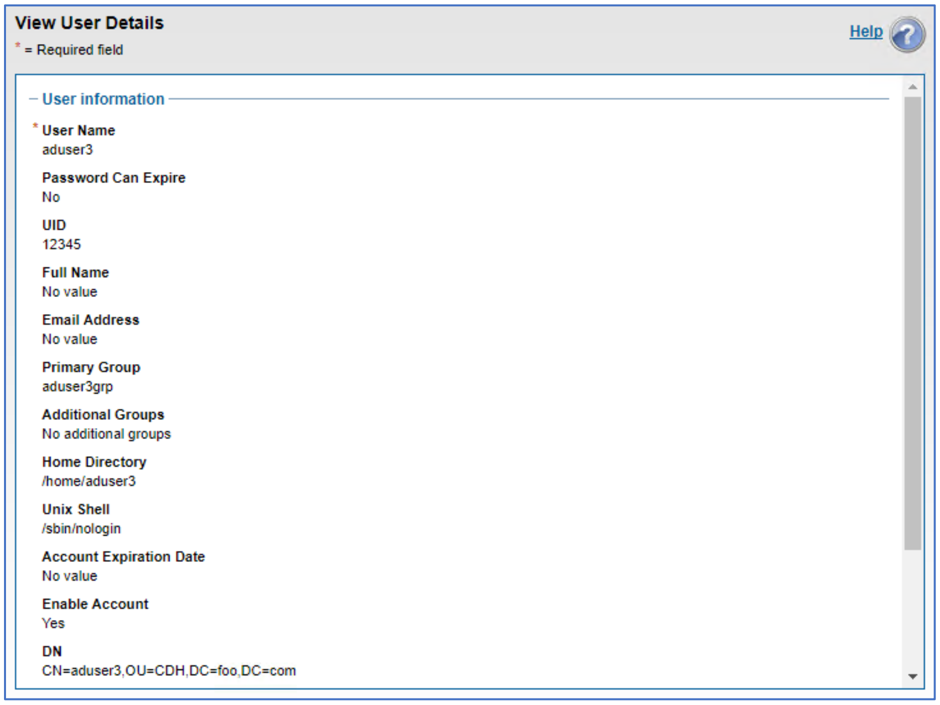

Membership & Roles

Current Access Zone: zone1-cdh

Providers: LDAP: AD-LDAP

Manage Users

User Name	Full Name	Primary Group	Account Status	Actions
aduser3		aduser3grp	Enabled	View Details


View User Details

* = Required field

User information

* User Name
aduser3

Password Can Expire
No

UID
12345

Full Name
No value

Email Address
No value

Primary Group
aduser3grp

Additional Groups
No additional groups

Home Directory
/home/aduser3

Unix Shell
/sbin/nologin

Account Expiration Date
No value

Enable Account
Yes

DN
CN=aduser3,OU=CDH,DC=foo,DC=com

Figure 35. AD based user account