Best Practices

Dell EMC Isilon OneFS: macOS Network Storage User Experience and Performance Optimizations

Abstract

This document describes performance and user-experience optimizations for Apple® macOS® 10.13+ with the Dell EMC™ Isilon™ OneFS operating system version 8.1.x.

April 2019

H17613.2

Revisions

Date	Description
February 2019	Initial release
March 2019	Minor updates
April 2019	Updates on .DS_Store behavior and flow control recommendations

Acknowledgements

This paper was produced by the following members of Dell EMC storage engineering:

Author: Gregory Shiff

Support: Simon Haywood, Rafal Szczesniak

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright © <pub date – rev date> Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners. [4/15/2019] [Best Practices] [H17613.2]

Table of contents

Executive summary

Many environments and workflows that feature Dell EMC™ Isilon™ systems as the primary shared storage platform are highly reliant on systems running Apple® macOS® operating system software. This guide provides performance-tuning and user-experience configurations for macOS 10.13+ with the Isilon OneFS operating system, version 8.1.x.

Audience

This guide assumes the reader is proficient with macOS, the macOS command-line interface, and Unix-style operating systems in general.

1 Introduction

This paper focuses on the SMB protocol with macOS 10.13+ and Isilon OneFS 8.1.2+. At the time of publication, macOS 10.14 (Mojave) is the current version of macOS.

Historically, many optimizations were necessary on macOS to achieve the most performance out of network storage. These included adjustments such as Send and Receive Windows, Window Scaling, Inflight Bandwidth Calculation, TCP Slow Start, and others. Fortunately, many such optimizations are no longer necessary due to improvements in how macOS dynamically auto-tunes its network performance. In many ways, management is now easier for macOS administrators, but it remains critical to understand what changes are possible and how they impact the way that macOS interacts with networked storage.

The following performance optimizations and user experience changes are covered in this document:

- Edits to /etc/sysctl.conf, including various settings of delayed_ack
- Settings in com.apple.desktopservices
  - UseBareEnumeration
  - DSDontWriteNetworkStores
- Edits to /etc/nsmb.conf
  - SMB signing
  - SMB session signing
  - Directory caching
  - SMB notifications
  - Force protocol version
  - Alternate data streams (named Streams)
  - Apple SMB extensions
- Jumbo Frames
- Memory allocation to network stack, including changes to default nvram boot-args

No single optimization is right for every environment. To take advantage of the tuning parameters outlined in this paper, testing is critical to determine how each change affects the performance of storage clients and their unique workloads.

2 Network storage protocols and macOS

Three network storage protocols are supported by macOS, to varying degrees: AFP (Apple protocol), SMB, and NFS. There is some support for FTP with macOS, but this is out of scope for this document.

AFP is unsupported by Isilon, and since macOS 10.9, AFP is not the default file-sharing protocol for macOS. As such, it is not covered in this document.

Historically, NFS was the protocol of choice for connecting macOS systems to Isilon. This is no longer the case. Testing described in this paper concluded that SMB is just as performant, and in many cases more performant, than NFS on macOS.

SMB is the protocol of choice for macOS clients connecting to Isilon storage. This paper shows that SMB consistently outperformed NFS on macOS 10.13+, and describes additional topics:

- Alternate data streams (named streams) support
  - Isilon has robust support for macOS resource fork data stored through alternate data streams over SMB.
  - OneFS does not support macOS clients writing resource fork data to alternate data streams over NFS.
- Full ACL support
  - SMB fully supports file system ACLs for fine-grained control of file access permissions.
  - OneFS supports NFSv4 ACLs which can be used on macOS, but these can introduce further complexity, especially in multi-protocol, multi-platform environments.
- Isilon support for Apple-specific extensions to the standard SMB protocol
  - The Apple SMB implementation (SMBX) has added a few macOS-specific features. Isilon provides support for these Apple-specific extensions to SMB.

SMB multichannel is a major advantage of SMB3. Isilon OneFS 7.1.1+ supports SMB3 multichannel by default. While macOS 10.10+ has support for SMB3, as the time of this publication, macOS does not support SMB3 multichannel. Thus, macOS users will see little to no performance differences between SMB2 and SMB3. If future releases of macOS support SMB3 multichannel functionality, Isilon users should see even greater throughput and fault tolerance.

See the following link for macOS SMB3 support: https://help.apple.com/deployment/macos/

3 Editing sysctl.conf

On macOS, the **/etc/sysctl.conf** file can be edited to change some of the default settings in the macOS kernel at boot time. There is also a CLI command, **sysctl**, that when run as root will make temporary changes to those same settings. To make the changes persist across reboots, editing the **/etc/sysctl.conf** file is required. If the file is not present on a system, a text editor can be used to create it.

3.1 TCP delayed acknowledgement (delayed_ack)

There are many explanations for TCP delayed acknowledgement (delayed_ack), such as the description in the article TCP Performance problems caused by interaction between Nagle's Algorithm and Delayed ACK. For most uses of TCP, delayed acknowledgement is a good thing and makes network communication more efficient. TCP acknowledgements are added to subsequent data packets. However, in practice on macOS, particularly with network connections lower than 1 GbE, TCP delayed acknowledgement can cause performance degradation.

There are four options in macOS for setting the characteristics of **delayed_ack**:

- delayed_ack=0: Responds after every packet (OFF)
- delayed_ack=1: Always employs delayed_ack; 6 packets can get 1 ack
- delayed_ack=2: Immediate ack after 2nd packet; 2 packets per ack (Compatibility Mode)
- delayed_ack=3: Should auto detect when to employ delayed ack; 3 packets per ack

The default setting for macOS clients is **delayed_ack=3**. However, testing has shown that in some environments — particularly where the client is reading and writing simultaneously (such as what happens when an application is rendering a video sequence) — changing the setting to **delayed_ack=0** significantly improves performance.

It should be noted that environments vary, and that the settings should be tested in each environment. Sometimes, a setting of **delayed_ack=1** or **delayed_ack=2** will work best. It is important to also understand the impact of the change on other parts of the system.

To query the current setting of the client, enter the following at the macOS command line:

```
$ sudo sysctl -a net.inet.tcp.delayed_ack
```

Example response:

```
net.inet.tcp.delayed_ack: 3
```

To change this setting, enter the following at the macOS command line:

```
$ sudo sysctl -w net.inet.tcp.delayed_ack=0
```

Example response:

```
net.inet.tcp.delayed_ack: 3 -> 0
```

To make the setting persist over a reboot, edit the **/etc/sysctl.conf** file on the macOS client:

- Create or edit the file at **/etc/sysctl.conf**
- Add the line **net.inet.tcp.delayed_ack=0**

4 Defaults and com.apple.desktopservices

The way macOS Finder treats directories accessed through SMB can be changed by using the CLI defaults command to update preferences in **com.apple.desktopservices**.

Apple has outlined a number of these optimization in the support article Adjust SMB browsing behavior in macOS High Sierra 10.13 and later.

Users must log out and log in again for changes to take effect.

4.1 .DS_Store files on network shares

The macOS Finder automatically creates **.DS_Store** files in every folder it accesses. This file stores metadata about how to display that directory's contents. The reading and writing of these files can slow down performance when listing the contents of directories with high file counts.

It is possible to prevent macOS from creating .DS_Store files on network shares in macOS 10.13. As of this writing, it is no longer possible to stop these files from being created on network shares in macOS 10.14. However, it is possible to prevent macOS 10.14 from reading the .DS_Store file prior to listing a directory's contents. In the absence of a .DS_Store file or if reading the .DS_Store file is suppressed, macOS will list the contents of a folder in alphanumeric order only upon initial open.

The following macOS CLI command prevents creation of .DS_Store files in macOS 10.13:

```
defaults write com.apple.desktopservices DSDontWriteNetworkStores -bool TRUE
```

After running this command, the user will need to log out and log in for the changes to take effect.

Running the above command has a slightly different effect in macOS 10.14. In 10.14, .DS_Store files are still created, but macOS will not read it prior to listing the contents of a directory, thus speeding up the listing of directories with high file counts.

For more information on .DS_Store files, see the .DS_Store Wikipedia article.

4.2 Gather directory metadata before displaying contents

The default behavior in macOS versions 10.12 and earlier was to always read a directory's complete metadata before displaying its contents. The advantage to a client system reading a directory's complete metadata before showing its contents is that file attributes will be more reliably displayed. The disadvantage is that reading this metadata can delay the listing of a directory's contents. To force macOS versions 10.13 and later to read a directory's metadata before displaying its contents, apply the following CLI command:

```
defaults write com.apple.desktopservices UseBareEnumeration -bool FALSE
```

After running this command, the user will need to log out and log in for the changes to take effect.

This setting may slow down the listing of a directory's contents. As always, testing is key.

5 Editing /etc/nsmb.conf

The **/etc/nsmb.conf** file is a plain text file that offers configuration settings for how macOS treats SMB shares. If this file is not present on a system, it will need to be created with a text editor. The /etc/nsmb.conf file is divided into sections as outlined on its man page. For these edits, defaults for all SMB shares need to be set. Therefore, the first line in the file should be **[default]** followed by the parameters and their values:

```
[default]
parameter=value
parameter2=value2
parameter3=value3
```

After making changes to /etc/nsmb.conf, unmount and remount SMB shares from the macOS system for changes to take effect.

5.1 SMB signing

Packet signing increases the security of SMB connections by helping prevent **man-in-the-middle** attacks. Essentially, a digital signature attached to each packet helps the client system confirm that data has not been tampered with while in transit. Packet signing is enabled by default in OneFS and is also the default behavior for SMB2 and SMB3 connections on macOS versions 10.11.5 through 10.13.3. SMB signing is off by default in versions 10.13.4 and later.

The extra overhead of packet signing can cause significant performance degradation on the latency-sensitive, high-performance workloads common to Isilon. Therefore, it is recommended that these workloads take place on private, secure networks, and that packet signing is disabled on the macOS client.

Apple outlines disabling SMB signing in the support article Turn off packet signing for SMB 2 and SMB 3 connections.

To disable SMB signing on macOS, add the following entry to **/etc/nsmb.conf:**

```
signing_required=no
```

After making changes to /etc/nsmb.conf, unmount and remount SMB shares from the macOS system for the changes to take effect.

5.2 SMB session signing

SMB3 introduces security enhancements that help prevent man-in-the-middle attacks during the **initiation** of an SMB client connection to an SMB server. This is different from **SMB signing** which adds a digital signature to each packet. SMB **session** signing can be described as a way to protect an SMB session from being tampering with as it commences. In certain scenarios, particularly where macOS client systems are bound anonymously to a directory server, this may cause authentication errors when a system tries to connect to an SMB share.

In situations where proper network credentials are not working from macOS systems running version 10.13+, disabling SMB session signing may resolve the issue. Similar to disabling SMB signing, this reduces the security of an SMB connection and is recommended on systems running on private, secure networks.

Apple describes session signing and how to disable it in the support article If you can't mount SMB share hosted by a Mac bound to Open Directory.

Add the following line to the **/etc/nsmb.conf** file:

```
validate_neg_off=yes
```

After making changes to /etc/nsmb.conf, unmount and remount SMB shares from the macOS system for the changes to take effect.

5.3 Directory caching

Local filesystem metadata about the contents of network shares is cached by macOS. The amount of metadata cached increases along with the total amount of installed system memory. This can cause problems in workflows where the contents of network shares change frequently and all client systems need to see the updates as they happen.

Apple has outlined how to disable this behavior at the bottom of the article, Adjust SMB browsing behavior in macOS High Sierra 10.13 and later.

To disable directory caching on macOS 10.13 and later, add the following entry to **/etc/nsmb.conf:**

```
dir_cache_off=yes
```

After making changes to /etc/nsmb.conf, unmount and remount SMB shares from the macOS system for the changes to take effect.

5.4 SMB change notifications

The SMB server provides macOS with updates or changes to mounted file shares. SMB notifications provided by Isilon provide the macOS client with this information in a OneFS environment. With a busy file share, this may result in the macOS Finder refreshing itself frequently and users may notice their file listings fluctuating or users may get booted out of folder that is being browsed. To avoid this behavior, it is possible to disable the Finder from requesting SMB change notifications.

Applying this setting can break workflows that require SMB notifications for folder listings to be current. **It can also lead to data corruption and other issues where multiple users are accessing the same files and directories.** Therefore, as with all macOS optimizations, great care and testing is required to make sure that enabling this setting does not cause workflow problems.

To disable SMB notification, add the following line to the **/etc/nsmb.conf** file:

```
notify_off=yes
```

After making changes to /etc/nsmb.conf, unmount and remount SMB shares from the macOS system for the changes to take effect.

5.5 Force SMB protocol version

Under certain circumstances, it may be desirable to force macOS to connect through a particular version of SMB. The following edits to **/etc/nsmb.conf** force particular SMB versions in macOS 10.12+. Note that macOS uses a binary bitmap to specify which version of SMB to use. Add **'protocol_vers_map='** and the appropriate value to force protocol types. See the following examples.

```
# Protocol version is specified using binary bitmap
# 1 => 0001 => SMB 1 only
# 2 => 0010 => SMB 2 only
# 3 => 0011 => SMB 1 or 2
# 4 => 0100 => SMB 3 only
# 6 => 0110 => SMB 2 or 3

# To Force SMB 3 only:
protocol_vers_map=4

# To Force SMB 2 or 3 only
protocol_vers_map=6
```

5.6 Enabling alternate data streams (named streams)

Isilon OneFS supports storing Apple resource fork data transparently in an alternate data stream, also called a named stream. This prevents the creation of **AppleDouble files**, increases macOS performance when writing to Isilon, and helps prevent file corruption.

To better understand this feature, refer to the following background information.

When macOS stores a file, that file is composed of two **forks**. There is a resource fork which contains extended attributes about the file itself, such as the file's icon. There is also a data fork which contains the file data itself. When macOS writes data to local file systems, such as HFS+ or APFS-formatted volumes, the resource fork data is stored in an alternate data stream and is not visible to users or systems accessing those files; only the file itself is shown with all of its associated metadata.

When macOS writes to a storage system that does **not** support alternate data streams, the resource fork data still needs to be stored somewhere. In this case, macOS stores the resource fork data in AppleDouble files. These are files beginning in **._** followed by the file name. For example, saving a file called **file.dat** results in two files being created on the file system: **file.dat** and **._file.dat**. This degrades performance because it increases the file operations on the target storage system. File corruption can also become an issue if a non-macOS system overwrites or deletes the AppleDouble files.

As mentioned previously in this section, OneFS supports storing resource fork data in an alternate data stream which prevents the creation of AppleDouble files. It is the default behavior of macOS 10.5+ to make use of an alternate data stream on an SMB server if the feature is available (as it is on Isilon). When writing through NFS to Isilon, macOS will revert to storing resource fork data as AppleDouble files.

Isilon intelligently manages resource fork data so even systems connected by NFS can interact with files created or used by SMB-connected macOS systems with no corruption of the resource fork data and no need for AppleDouble files.

While it is not necessary to force macOS to use alternate data streams (the default since macOS version 10.5), some administrators feel more comfortable explicitly enabling this feature in **/etc/nsmb.conf** by adding the following line:

```
streams=yes
```

The following archived support document outlines this functionality: [Mac OS X v10.5, v10.6: About named streams on SMB-mounted NAS, Mac OS X, and Windows servers; "-36" or "-50" alerts may appear]().

5.7 Apple SMB extensions

Apple has added several macOS-specific features to their implementation of SMB. Primarily, these features address the handling of file metadata stored in alternate data streams as described previously. Isilon supports these features:

ReadDirAttr: This feature changes how macOS handles reads of file metadata stored in alternate data stream when listing the contents of large directories. Finder info, access rights, and resource fork size are returned more efficiently for the files in the directory.

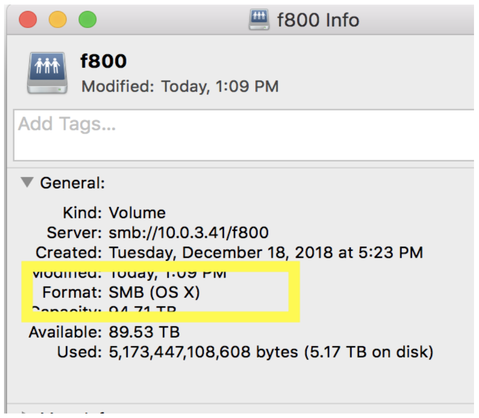
OsxCopyFile: With the SMB2 protocol, Microsoft implemented server-side optimizations when copying files between directories on the file share. The extension introduced by Apple ensures that all Apple-specific file metadata is properly copied along with the file itself. The copy process is also simplified as it is executed in just one request as opposed to splitting the requests into logical chunks which was the case in the original feature.

In versions 10.13+, macOS has changed how it interacts with SMB servers to pull directory file listings. Unintuitively, in workflows that have directories with high file counts, this can cause slower performance on SMB servers that support Apple-specific SMB extensions, such as OneFS.

It is possible to disable Apple-specific SMB features on macOS clients. While this may improve high-file-count directory listings, it means macOS clients can no longer take advantage of the benefits of these extensions such as better server-side file-copy performance. To disable the macOS client from making use of these extensions, add the following line to **/etc/nsmb.conf**:

```
aapl_off=true
```

The following screen captures show the share information before and after Apple extensions were disabled on the macOS client. Notice that before they are disabled, macOS identifies Isilon as a fully native file share, while afterwards, Isilon shows up as a more generic share.



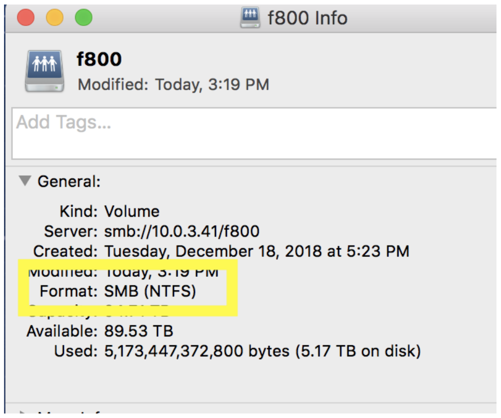
f800 Info

f800
Modified: Today, 1:09 PM

Add Tags...

General:

```
     Kind: Volume
   Server: smb://10.0.3.41/f800
  Created: Tuesday, December 18, 2018 at 5:23 PM
 Modified: Today, 1:09 PM
   Format: SMB (OS X)
 Capacity: 94.71 TB
Available: 89.53 TB
     Used: 5,173,447,108,608 bytes (5.17 TB on disk)
```



f800 Info

f800
Modified: Today, 3:19 PM

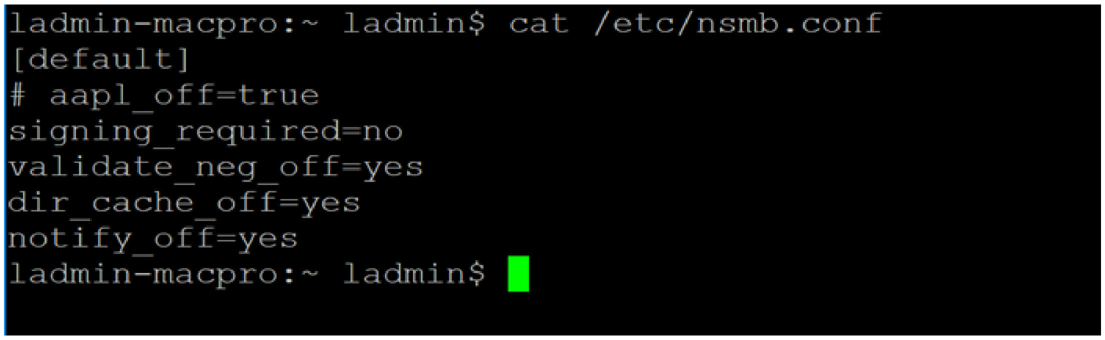
Add Tags...

General:

```
     Kind: Volume
   Server: smb://10.0.3.41/f800
  Created: Tuesday, December 18, 2018 at 5:23 PM
 Modified: Today, 3:19 PM
   Format: SMB (NTFS)
 Capacity: 94.71 TB
Available: 89.53 TB
     Used: 5,173,447,372,800 bytes (5.17 TB on disk)
```

5.8 Example /etc/nsmb.conf

The following shows an example of the **/etc/nsmb.conf** file with various setting enabled. Notice that the disabling of Apple-specific SMB extensions is present in the file but has been commented out.



```
ladmin-macpro:~ ladmin$ cat /etc/nsmb.conf
[default]
# aapl_off=true
signing_required=no
validate_neg_off=yes
dir_cache_off=yes
notify_off=yes
ladmin-macpro:~ ladmin$
```

6 Flow control, jumbo frames, and macOS

Using jumbo frames (9,000 MTU) can dramatically increase macOS SMB performance on network connections greater than 1 GbE. However, implementation of jumbo frames requires more than just setting a NIC port to use 9,000 MTU. The macOS client, network switch, and Isilon storage must all be properly configured.

Keep in mind that changes to network settings on the switch, clients, and storage will cause connections to drop. As such, these changes should not be undertaken during active production.

Starting with the network, if jumbo frames are to be used, the network on which the macOS client connects to the Isilon storage (and the network using 9,000 MTU) should be a dedicated production network and not used for non-storage traffic (such as email or internet browsing). Essentially, MTU sizes should be consistent on the network and that network should not be cluttered with non-storage related traffic.

The ports on the network switch itself must be configured to use jumbo frames, which means setting those ports to greater than 9,000 MTU. **On the switch used for testing in this document, the ports were set to 9,412 MTU**. Refer to the documentation for the specific switch in use for optimal port MTU settings**.** The extra overhead makes room for any additional data attached to the network packets, such as VLAN header information. In testing, if the switch ports were set to exactly 9,000 MTU, macOS clients hung when connecting to network shares, even on a flat network. Administrators should refer to their switching documentation for proper port configuration, since some switches may transparently add additional MTU to a port and others need to be explicitly configured.

Another critical setting for using macOS with jumbo frames is to enable **flow control** on the switch. Without flow control enabled, macOS jumbo frame performance through the switch was much lower than the standard setting of 1,500 MTU. Other platforms tested were not as sensitive to flow control; it made little to no difference in their performance but was required on macOS to achieve good performance using jumbo frames.

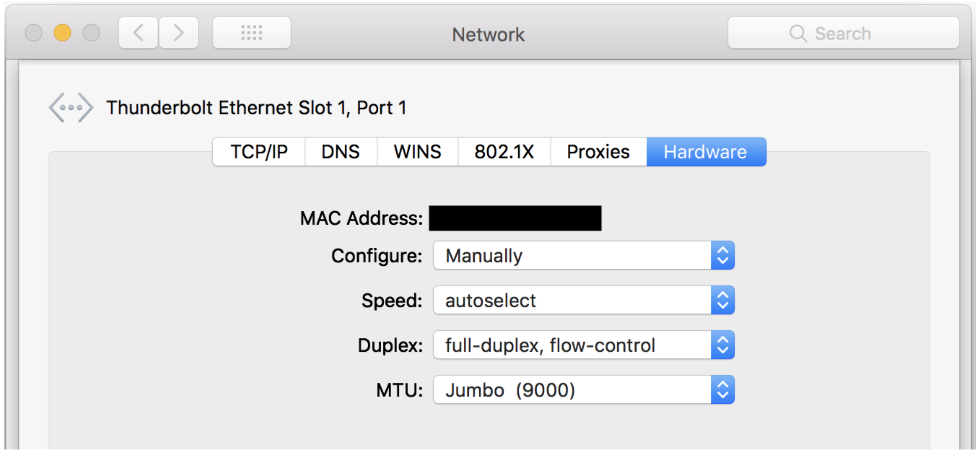
Enabling flow control on switch ports in use by macOS and Isilon is recommended, even when running at 1,500 MTU. Isilon storage is very fast and can overflow the macOS network buffers causing dropped packets and throughput degradation.

Once the ports are properly configured on the network switch, the external network ports on the Isilon storage must be set for 9,000 MTU. Refer to the OneFS documentation on the Isilon Info Hubs for setting the external network ports on the Isilon storage to 9,000 MTU.

Now that OneFS and the switch are set to jumbo frames, the NIC ports used by the macOS client can be configured. There are two settings that must be configured on the macOS client: **Jumbo (9000)** MTU and **Duplex**: **full-duplex, flow-control**. Setting Duplex to **full-duplex** with **no flow-control** resulted in poor performance, even when flow control was disabled on the switch. To realize the benefit of jumbo frames on macOS, flow control and MTU settings must be properly configured on both the client and the network switch.

In the macOS GUI, this can be configured in **System Settings > Network**. Select the appropriate NIC port, select **Advanced Settings**, and select the **Hardware** tab.

In the hardware tab, configure the port **Manually** and set the **Duplex** and **MTU** controls.



MAC Address: [redacted]
Configure: Manually
Speed: autoselect
Duplex: full-duplex, flow-control
MTU: Jumbo (9000)

Alternatively, network settings can be adjusted in macOS using the terminal command: **networksetup**

```
$networksetup -ListAllHardwarePorts
         This command will list all network ports available and their
         <DeviceIDs>

$networksetup -ListValidMedia <DeviceID>
   This command will show what options can be set for a particular <DeviceID>

$networksetup -SetMedia <DeviceID> autoselect full-duplex flow-control
   This example command sets <DeviceID> to auto-select speed with full-duplex
and flow-control

$networksetup -SetMtu <DeviceID> 9000
   This example command sets <DeviceID> to 9000mtu
```

7 Increasing network memory allocation

Increasing the amount of memory that macOS allocates for network buffering can increase throughput for network connections greater than 1 GbE.

Depending on the total system memory and the network connection type, various values can be set. 256 MB is a good starting point for 10 GbE and higher connections.

For macOS 10.13+, these commands need to be executed while booted into recovery mode. Apple describes the process of booting to recovery mode in the article Mac startup key combinations.

Once in recovery mode, launch the Terminal application and use the following commands to add a boot argument:

```
nvram boot-args="ncl=131072"
```

This setting is the number of 2 KB clusters assigned to the network buffer and is calculated as follows:

(131,072 * 2) / 1,024 = 256 MB

After setting this **nvram** boot argument, reboot the macOS system for the changes to take effect. Once rebooted, run the following terminal command to verify the boot argument:

```
nvram -xp
```

This displays current boot arguments in .xml format.

It is possible to set this value even higher, such as to 512 MB (NCL = 262144). Administrators need to be mindful of taking memory resources from the rest of the system and understand that at a certain point, allocating more memory to these buffers will have minimal returns in terms of added throughput performance.

To remove this **nvram** setting, run the following terminal command:

```
nvram -d boot-args
```

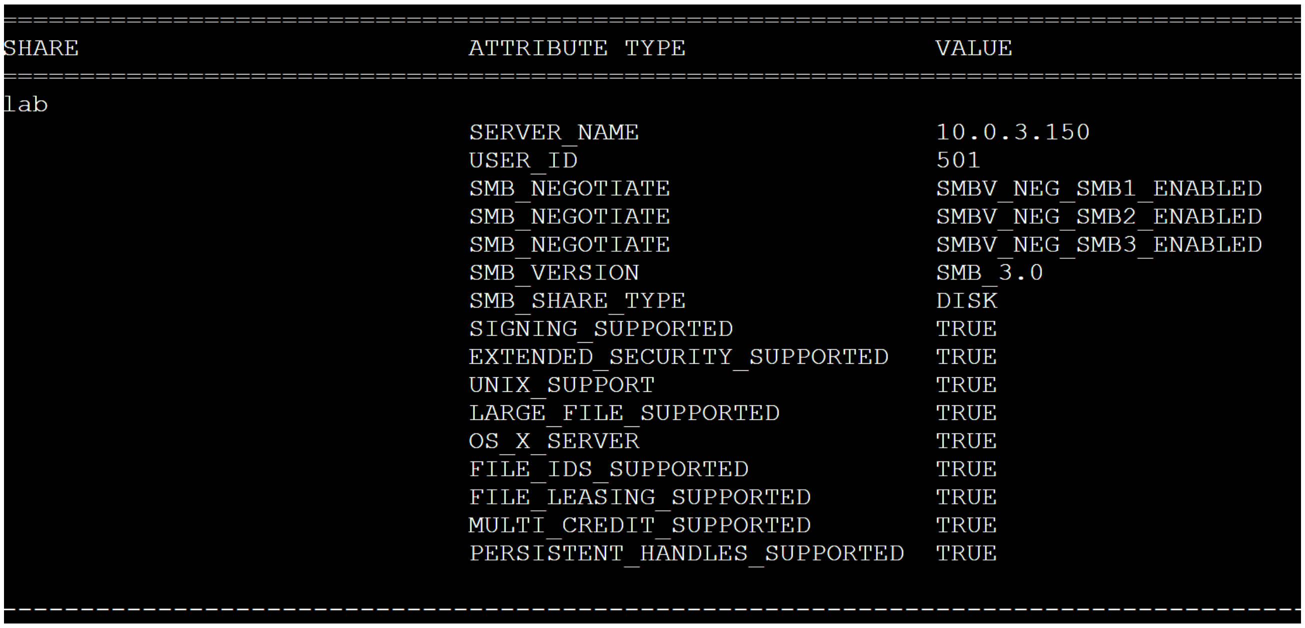
8 Examining SMB mounts with smbutil

The macOS CLI utility **smbutil** can provide useful information about what features are available and enabled on SMB-mounted file shares. The following command displays information about currently mounted SMB shares and their supported features:

```
smbutil statshares -a
```

Running this command can help administrators verify that edits made to **/etc/nsmb.conf** have taken effect as well as verify the SMB protocol version that is currently in use with that particular share.

The following shows sample output of this command on a macOS 10.13 system with no edits to its /etc/nsmb.conf file while connecting to an Isilon cluster running OneFS 8.1.0.2



```
==================================================================================================
SHARE                              ATTRIBUTE TYPE                     VALUE
==================================================================================================
lab
                                   SERVER_NAME                        10.0.3.150
                                   USER_ID                            501
                                   SMB_NEGOTIATE                      SMBV_NEG_SMB1_ENABLED
                                   SMB_NEGOTIATE                      SMBV_NEG_SMB2_ENABLED
                                   SMB_NEGOTIATE                      SMBV_NEG_SMB3_ENABLED
                                   SMB_VERSION                        SMB_3.0
                                   SMB_SHARE_TYPE                     DISK
                                   SIGNING_SUPPORTED                  TRUE
                                   EXTENDED_SECURITY_SUPPORTED        TRUE
                                   UNIX_SUPPORT                       TRUE
                                   LARGE_FILE_SUPPORTED               TRUE
                                   OS_X_SERVER                        TRUE
                                   FILE_IDS_SUPPORTED                 TRUE
                                   FILE_LEASING_SUPPORTED             TRUE
                                   MULTI_CREDIT_SUPPORTED             TRUE
                                   PERSISTENT_HANDLES_SUPPORTED       TRUE

--------------------------------------------------------------------------------------------------
```

Notice in the screenshot that **OX_X_SERVER** and **UNIX_SUPPORT** are both set to true. This is because OneFS is providing Apple SMB extensions. Also note that SMB1, SMB2, and SMB3 are all available as connection options, as indicated by the **SMB_NEGOTIATE** lines, but that the connection is through SMB3 as indicated by the **SMB_SHARE_TYPE** line.

9 Conclusion

The tight integration of Apple hardware and macOS remain a favorite of many professional users. As such, delivering the highest performance is a priority for macOS administrators. By understanding how macOS interacts with network storage, performance can be optimized for each environment.

Choosing correct network storage protocol, MTU settings, security features, and macOS Finder functionality are critical to this process. Apple provides a manageable mix of parameters to make meaningful changes in each category. As with any complex system, testing is critical to ensure that changes to the default behavior have the intended consequences.

Isilon treats macOS as a first-class citizen. By tuning client system interaction with the storage, users can take advantage of Isilon performance across diverse workflows.

A Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical documents and videos provide expertise that helps to ensure customer success on Dell EMC storage platforms.