White Paper

IDENTITIES, ACCESS TOKENS, AND THE ISILON ONEFS USER MAPPING SERVICE

Abstract

The OneFS user mapping service combines a user's identities from different directory services into a single access token and then modifies it according to the rules that you set. This paper explains how to map identities across directory services to uniformly control access to the OneFS file system.

October 2013

EMC²

Part Number H12417

Table of Contents

Introduction

The EMC® Isilon® NAS platform combines modular hardware with unified software to harness unstructured data. Powered by the distributed OneFS® operating system, an EMC Isilon cluster delivers a scalable pool of storage with a global namespace and multiprotocol data access.

Multiprotocol data access, however, usually coincides with networks that contain several directory services, such as Microsoft Active Directory for Server Message Block (SMB) and Lightweight Directory Access Protocol (LDAP) for Network File System (NFS). As directory services proliferate in multiprotocol environments, the OneFS user mapping service plays a central role in processing identities.

The mapping service provides an integrated way to specify complete, cross-domain access tokens. When a user with accounts in several directory services logs in to an Isilon cluster, the user mapper combines the user's identities and privileges from all the directory services into a native access token. OneFS uses the token to identify the user and control access to directories and files.

The user mapper lets you combine and control a user's identities. After the user mapper collects a user's identities from the directory services, the user mapper modifies the token by applying a set of rules—rules that you, as a system administrator, can create.

You can, for example, set a user-mapping rule to merge an Active Directory identity and an LDAP identity into a single token that works for access to files stored over both SMB and NFS. The token can include groups from Active Directory and LDAP.

The mapping rules that you create can solve identity problems by manipulating access tokens in many ways, including the following:

- Authenticate a user with Active Directory but give the user a UNIX identity
- Merge several identities into a single token
- Select a primary group from competing choices in Active Directory or LDAP

This paper explains how OneFS maps identities. By controlling identities with the user mapping service, you can solve identity problems in a NAS security context of multiprotocol data access governed by disparate directory services.

Identity management, authentication, and access control

The OneFS operating system presents an Isilon cluster's file system as a single shared volume—a single *namespace*—that supports common data-access protocols, such as SMB, FTP, and NFS. Linux machines access the Isilon cluster's file system using NFS; Microsoft Windows computers access the file system using SMB. The default shared directory of OneFS, which is `ifs`, lets clients running Windows, UNIX, Linux, or Mac OS X access the same directories and files.

To securely support NFS and SMB clients, OneFS does three things:

- Connects to directory services, such as Active Directory, NIS, and LDAP, which are also known as identity management systems and authentication providers; a directory service provides a security database of user and group accounts along with their passwords and other account information

- Authenticates users and groups
- Controls access to directories and files

When a user connects to an Isilon cluster, OneFS checks the directory services to which it is connected for an account for the user. If OneFS finds an account that matches the user's login name, OneFS verifies the user's identity—that is, it authenticates the user. During authentication, OneFS creates an access token for the user. The token contains the user's full identity, including group memberships, and OneFS uses the token later to help control access to directories and files.

Because OneFS can authenticate users with different directory services, OneFS must map a user's account from one directory service to the user's accounts in other directory services—a process known as *user mapping*. A Windows user account managed in Active Directory, for example, is mapped by default to a corresponding UNIX account with the same name in NIS or LDAP. As a result, with a single token, a user can access files that were stored by a Windows computer over SMB and files that were stored by a UNIX computer over NFS.

You can let OneFS manage the mappings with its default settings, or you can set rules with the user mapping service to control how OneFS associates user accounts.

Access zones

With OneFS 7.0 or later, the user mapping service works in a OneFS access zone. An access zone is a context that you can set up to control access based on an incoming IP address. The purpose of an access zone is to define a list of authentication providers that apply only in the context of the zone. You can then set user mapping rules in the zone to manipulate the access tokens of the users from its authentication providers.

But until you add an access zone, your NFS mounts, SMB shares, and authentication providers operate in the default access zone, which is known as the *system access zone*. Although you can use the user mapper in the access zones that you create as well as in the system access zone, the mapping rules apply only in the access zone in which you created them.

There is no requirement to create an access zone; you can use the default system access zone to apply rules with the user mapping service.

To view a list of access zones on your Isilon cluster, log in to a node with SSH and run the following command:

```
isi zone zones list
```

In the following output, only one access zone exists—the default zone named `System`:

```
Name
------
System
------
Total: 1
```

The following command shows the authentication providers for an access zone—for example, the default zone named System:

```
isi zone zones view System

                Name: System
          Cache Size: 4.77M
       Map Untrusted:
          SMB Shares: -
      Auth Providers: -
      Local Provider: Yes
        NetBIOS Name:
      All SMB Shares: Yes
  All Auth Providers: Yes
  User Mapping Rules: -
Home Directory Umask: 0077
  Skeleton Directory: /usr/share/skel
             Zone ID: 1
```

Although you can create access zones to consolidate SMB shares, access zones are not NFS aware. For more information on access zones and the how to manage them, see the OneFS Administration Guide and the OneFS Command Reference.

Situating the user mapping service

The user mapper is one of several discrete OneFS mechanisms that take part in creating and manipulating an access token.

When you log on to an Isilon cluster, the user mapper expands your identity to include your other identities from all the directory services, including Active Directory, LDAP, and NIS. After the user mapper maps your identities across the directory services, OneFS generates an access token that includes the identity information associated with your accounts. A token, for instance, includes the following identifiers:

- A UNIX user identifier, or UID
- A UNIX group identifier, or GID
- A security identifier, or SID, for a Windows user account
- A primary group SID for a Windows group account

The token also contains privileges. A privilege is the right that you, as an administrator, assign to a role so that a user or group who is a member of the role can do something on the system, such as executing a command. On OneFS, an access token lists privileges that stem from administrative role-based access control (RBAC).

A file, meanwhile, contains permissions. On OneFS, the permissions appear as an access control list, or ACL. The ACL controls access to directories, files, and other securable system objects. When a user tries to access a file, OneFS compares the identities in the user's access token with the file's ACL. OneFS grants access when the file's ACL includes an access control entry, or ACE, that allows the identity in the token to access the file and does not include an ACE that denies the identity access.

You can visualize how OneFS compares the access token of a user with the ACL of a file by viewing a user's token and a file's ACL side by side. The following example shows a user's token on an Isilon cluster running OneFS 7. (All the commands and examples in this paper are from an Isilon cluster running OneFS 7.)

```
isi auth mapping token --user=MAINE-UNO\jsmith
          User
            Name : MAINE-UNO\jsmith ❶
             UID : 1000000
             SID : S-1-5-21-3542649673-1571749849-686233814-1117
         On Disk : S-1-5-21-3542649673-1571749849-686233814-1117
           ZID: 1
          Zone: System
   Privileges: -
Primary Group
           Name : MAINE-UNO\domain users
             GID : 1000000
             SID : SID:S-1-5-21-3542649673-1571749849-686233814-513
Supplemental Identities
            Name : MAINE-UNO\marketing ❷
             GID : 1000001
             SID : SID:S-1-5-21-3542649673-1571749849-686233814-1109
            Name : Users
             GID : 1545
             SID : S-1-5-32-545
             ------------------
```

And here is the ACL for a file on the cluster:

```
-rwxr--r-- + 1 MAINE-UNO\jsmith MAINE-UNO\marketing 2056 Feb 2 10:18
adocs.txt
OWNER: user:MAINE-UNO\jsmith
GROUP: group:MAINE-UNO\marketing
0: user:MAINE-UNO\jsmith allow ❸
file_gen_read,file_gen_write,file_gen_execute,std_write_dac
1: group:MAINE-UNO\marketing allow file_gen_read ❹
2: everyone allow file_gen_read
```

❶ In the token, the user's identity is MAINE-UNO\jsmith, which is an Active Directory account.

❷ The token also shows that the user is a member of the marketing group.

❸ In the ACL, an access control entry shows that MAINE-UNO\jsmith is allowed to access the file. The ACE also lists the user's permissions, such as file_gen_write, which is permission to write to the file.

❹ This ACE shows that members of the marketing group are allowed to read the file.

This paper dissects access tokens with increasing rigor. Later sections analyze the anatomy of an access token that combines identities from different directory services. Later sections also demonstrate how to modify an access token by setting rules with the user mapping service.

For information about access control lists, including a description of the permissions and how they correspond to POSIX mode bits, see the paper titled "EMC Isilon Multiprotocol Data Access with a Unified Security Model" on the EMC website.

Managing a token

On OneFS, four system processes play a role in generating a token:

1. Looking up a user in the directory services to which OneFS is connected
2. Associating the user's *identifiers*—that is, *ID* mapping—across the directory services
3. Combining access tokens from different directory services—that is, *user* mapping
4. Determining an on-disk identity

Looking up users in directory services

OneFS works with the following directory services to authenticate users and control access to files:

- Microsoft Active Directory (AD), a popular directory service that uses domain controllers to authenticate users and authorize access to resources; to work with UNIX and Linux systems, Active Directory includes optional support for LDAP attributes with an implementation of RFC 2307
- Lightweight Directory Access Protocol (LDAP), an application protocol for querying and modifying directory services
- Network Information Service (NIS), a client/server directory service for distributing system information such as user names
- Local users and local groups: unlike Active Directory, LDAP, and NIS, which are external systems, the local provider is an internal component of OneFS
- File provider for working with accounts in `/etc/spwd.db` and `/etc/group` files; the file provider, which is also a OneFS component, lets you copy UNIX user and group account information from other systems

When a user connects to an Isilon cluster, OneFS looks up the user in the directory services and expands the user's access token to include the user's other identities and groups. Both the ID mapping service and the user mapping service play a role in expanding the user's identity to include account information from other directory services.

To view the directory services that a cluster running OneFS 7.0 or later is connected to, log in to a node and run the following command as root:

```
isi auth status
```

```
ID                                              Active Server                Status
-------------------------------------------------------------------------------------
lsa-activedirectory-provider:YORK.HULL.EXAMPLE.COM yorkdns.york.hull.example.com online
lsa-local-provider:System                       -                            active
lsa-file-provider:System                        -                            active
lsa-ldap-provider:colorado                      -                            online
-------------------------------------------------------------------------------------
Total: 4
```

In the OneFS user interfaces, directory services are usually called *authentication providers*.

ID mapping

Although their names are similar, the ID mapping service differs from the user mapping service. The goal of the ID mapping service is to map Windows SIDs to UNIX UIDs and vice versa in order to provide consistent access across protocols.

During authentication, the ID mapping service associates Windows identifiers with UNIX identifiers. When a user connects to an Isilon cluster over SMB, the ID mapping service maps the user's SIDs to UIDs and GIDs for access to files that were stored over NFS. By default, the ID mapping service matches accounts with the same name.

In contrast to SMB, when a user connects to the cluster over NFS, the ID mapping service does not map the UID and GIDs to SIDs by default, because the default identity stored on disk is already in the form of a UID and GIDs. Thus, with NFS, mapping the UID and GIDs to SIDs is unnecessary.

The following example illustrates the default ID mapping for a Windows user who authenticates with Active Directory, but has a corresponding account with the same name in LDAP. Because the user also has an account in LDAP with the same name, OneFS by default maps the user's Windows SID to the user's UID in LDAP and adds the UID to the token. (The formatting of an access token looks slightly different with OneFS 7.0 or later than with OneFS 6.5 or earlier; as this paper focuses exclusively on OneFS 7.0 or later, the following token is from OneFS 7.0. It has been abridged to remove some immaterial information.)

```
isi auth mapping token --user=york\\stand
         User
           Name : YORK\stand ❶
            UID : 4236 ❷
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111 ❸
        On Disk : 4236
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
          Name : YORK\york_sh_udg
            GID : 1000000 ❹
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133
Supplemental Identities
           Name : YORK\sd-york space group
            GID : 1000002
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
```

❶ The primary user name, which in this case comes from Active Directory

❷ The UID of the account for the user named stand in LDAP.

❸ The user's SID from Active Directory.

❹ Because the primary group account is from Active Directory, and no matching group name was found in LDAP, OneFS automatically allocates a GID to the group from the 1,000,000 to 2,000,000 range. OneFS also assigns a GID to the group in the list of supplemental identities.

You can manage some aspects of ID mapping by using the `isi auth mapping` command; see the OneFS Command Reference and the OneFS Administration Guide.

User mapping

While the ID mapper links account identifiers like UIDs and SIDs across directory services, the user mapping service combines access tokens from different directory services into a single token. For example, the user mapping service maps a Windows account named DESKTOP\jane from Active Directory to a UNIX account named jane from LDAP and generates an access token that combines the group membership information from the two accounts. Later sections of this paper provide examples that illustrate how the user mapping service combines tokens.

On-disk identity

After the user mapper combines tokens, OneFS determines an authoritative identifier for the primary user and group—the preferred on-disk identity. Your configuration options for the on-disk identity are UNIX, SID, and native. When you upgrade from a previous version of OneFS, the on-disk identity is set to UNIX, which was the default identity type on older versions of OneFS. The SID on-disk identity is for a homogeneous network of Windows systems managed only with Active Directory. On new installations, the on-disk identity is set to native.

Although you can change the type of on-disk identity, the native identity option is likely to be the best for a network with UNIX and Windows systems. In native mode, OneFS favors setting the UID as the on-disk identity because doing so improves NFS performance. OneFS stores only one type of identifier—either a UID and a GID or a SID—on disk at a time.

The native configuration option for the on-disk identity allows the OneFS authentication daemon to select the correct identity to store on disk by checking for the on-disk identity types in the following order:

1. Algorithmic mapping: An SID that matches S-1-22-1-UID or S-1-22-2-GID in the internal ID mapping database is converted back to the corresponding UNIX identity and the UID and GID are set as the on-disk identity.
2. External mapping: A user that has an explicit UID and GID defined in a directory service like Active Directory with RFC 2307 attributes, LDAP, NIS, or the OneFS file provider or local provider with the UNIX identity set as the on-disk identity.
3. Persistent mapping: Mappings stored persistently in the identity mapper database; an identity with a persistent mapping in the identity mapper database uses the destination of that mapping as the on-disk identity, a situation that primarily occurs with manual ID mappings. For example, if there is an ID mapping of GID:10000 to S-1-5-32-545, a request for the on-disk storage of GID:10000 returns S-1-5-32-545.
4. No mapping: If a user lacks a UID or GID even after querying the other directory services and identity databases, its SID is set as the on-disk identity. In addition, to make sure a user can access files over NFS, OneFS allocates a UID and GID from a preset range of 1,000,000 to 2,000,000 and then stores them persistently for later authentication requests. In native mode, a UID or GID that OneFS generates is never set as the on-disk identity.

You can view the on-disk identity by logging in to OneFS with SSH and running the following authentication mapping command. Here is an example in which the on-disk identity option is set to native. In this case, the user has no explicit mapping defined in an external directory service and no mapping stored persistently in the identity mapper database. As a result, OneFS sets the user's SID from Active Directory as the on-disk identity; the following output is truncated to focus on the on-disk identity:

```
isi auth mapping token --user=york\\stand
         User
           Name : YORK\stand
            UID : 1000000 ❶
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111
        On Disk : S-1-5-21-1195855716-1269722693-1240286574-591111 ❷
```

❶ A UID that OneFS automatically generated because the user lacked it.

❷ The user's on-disk identity, which in this case is the SID from Active Directory. The SID, instead of the UID, is set as the on-disk identity because the on-disk identity type is set to native and because the UID was automatically generated by OneFS.

Note: All the examples, best practices, and use cases in this paper assume that the on-disk identity is set to native.

For more information on setting the on-disk identity, see the OneFS Administration Guide. The EMC Isilon documentation portal includes additional best practices on working with several directory services. In particular, see the white paper titled "EMC Isilon Multiprotocol Data Access with a Unified Security Model" on the EMC website.

As a best practice, if you change the on-disk identity, you should run the repair permissions job; for instructions, see the OneFS Administration Guide.

Mapping users across systems and identities

User mapping provides a way to control access by specifying a user's complete list of security identifiers, user identifiers, and group identifiers. OneFS uses the identifiers—which are commonly called SIDs, UIDs, and GIDs respectively—to determine ownership and check access.

With the user mapping service, you can apply rules to manipulate a user's access token by modifying which identity OneFS uses, adding supplemental user identities, and changing a user's group membership. OneFS maps users only during login or protocol access.

Default mappings

If you do not set up rules, a user authenticating with one directory service receives full access to the identity information in other directory services when the account names are the same. For example, a user who authenticates with an Active Directory domain as Desktop\jane automatically receives identities for the corresponding UNIX user account for jane from LDAP or NIS.

In the most common scenario, OneFS is connected to two directory services, Active Directory and LDAP. In such a case, the default mapping provides a user with a UID from LDAP and a SID from the default group in Active Directory. The user's groups come from Active Directory and LDAP, with the LDAP groups added to the list. To pull groups from LDAP, the mapping service queries the `memberUid`. The user's home directory, gecos, and shell come from Active Directory.

The following series of examples demonstrate how OneFS builds an access token for a Windows user who authenticates with Active Directory, but has a corresponding account with the same name in LDAP. No user mapping rules are in place.

First, you can view a user's token from only Active Directory by running the following command and targeting the user's Active Directory domain account. The output is abridged to remove some immaterial information.

```
isi auth users view --user=york\\stand --show-groups
                    Name: YORK\stand
                      DN: CN=stand,CN=Users,DC=york,DC=hull,DC=example,DC=com
              DNS Domain: york.hull.example.com
                  Domain: YORK
                Provider: lsa-activedirectory-provider:YORK.HULL.EXAMPLE.COM
        Sam Account Name: stand
                     UID: 4326
                     SID: S-1-5-21-1195855716-1269722693-1240286574-591111
           Primary Group
                        ID : GID:1000000
                      Name : YORK\york_sh_udg
      Additional Groups: YORK\sd-york space group
                         YORK\york_sh_udg
                         YORK\sd-york-group
                         YORK\sd-group
                         YORK\domain users
```

Next, you can view a user's token from only LDAP by running the following command and targeting the user's LDAP account. The output is abridged.

```
isi auth user view --user=stand --show-groups
                    Name: stand
                      DN: uid=stand,ou=People,dc=colorado4,dc=hull,dc=example,dc=com
              DNS Domain: -
                  Domain: LDAP_USERS
                Provider: lsa-ldap-provider:Unix LDAP
        Sam Account Name: stand
                     UID: 4326
                     SID: S-1-22-1-4326
           Primary Group
                        ID : GID:7222
                      Name : stand
      Additional Groups: stand
                         sd-group
                         sd-group2
```

When there are no mapping rules and when the user logs in to the cluster over SMB, OneFS authenticates the user with Active Directory and builds an access token that prioritizes the account information from Active Directory, but appends the supplemental groups from the UNIX LDAP token to the end of the final token:

```
isi auth mapping token --user=york\\stand
         User
           Name : YORK\stand ❶
            UID : 4326 ❷
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111 ❸
        On Disk : 4326
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
          Name : YORK\york_sh_udg ❹
            GID : 1000000
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133
Supplemental Identities
           Name : YORK\sd-york space group ❺
            GID : 1000002
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
           Name : YORK\sd-york-group
            GID : 1000004
            SID : S-1-5-21-1195855716-1269722693-1240286574-475739
           Name : YORK\sd-group
            GID : 1000003
            SID : S-1-5-21-1195855716-1269722693-1240286574-169779
           Name : YORK\domain users
            GID : 1000001
            SID : S-1-5-21-1195855716-1269722693-1240286574-513
           Name : Users ❻
            GID : 1545
            SID : S-1-5-32-545
           Name : sd-group ❼
            GID : 100001
            SID : S-1-22-2-100001
           Name : sd-group2
            GID : 100002
            SID : S-1-22-2-100002
```

❶ The primary username is from Active Directory

❷ The user's UID is from LDAP

❸ The user's SID is from Active Directory

❹ The primary group is from Active Directory

❺ These supplemental identities are from Active Directory, as indicated by the name of the domain before the name of the group

❻ The group named Users and its GID of 1545 is a built-in OneFS local group that comes from the OneFS local provider; it appears in the token by default because the OneFS operating system adopts the standard Microsoft Windows practice of adding the Domain Users group to the local users group when the system is joined to an Active Directory domain

❼ These last two groups are appended to the token from the UNIX LDAP token

Note, however, that the mapping service leaves out the user's LDAP primary group. You can add the primary group from LDAP to the final token by creating a user mapping rule; a later section of this paper demonstrates how.

By default, when you run the `isi auth mapping` command with a UNIX username, OneFS looks up the UNIX user's information from LDAP without mapping it to the UNIX user's Active Directory account information. Why? Because OneFS gives preference to using a UID to maximize NFS performance. If OneFS instead showed the information from Active Directory as well, the results of the command would have visual symmetry with the result of an `isi auth mapping` request for an AD user—which includes the information from LDAP. But the visual symmetry would come at the expense of NFS performance.

Manipulating tokens with rules

To control a user's identities, the user mapper provides three primary mechanisms:

1. An object model for access tokens. Although you can create rules without understanding the object model, it might help you visualize how OneFS processes tokens.
2. Rules for manipulating tokens. The rules include operators that determine what the rule does and options that determine how the action is carried out.
3. An engine for processing the rules and applying them to tokens. The engine imposes some constraints on rules and processes rules in a sequence that can affect their application.

Taken together, the three mechanisms provide a framework for manipulating tokens so that you can address use cases common to environments with several directory services:

- Merging several identities into a single token that works across protocols and includes supplemental groups from both Active Directory and LDAP
- Selecting a primary group when there are competing choices from Windows and UNIX
- Managing identities when Active Directory and LDAP serve as authentication providers; for example, you can authenticate with Active Directory but use a UNIX identity
- Managing identities when NIS and Active Directory serve as authentication providers

You can address each of these use cases by controlling the elements of mapping rules through the Web administration interface or the command-line interface. In addition, the OneFS Platform API lets you query or manipulate aspects of the mappings with automation.

The elements of mapping rules

The following elements affect how the user mapper applies a rule:

- The operator, which determines the operation that a rule carries out
- Fields for usernames
- Options
- A parameter
- Wildcards

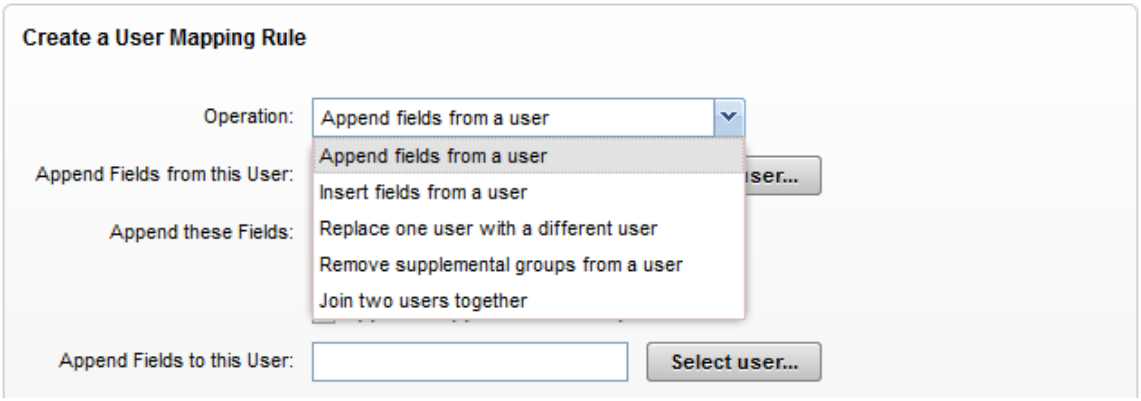
This section describes how the user mapping service interprets operators, fields, options, and wildcards. Later sections demonstrate how to combine these elements to form rules.

Creating rules with operators and usernames

You combine an operator with usernames to create a rule. The operator determines what a rule does. There are five operators:

1. append
2. insert
3. replace
4. remove
5. join

Figure 1. The operators appear in the OneFS administration interface.



Operators

The following table describes the operators that can occur in a rule. When you create a rule with the OneFS command-line interface, you must specify an operator with a symbol. The operator affects the direction in which the mapping service processes a rule; the direction of a rule is discussed later. A rule can contain only one operator.

Operator	Symbol	Direction	Description
append	++	Left to right	Modifies an access token by adding fields to it. The mapping service appends the fields that are specified in the list of options (user, group, groups) to the first identity in the rule. The fields are copied from the second identity in the rule. All appended identifiers become members of the additional groups list. An append rule without an option performs only a lookup operation; you must include an option to alter a token. Options are discussed later.
insert	+=	Left to right	Modifies an existing token by adding fields to it. Fields specified in the options list (user, group, groups) are copied from the new identity and inserted into the identity in the token. When the rule inserts a primary user or primary group, it becomes the new primary user or primary group in the token. The previous primary user or primary group moves to the additional identifiers list. Modifying the primary user leaves the token's username unchanged. When inserting the additional groups from an identity, the service adds the new groups to the existing groups.
replace	=>	Left to right	Removes the token and replaces it with the new token that is identified by the second username. If the second username is left blank, the mapping service removes the first username in the token, leaving no username, and then login fails with a no such user error.
remove groups	--	Unary	Modifies a token by removing the supplemental groups.
join	&=	Bidirectional	Merges the new identity into the token. If the new identity is the second user, the mapping service inserts it after the existing identity; otherwise, the service inserts it before the existing identity.

	The location of the insertion point is relevant when the existing identity is already the first in the list because OneFS uses the first identity to determine the ownership of new file system objects. A join rule is bidirectional; if the first username matches the token, the mapping service evaluates the rule as a left to right rule. If the second username matches the token, the service expands the wildcards in the first user and resolves the first user before applying the rule.

Usernames

In a rule, you can specify a user by the name of a UNIX user or by the name of an Active Directory user. To specify an Active Directory user, you must use the DOMAIN\user format, where DOMAIN is the name of an Active Directory domain. In a domain name or a user name, you can add a wildcard by using an asterisk (*).

In its most basic form, a rule combines a source username with an operator and a target username in the following format:

```
sourceUsername operator targetUsername
```

Example rule with the replace operator

Consider the following access token for a user from Active Directory named user_9440:

```
isi auth mapping token YORK\\user_9440
         User
             Name : YORK\user_9440 ❶
              UID : 1000001 ❷
              SID : S-1-5-21-1195855716-1269722693-1240286574-11547
          On Disk : S-1-5-21-1195855716-1269722693-1240286574-11547
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
             Name : YORK\domain users
              GID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-513
Supplemental Identities
             Name : Users
              GID : 1545
              SID : S-1-5-32-545
```

❶ An Active Directory user account

❷ A UID that was automatically generated by OneFS

Meanwhile, LDAP contains a user named lduser_010 with the following account information:

```
isi auth mapping token lduser_010
                   User
                     Name : lduser_010 ❶
                      UID : 10010 ❷
                      SID : S-1-22-1-10010 ❸
                  On Disk : 10010
                    ZID: 1
                   Zone: System
              Privileges: -
           Primary Group
                     Name : example
                      GID : 10000
                      SID : S-1-22-2-10000
                  On Disk : 10000
Supplemental Identities
                     Name : ldgroup_20user
                      UID : -
                      GID : 10026
                      SID : S-1-22-2-10026
```

❶ A UNIX user account in LDAP named lduser_010

❷ The user's UID in LDAP

❸ OneFS generated a SID for the account; the SID contains the UID from LDAP

The following rule uses the symbol for the replace operator to replace the Active Directory user with the user from LDAP named lduser_010:

```
isi zone zones modify System --user-mapping-rules="YORK\\user_9440 =>
lduser_010"
```

After setting the rule, you can view it with the following command:

```
isi zone zones view System
                Name: System
          Cache Size: 4.77M
       Map Untrusted:
          SMB Shares: -
      Auth Providers: -
      Local Provider: Yes
        NetBIOS Name:
      All SMB Shares: Yes
  All Auth Providers: Yes
  User Mapping Rules: YORK\user_9440 => lduser_010
Home Directory Umask: 0077
  Skeleton Directory: /usr/share/skel
       Audit Success: -
       Audit Failure: -
             Zone ID: 1
```

The rule changes the access token for the Active Directory user by replacing the identity from Active Directory with the identity from LDAP; it's the same user, but now the identity information comes from LDAP:

```
isi auth mapping token YORK\\user_9440
         User
             Name : lduser_010 ❶
              UID : 10010 ❷
              SID : S-1-22-1-10010 ❸
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
             Name : example
              GID : 10000
              SID : S-1-22-2-10000
Supplemental Identities
             Name : ldgroup_20user
              GID : 10026
              SID : S-1-22-2-10026
```

❶ A user account from LDAP

❷ The user's UID from LDAP

❸ The SID generated from the user's UID in LDAP

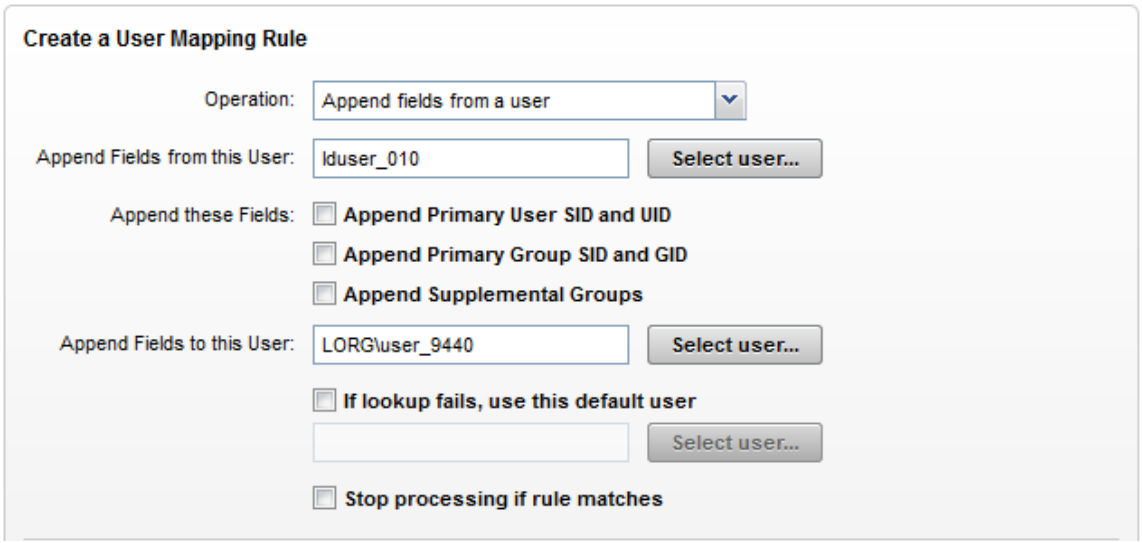
Fields and options

Rules can include options that target the fields of an access token. A field represents a section of an access token, such as the primary UID and primary user SID from a user that you select. When you create a rule, you can add an option to manipulate how OneFS combines sections of two identities into a single token. For example, an option can force OneFS to append only the supplement groups to a token.

A token includes the following fields that you can manipulate with user mapping rules:

- username
- unix_name
- primary_uid
- primary_user_sid
- primary_gid
- primary_group_sid
- additional_ids (includes supplemental groups)

Figure 2: You can see some of the fields in the OneFS administrative interface (the User in the interface is the same as the username)



You can also see the fields in a OneFS access token:

```
isi auth mapping token --user york\\stand
         User
           Name : YORK\stand ❶
            UID : 100000 ❷
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111 ❸
        On Disk : S-1-5-21-1195855716-1269722693-1240286574-591111
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
          Name : YORK\york_sh_udg
            GID : 1000000 ❹
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133 ❺
Supplemental Identities ❻
           Name : YORK\sd-york space group
            GID : 1000002
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
```

❶ The primary username

❷ The primary UID

❸ The primary user SID

❹ The primary GID

❺ The primary group SID

❻ The additional IDs are the supplemental groups that appear under the Supplemental Identities heading

The primary UID and the primary user SID always correspond to the same user; they can, however, differ from the user listed in the username field, as later examples will show. The primary GID and primary group SID always correspond to the same group.

Options

The options control how a rule combines identity information in a token. The break option is the exception: it stops OneFS from processing additional rules.

Although several options can apply to a rule, not all options apply to all operators. The following table describes the effect of each option and the operators that they work with.

Option	Description
user	The user option works with the insert and append operators. It copies the primary UID and primary user SID to the token.
group	The group option works with the insert and append operators. It copies the primary GID and primary group SID to the token.
groups	The groups option works with the insert and append operators. It copies all the additional identifiers to the token. (The additional identifiers exclude the primary UID, the primary GID, the primary user SID, and the primary group SID.)
default_user	The default user option works with all the operators except remove groups. If the mapping service fails to find the second user in a rule, the service tries to find the username of the default user. The name of the default user cannot include wildcards. When you set the option for the default user in a rule with the command-line interface, you must set it with an underscore: `default_user`.
break	The break option works with all the operators. The break option stops the mapping service from applying rules that follow the insertion point of the break option. The mapping service generates the final token at the point of the break.

Parameter

There is one global parameter: the default UNIX user. Not to be confused with the default user option, the default UNIX user ensures that the final token contains a primary UID and GID that was not automatically generated by OneFS. The parameter applies to all the rules that you set in an access zone.

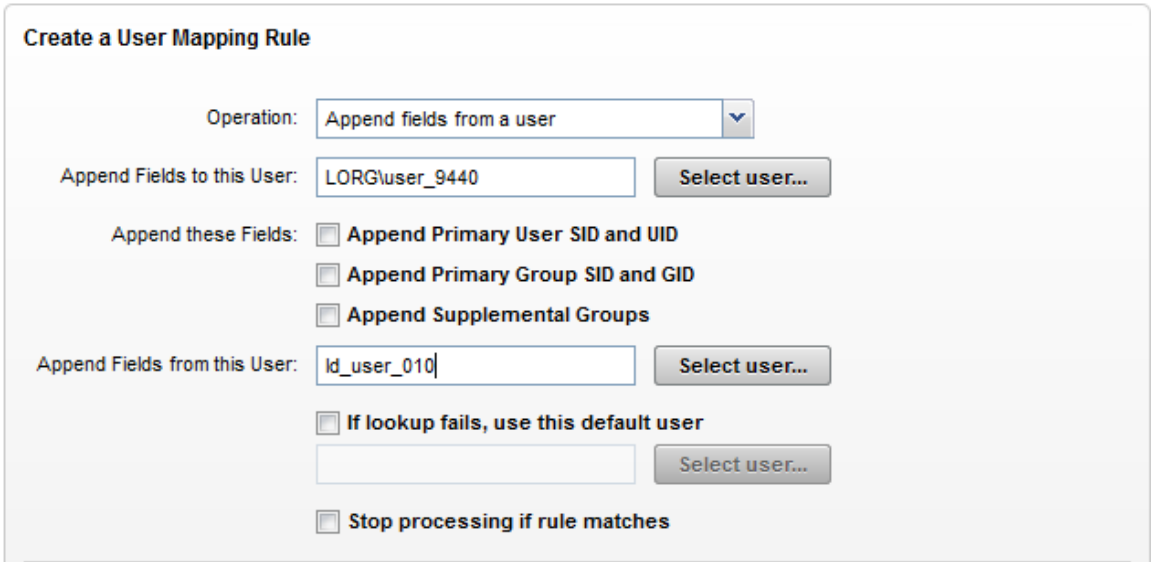
When the default UNIX user parameter is set, OneFS discards an identity without a primary UID and GID and replaces the identity with the default UNIX user. If there is no default UNIX user or if the default UNIX user has no UID or GID, OneFS denies access with an error that says `no such user`.

If you set the parameter, you should assign the default UNIX user to a well-known account without the power to execute administrative commands—for example, the nobody account.

Figure 3. In the OneFS web administration interface, you set the parameter for the default UNIX user with this form either before or after you create one or more rules.

In contrast, the default user option applies only to a rule that includes it. You set the default user option in the OneFS Web administration interface when you create a rule, as shown in Figure 4.

Figure 4. Setting the default user option



To summarize, the parameter for the default UNIX user replaces a token that lacks an authoritative UID and a primary GID after OneFS applies all the rules in the zone. The option for the default user selects a user to include in a token when the lookup of a username in a rule fails.

Example rule with the append operator and the user option

Here is an example of how a rule with the append operator and the user option affects a token. The user option appends the primary UID and primary user SID to the token.

Consider the following access token for a user from Active Directory named user_9449:

```
isi auth mapping token YORK\\user_9449
         User
             Name : YORK\user_9449
```

```
              UID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-11556
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
             Name : YORK\domain users
              GID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-513
 Supplemental Identities
             Name : Users
              GID : 1545
              SID : S-1-5-32-545
```

The following rule contains the append operator with the user option to append the LDAP user's primary UID and the primary SID to the token of the Active Directory user:

isi zone zones modify System --user-mapping-rules="YORK\\user_9449 ++ lduser_010 [user]"

After setting the rule, you can view it with the following command. The output is abridged to focus on the rule:

```
isi zone zones view System
               Name: System
                ...
          User Mapping Rules: YORK\user_9449 ++ lduser_010 [user]
```

The rule produces the following access token. The user option adds the LDAP user's primary UID and primary GID, highlighted in bold, to the Supplemental Identities section of the token:

```
isi auth mapping token YORK\\user_9449
         User
             Name : YORK\user_9449
              UID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-11556
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
             Name : YORK\domain users
              GID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-513
Supplemental Identities
             Name : Users
              GID : 1545
              SID : S-1-5-32-545
             Name : lduser_010
              UID : 10010
              SID : S-1-22-1-10010
```

More examples appear in later sections.

How OneFS evaluates rules

This section describes how OneFS processes user mapping rules as it generates a final access token.

Rules compare usernames

A rule contains at least one username, and every identity includes a username. For a mapping rule to apply, OneFS must match the username in the rule with an identity.

A username is either an Active Directory account name coupled with a domain name or a UNIX name without a domain name. The UNIX name may be the same as the Active Directory account name. If a username is not coupled with a domain name, OneFS considers it a UNIX name.

To match a username in a rule with an Active Directory user, you must couple the name of the domain or a wildcard with the username, like this:

```
DOMAIN\\username
```

or

```
*\\username
```

To match a UNIX name in LDAP, NIS, or the local provider, you must set the name in the rule without a domain name or without a wildcard for the domain.

When OneFS compares a username with an identity, OneFS ignores case by default: For usernames, OneFS makes no distinction between uppercase and lowercase unless you turn off the option to normalize usernames. (See, for instance, the `isi auth ads modify` command with the `lookup-normalize-users` option in the OneFS Command Reference.)

The direction of a rule

A rule applies in one of three directions: left to right, bidirectional, or unary. The rule's direction determines how the user mapping service matches the usernames in the rule and modifies the token.

Direction	Description
Left to right	A left to right rule contains two usernames. The mapper compares the first username with the token. If the first username matches the token, the mapping service expands the wildcards of the second username and then resolves the second username by looking it up in the authentication providers. If the lookup finds the second username, the mapping service applies the rule.
Bidirectional	A bidirectional rule contains two usernames. If the first username matches the token, the mapping service evaluates the rule as a left to right rule. If the second username matches the token, the service expands the wildcards in the first user and resolves the first user before applying the rule.
Unary	A unary rule contains only one username. If the username matches an identity, the mapping service applies the rule.

The order of evaluation

OneFS applies rules to a token sequentially. After OneFS applies all the rules in the list of rules, OneFS generates the final access token. You can, however, stop OneFS from applying more rules by inserting the break option in a rule. OneFS applies no rules that appear after the break option.

The direction of a rule and the order in which OneFS evaluates rules particularly affects how OneFS processes wildcards.

Wildcard matching

A rule can match usernames with a wildcard. If you set the Active Directory domain or the Active Directory account name as an asterisk, for instance, OneFS processes it as a wildcard. A wildcard matches any name in the field.

Further: If you use a wildcard in the source username and the target username, OneFS expands the wildcard in the target to match the value from the source.

Here are two examples of rules that include wildcards:

Source Username	Operator	Target Username	
*	join	TESTER*	❶
**	replace	guest	❷

❶ This rule matches any identity with a UNIX username and expands it to include the user with an Active Directory account of the same name in the TESTER domain. Since a join rule is bidirectional, the rule also matches any identity in the TESTER domain and expands it to match any UNIX user with the same username.

❷ This rule matches any source username in any Active Directory domain and replaces it with the UNIX user named guest.

The user mapping service rejects a rule with a nonsensical combination of wildcards. A nonsensical combination includes rules that cannot expand a wildcard from a previously matched wildcard. The mapping service also rejects a wildcard whose matching is incompatible with the direction of the rule's operation.

The following examples contain invalid wildcard-matching rules:

Source Username	Operator	Target Username
*\jdoe	join	TEST DOMAIN*
guest	replace	*

The first rule leaves a wildcard unmatched in both directions. In the second rule, the replace operator matches left to right, but the wildcard on the right is unmatched, making the rule invalid.

How OneFS creates a final access token

After OneFS applies user mapping rules, it transforms the list of identities into a final access token. OneFS creates the final access token in the following sequence:

1. OneFS combines identities by processing the user mapping rules. OneFS selects the first identity in the list as the primary identity. OneFS adds the identifiers from the additional identities to the first identity's list of additional IDs. If there are no rules, OneFS combines identities by applying its default rules.
2. After OneFS processes the rules and combines the identities, it checks whether the default UNIX user parameter is set. If the parameter is set and the token does not include a primary UID and GID yet, OneFS assigns the token the primary UID and GID of the default UNIX user.
3. OneFS generates a UID and a GID. If the primary user does not have a UID or the primary group does not have a GID, OneFS generates them.
4. OneFS selects the identity to store as the on-disk identity.

The final access token conforms to the following conditions:

- The token contains no duplicate identifiers. For instance, the token may not duplicate one of the primary identifiers in the list of additional identifiers.
- Although the list of additional identifiers may be empty, all the primary identifiers must exist.

OneFS uses the primary identifiers when it creates file system objects like directories and files. The permissions of files and directories may include UIDs and GIDs as well as SIDs. The on-disk identity determines which identifiers to include in permissions for directories and files. To control access to a file, OneFS compares all the identifiers in a token to a file's ACL and POSIX mode bits.

Creating user mapping rules

You can set user mapping rules with either the Web administration interface or the command-line interface. You can, for example, connect to a node by SSH and run commands to manage the user mappings by using the `isi zone zones modify` command.

Because OneFS is a distributed operating system, when you change a mapping on one node, OneFS propagates the change to the other nodes. OneFS stores the user mappings in the system configuration tree, which is referred to as gconfig. After a change, the OneFS authentication service reloads the configuration.

In OneFS 7.0 or later, you create a mapping rule in an access zone, and the rule applies only in the context of its zone. Because OneFS 6.x contains no access zones, a mapping rule is not limited to a zone. For instructions on how to set user mapping rules with the Web administration interface, see "Create an access zone" in the OneFS Administration Guide for OneFS 7.0 or later.

Creating a mapping rule from the command-line interface

Here is an example of how to add a mapping rule to a zone by using the command-line interface on OneFS 7.0 or later. First, consider the following token for a user from Active Directory:

```
isi auth mapping token YORK\\user_9440

         User
             Name : YORK\user_9440
              UID : 1000201
              SID : S-1-5-21-1195855716-1269722693-1240286574-11547
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
             Name : YORK\domain users
              GID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-513
 Supplemental Identities
             Name : Users
              GID : 1545
              SID : S-1-5-32-545
```

The following command creates a rule in the default access zone that merges the YORK\user_9440 from AD with a user from LDAP named lduser_010:

```
isi zone zones modify System --add-user-mapping-rules "YORK\user_9440 &=
lduser_010"
```

You can run the following command to see the rule:

```
isi zone zones view System

                Name: System
          Cache Size: 4.77M
       Map Untrusted:
          SMB Shares: -
      Auth Providers: -
      Local Provider: Yes
        NetBIOS Name:
      All SMB Shares: Yes
  All Auth Providers: Yes
  User Mapping Rules: YORK\user_9440 &= lduser_010
Home Directory Umask: 0077
  Skeleton Directory: /usr/share/skel
             Zone ID: 1
```

And then you can check the token again to see the changes:

```
isi auth mapping token YORK\\user_9440
         User
             Name : YORK\user_9440
              UID : 1000201
              SID : S-1-5-21-1195855716-1269722693-1240286574-11547
           ZID: 1
         Zone: System
   Privileges: -
Primary Group
             Name : YORK\domain users
              GID : 1000000
              SID : S-1-5-21-1195855716-1269722693-1240286574-513
Supplemental Identities
             Name : Users
              GID : 1545
              SID : S-1-5-32-545

             Name : lduser_010
              UID : 10010
              SID : S-1-22-1-10010

             Name : example
              GID : 10000
              SID : S-1-22-2-10000

             Name : ldgroup_20user
              GID : 10026
              SID : S-1-22-2-10026
```

A note about adding more rules with the command-line interface

You can create the first mapping rule with the option in the following command:

```
isi zone zones modify System --user-mapping-rules
```

If you try to add a second rule with the command above, however, it replaces the existing rule rather than adding the new rule to the list of rules. To add more rules to the list of rules, you must use the following option:

```
isi zone zones modify System --add-user-mapping-rules
```

Rule syntax and the username format

You create rules with the command-line interface using the following syntax:

```
username1 operator username2 [options]
```

If you leave one of the fields in a rule empty, OneFS omits the field when it processes the rule. The exception is the operator: a rule must contain an operator.

You can name a user in four ways:

Username format	Description
DOMAIN\user	This format matches a user account in an Active Directory domain.
username	This format matches an identity's UNIX account name. The UNIX account name can be in NIS, LDAP, the OneFS local provider, or the OneFS file provider.
empty string	An empty string can appear with the replace operator to remove the identity that the rule matches. An empty string must appear in double quotation marks without a space or other characters.
wildcard	You can replace the name of a domain, the name of an Active Directory account, and the name of a UNIX account with a wildcard, which you set with an asterisk (*).

If a username contains a special character, such as a space, you must enclose it in double quotation marks, like this:

```
TEST_DOMAIN\"John Doe" => jdoe
```

Make sure you enclose only the username, not the domain, in double quotes.

Examples of rules

The following examples combine the username formats with operators to form rules. Several of the rules include an option to specify how OneFS processes the rule. Here is an example that replaces the identity of the jane account from Active Directory with the identity of the jane account from LDAP. The break option forces OneFS to stop applying rules and to generate the token at the point of the break:

```
CORP\jane => jane [break]
```

The following rule uses wildcards to join users from the DESKTOP domain with UNIX users who have the same name in LDAP, NIS, or the local provider:

```
DESKTOP\* &= * [break]
```

Here is a rule that tightly restricts access by removing the identities of everybody other than those permitted by preceding rules:

```
*\* => ""
```

The following rule maps the administrator account from any Active Directory domain to the nobody account on OneFS. The rule exemplifies how to turn a powerful account into an innocuous account.

```
*\Administrator => nobody
```

Adding options to rules

You specify an option in square brackets, as some of the following examples demonstrate. The square brackets can include a comma-separated list of options. If you include the square brackets without listing an option, the mapping service processes the rule but applies no options.

The following command sets an insert rule that includes the group option. If you have already set a rule, however, this command replaces it:

```
isi zone zones modify System --user-mapping-rules="YORK\\user_9449 ++
lduser_010 [group]"
```

The following command adds an insert rule that includes the user option. Using this form of the command preserves an existing rule instead of replacing it:

```
isi zone zones modify System –-add-user-mapping-rules="YORK\\user_9449
++ lduser_010 [user]"
```

The following command adds an insert rule that includes the groups option:

```
isi zone zones modify System –-add-user-mapping-rules="YORK\\user_9449
++ lduser_010 [groups]"
```

The following command adds an insert rule that includes all the options that work with the insert operator:

```
isi zone zones modify System –-add-user-mapping-rules="YORK\\user_9449
++ lduser_010 [user,group,groups,default_user=nobody,break]"
```

Specifying the default user

You specify the default user as in the following example:

```
isi zone zones modify System --add-user-mapping-rules="*\jane += janey
[default_user=nobody]"
```

Specifying the default UNIX user

The syntax for specifying the default UNIX user differs from that of the default user. You must set the default UNIX user in angle brackets, as the failure of the following command shows:

```
isi zone zones modify System --add-user-mapping-rules="default_unix_user=lduser_011"

Rules parsing failed at '=': syntax error, unexpected '=', expecting BINARY_OP or
UNARY_OP
```

After placing the default UNIX user in brackets, the second attempt at running the command succeeds:

```
isi zone zones modify System --add-user-mapping-rules="<default_unix_user=lduser_011>"
```

You can now view the rules for the zone to see the default user for the rule that maps jane to janey and to see the default UNIX user that applies to all the rules up to that point in the series of rules:

```
isi zone zones view System
                Name: System
          Cache Size: 4.77M
       Map Untrusted:
          SMB Shares: -
      Auth Providers: -
      Local Provider: Yes
        NetBIOS Name:
      All SMB Shares: Yes
  All Auth Providers: Yes
  User Mapping Rules: *\Administrator => nobody [], *\johnd => *\jdoe [], IT\* &=
EX_CLUSTER\* [], YORK\user_9440 => lduser_010 [default_user=nobody], *\jane += janey
[default_user=nobody], <default_unix_user=lduser_011>
Home Directory Umask: 0077
  Skeleton Directory: /usr/share/skel
             Zone ID: 1
```

Creating rules in a text file

With the command-line interface, you can add mapping rules from a text file. Creating your mapping rules in a text file simplifies management and makes it much easier to add many rules to an access zone. The text file can include comments, usernames, wildcards, operators, and options.

You must encode a text file with user mapping rules in UTF-8. The file can contain comments set off by a number sign; a comment continues only to the end of the line. If you leave the file empty, OneFS resolves users without mapping them.

You create rules in the text file by using the same syntax as the rules that you set with the command-line interface, with the exception of the parameter for the default UNIX user:

```
username1 operator username2 [options]
```

Parameter format for default UNIX user in a text file

In a text file, you set a parameter by enclosing it in angle brackets and placing it in a separate line. The formatting of a default UNIX user that appears in the rule is the same as that for a username—for example:

```
<default_unix_user=guest>
```

Example of rules in a text file

Here is an example of how to create a set of user mapping rules in a text file and load the file into OneFS. The example text file contains comments that describe the rules. For this example, assume that the cluster is joined to an Active Directory domain named IT.

```
# Turn administrator accounts from all Active Directory domains into nobody:
*\Administrator => nobody
# Rename user johnd to jdoe from all domains but retain the original domain:
*\johnd => *\jdoe
# Join AD users and local users with the same username:
IT\* &= *
# Append the supplemental groups from IT\userx to every UNIX account:
* ++ IT\userx [groups]
```

With OneFS 7.0 or later, you can remove the comments and load the rules by running the following command. The command assumes that you first added a text file with the rules to the tmp directory and that you want to add the rules to the access zone named System. You also must run this command with the quotation marks—double, single, and especially the grave accent (`)—set in exactly the same way as the following command:

```
isi zone zones modify System --user-mapping-rules="`grep -v '#'
/tmp/rules.txt`"
```

Finally, you can view the rules to verify that OneFS loaded them:

```
isi zone zones view System
```

```
                Name: System
          Cache Size: 4.77M
       Map Untrusted:
          SMB Shares: -
      Auth Providers: -
      Local Provider: Yes
        NetBIOS Name:
      All SMB Shares: Yes
  All Auth Providers: Yes
  User Mapping Rules: *\Administrator => nobody
    *\johnd => *\jdoe
    IT\* &= *
    * ++ IT\userx [groups]
Home Directory Umask: 0077
  Skeleton Directory: /usr/share/skel
             Zone ID: 1
```

Best practices

EMC Isilon recommends the following best practices to simplify user mapping.

Active Directory with RFC 2307 and Windows Services for UNIX

A best practice is to use Microsoft Active Directory with Windows Services for UNIX and RFC 2307 attributes to manage Linux, UNIX, and Windows systems. Integrating UNIX and Linux systems with Active Directory centralizes identity management and eases interoperability, reducing the need for user mapping rules. Make sure your domain controllers are running Windows Server 2003 or later.

Naming users

The simplest configurations name users consistently so that each UNIX user corresponds to a similarly named Windows user. Such a convention allows rules with wildcards to match names and map them without explicitly specifying each pair of accounts.

No overlapping ID ranges

In networks with multiple identity sources, such as LDAP and Active Directory with RFC 2307 attributes, you should ensure that UID and GID ranges do not overlap.

It is also important that the range from which OneFS automatically allocates UIDs and GIDs does not overlap with any other ID range. The range from which OneFS automatically allocates a UID and GID is 1,000,000 to 2,000,000.

If UIDs and GIDs overlap across two or more directory services, some users might gain access to other users' directories and files.

Avoid common UIDs and GIDs

In addition, you should not use well-known UIDs and GIDs in your ID ranges because they are reserved for system accounts. UIDs and GIDs below 1000 are reserved for system accounts; do not assign them to users or groups.

No user principal names in mapping rules

You cannot use a user principal name in a user mapping rule. A user principal name (UPN) is an Active Directory domain and username that are combined into an Internet-style name with an @ sign, like an email address: jane@example.com. If you include a UPN in a rule, the mapping service ignores it and might return an error.

Rule ordering

OneFS processes every mapping rule by default. Processing every rule, though, can present problems when you apply a rule to deny all unknown users access. In addition, replacement rules may interact with rules that contain wildcard characters.

To minimize complexity, EMC Isilon recommends that you group rules by type and organize them in the following order:

1. Place the rules that replace an identity first to ensure that OneFS replaces all instances of the identity.
2. Set join, add, and insert rules second.

3. Set rules that allow or deny access last.
4. Within each group of rules, put explicit rules before rules with wildcards; otherwise, the explicit rules might be skipped.

Adding the LDAP or NIS primary group to the supplemental groups

When an Isilon cluster is connected to Active Directory and LDAP, a best practice is to add the LDAP primary group to the list of supplemental groups. This best practice lets OneFS honor group permissions on files created over NFS or migrated from other UNIX storage systems. Although the following examples cite LDAP, the same practice holds when an Isilon cluster is connected to both Active Directory and NIS.

By default, OneFS leaves the LDAP primary group, which is named stand in the following examples, off the supplemental groups list:

```
isi auth mapping token --user york\\stand


         User
           Name : YORK\stand
            UID : 100000
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
           Name : YORK\york_sh_udg
            GID : 1000008
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133
Supplemental Identities
           Name : YORK\sd-york space group
            GID : 1000010
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
           Name : YORK\sd-york-group
            GID : 1000011
            SID : S-1-5-21-1195855716-1269722693-1240286574-475739
           Name : YORK\sd-group
            GID : 100001
            SID : S-1-5-21-1195855716-1269722693-1240286574-169779
           Name : YORK\domain users
            GID : 1000009
            SID : S-1-5-21-1195855716-1269722693-1240286574-513
           Name : Users
            GID : 1545
            SID : S-1-5-32-545
           Name : sd-group2
            GID : 100002
            SID : S-1-22-2-100002
```

Because OneFS does not, by default, add the LDAP primary group to the supplemental groups, you should create a rule to add it. The following join rule fully unites the identities of a user with accounts in Active Directory and LDAP:

```
*\* &= *
```

After you implement the rule, OneFS includes the LDAP primary group in the list of supplemental identities. As the last entry demonstrates, the stand group now appears in the list:

```
isi auth mapping token --user york\\stand

         User
           Name : stand
            UID : 100000
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
           Name : YORK\york_sh_udg
            GID : 1000008
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133
Supplemental Identities
           Name : YORK\sd-york space group
            GID : 1000010
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
           Name : YORK\sd-york-group
            GID : 1000011
            SID : S-1-5-21-1195855716-1269722693-1240286574-475739
           Name : YORK\sd-group
            GID : 100001
            SID : S-1-5-21-1195855716-1269722693-1240286574-169779
           Name : YORK\domain users
            GID : 1000009
           SID : S-1-5-21-1195855716-1269722693-1240286574-513
           Name : Users
            GID : 1545
            SID : S-1-5-32-545
           Name : sd-group2
            GID : 100002
            SID : S-1-22-2-100002
           Name : stand
           GID : 100000
           SID : S-1-22-2-100000
```

Use cases

The user mapping service provides an extensible framework for manipulating tokens, so that you can address use cases common to environments with several directory services.

Merging Windows and UNIX tokens

When Windows and UNIX usernames fail to match each other across directory services, you can write rules that use either the join or the append operator to merge two usernames into a single token. For example, if a user's Windows username is win_bob and the user's UNIX username is UNIX_bob, you can join them by writing the following rule:

```
MYDOMAIN\win_bob &= UNIX_bob []
```

Another option is to write a rule that appends the UNIX account to the Windows account. The append operator adds information from one identity to another: OneFS appends the fields that the options specify from the source identity to the target identity. OneFS appends the identifiers to the additional groups list. Here is an example of an append rule with the groups option:

```
MYDOMAIN\win_bob ++ UNIX_bob [groups]
```

Getting the primary group from LDAP

By default, the user mapping service combines information from AD and LDAP but gives precedence to the information from AD. Mapping rules, however, can control how OneFS combines the information. You can, for example, retrieve the primary group information from LDAP instead of AD. The following mapping rule inserts the primary group information from LDAP into a user's access token:

```
*\* += * [group]
```

The following rule appends the other information from LDAP to a user's access token:

```
*\* ++ * [user,groups]
```

Denying access with the default UNIX user parameter

This scenario shows you how to deny access to a cluster if a user does not have an account in both Active Directory and NIS. The scenario includes two rules. The first rule replaces the primary group GID of an Active Directory user with the GID of the corresponding UNIX user in NIS. The second rule maps a user who does not have an account in NIS to a nonexistent user, thereby denying access to the cluster.

Here is how to add the first rule:

```
isi zone zones modify System --add-user-mapping-rules='*\* += * [group]'
```

After you set this rule, when a user connects to the cluster with an Active Directory account, OneFS looks up the user in NIS. When OneFS finds the user, it replaces the user's primary group GID from Active Directory with the GID of the UNIX user from NIS. By default, OneFS has already mapped the AD user account to its matching NIS user account.

Here is how to add the second rule. (When you set the rule from the command-line interface, the rule should appear on a single line.)

```
isi zone zones modify System --user-mapping-
rules='<default_unix_user=this-user-does-not-exist>'
```

This rule uses the default UNIX user parameter to map a user without both a matching UNIX user account and primary group to a nonexistent user with the following name:

```
this-user-does-not-exist
```

With such a rule in place, if a user does not exist in NIS, OneFS looks up the default_unix_user to obtain its GID. Since the default_unix_user maps to a nonexistent user, an attempt to look up the user fails, and as a result authentication also fails. The rule guarantees that primary UID and primary GID assignment either works correctly across both Active Directory and NIS or fails entirely.

Dumping a map of AD groups for LDAP

In a multiprotocol storage system with a permissions model that works with UNIX POSIX mode bits and Windows ACLs, permissions management becomes a nontrivial task. It requires analysis and planning to integrate Active Directory and LDAP with OneFS.

Before you set up your file shares, you should analyze and document the access requirements for your directory tree. If possible, avoid intertwining trees for different groups of users, such as engineering and marketing. You should also consider separating directory trees by protocol, creating a tree for your SMB users and another for your NFS users.

Adding applications to the requirements increases managerial complexity, and one way to simplify the complexity is to avoid mixing directory trees for NFS applications with trees for SMB applications or users. As a best practice, separate the directory trees for NFS applications from your SMB trees and from your NFS trees for users.

If you do mix directory trees for SMB users with those for NFS users, as most administrators do, try to integrate your Active Directory and LDAP systems before you set up your directory tree. In particular, you should strive to properly synchronize your LDAP groups with your Active Directory groups. OneFS automatically matches an Active Directory group with an LDAP group when the relative distinguished name of the group is the same.

One strategy is, of course, to use the user mapping service to combine groups from AD and LDAP into a single identity. But an advanced strategy to avoid a dependence on the mapping service is to export from Active Directory a map of your Active Directory groups, convert them to the LDAP Data Interchange Format (LDIF), and then ingest the LDIF contents into your LDAP server. The method is particularly useful if Active Directory is not using RFC 2307 attributes. Windows Server 2003 and Windows Server 2008, for instance, include a command-line tool named LDIFDE for exporting information in LDIF from Active Directory.

The result adds your independent group records from AD to LDAP, so that all your Linux and UNIX users receive the same GIDs and SIDs as your Windows users. It also gives you a chance to resolve GID collisions before they affect your Isilon cluster.

Conclusion

The OneFS user mapping service combines identities from different directory services. The mapping service unites a user's identities from LDAP, NIS, and Active Directory into a single access token that provides multiprotocol data access to files stored on an EMC Isilon cluster. Users who gain access to the cluster over SMB can access UNIX files stored over NFS with their LDAP accounts. Likewise, users who gain access to the cluster over NFS can access Windows files with their Active Directory accounts. As such, the user mapping service gives you the flexibility to solve identity problems in a NAS security context of multiprotocol data access governed by disparate directory services.

Appendix: Viewing mappings with the Platform API

The OneFS Platform API lets you view and modify your user mappings rules programmatically. You can also view the mappings for a user by identifying the user by username, UID, GID, or SID.

To connect to the OneFS Platform API, use a REST client, such as RESTClient, which you can add on to Mozilla Firefox for free, with the following settings for your HTTP requests:

1. Content-Type: application/json
2. Basic Authentication

You can view your user mapping rules in JSON with the following GET request:

```
https://192.0.2.2:8080/platform/1/auth/mapping/users/rules
```

You can also view the mapping for a user by specifying the user in the request. The following request displays the mapping for the root account in the system zone:

```
https://192.0.2.2:8080/platform/1/auth/mapping/users/lookup?user=root&zone=system
```

With a lookup request, you can append the following parameters to filter your queries: user, zone, uid, gid, sid, and primary_gid. Here is an example:

```
https://192.0.2.2:8080/platform/1/auth/mapping/users/lookup?uid=10000
```