

# Getting Started with



## Nas Permissions In OMV - A Primer

February 18th, 2021 - Rev 0.1

# Table of Contents

General:.....	3
1 An Immutable Law for Good Server Security	3
2 A Workstation Logon	3
3 NAS Share Access	4
4 Adding LAN Users to OMV	6
5 Shared Folder Permissions	8
6 Samba (SMB) Network Shares	9
7 ACL's - Extended Permissions	12
General.....	12
8 Permissions Overview	14
Practical Permissions Examples.....	14
A Media Share.....	15
A Group Share.....	15
A Restricted Share.....	16
A Personal / Private Share.....	17
9 The Bottom Line	18
10 Permissions Notes:	18

## General:

The purpose of this document is to provide an overview explanation of access control settings, using Samba network shares, in a peer-to-peer network. It's a brief explanation of permissions, as implemented within Openmediavault's GUI, with some usable examples. It does not apply, directly, to Domains or LDAP environments.

In Openmediavault's [New Users Guide](#), in the sections [Setting up a Shared Folder](#) and [Creating a SMB/CIF “Samba” Network Share](#), permission selections were made that will allow all local LAN users to connect to OMV server shares with **write** access. For home LAN's with one or two users, this may be adequate.

On the other hand, some home users may want to prevent children from deleting files and provision to allow guest login's with read only access. Further, small businesses may want to grant or restrict employee access to specific shares. These scenarios will require that permissions are implemented, for NAS share access control.

---

## An Immutable Law for Good Server Security

The password for the root account (the server's super user) should be strong and it should NOT be shared. While this may not be practical when operating a SOHO or business NAS server, the number of users who know the password for the root account should be held to a minimum. (In the business use case, there should be at least two administrators with root access.)

Openmediavault has another super user account “**admin**” that is used to log into the Web Admin GUI. Given that this user has “**root user like**” capabilities, **admin**'s password should not be shared either.

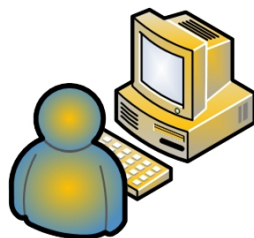
The reason why it is important to control who has access to the root and admin accounts and their passwords is, this level of access can be used to override or bypass all permissions discussed in this document.

---

## A Workstation Logon

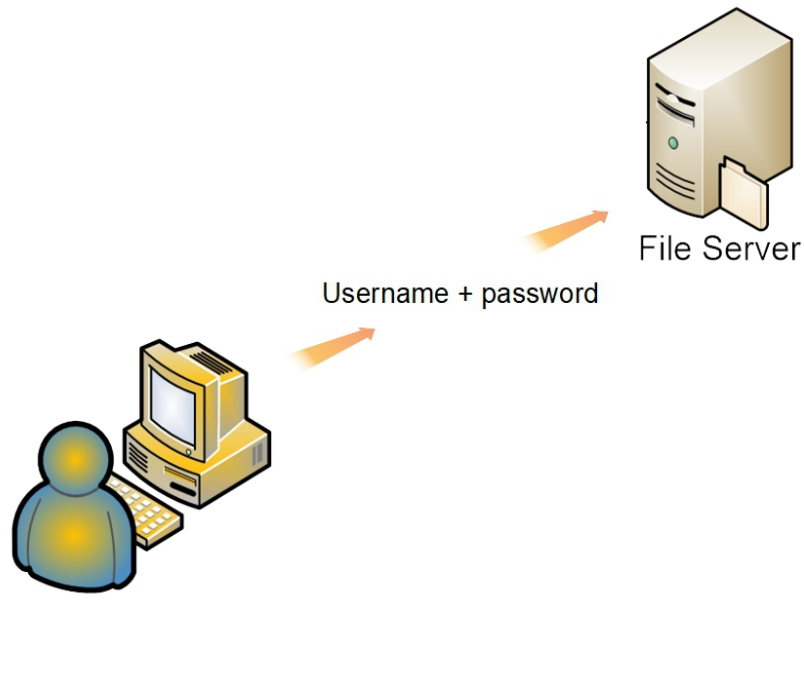
In most workgroup LAN environments, users log into their PC's using a unique username and password. These “credentials” are stored locally and have permissions associated with them that allow access to the local PC, it's folders and files, and other workstation resources. A username lookup is performed, the password is verified and if all match, access to the workstation is granted. A “local” server logon is very similar, allowing for local server administration.

Username + password

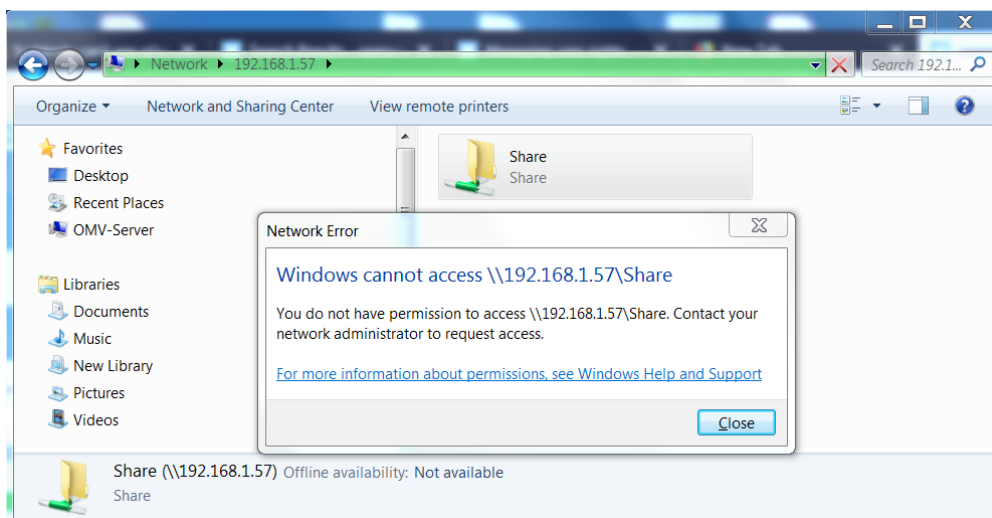


## NAS Share Access

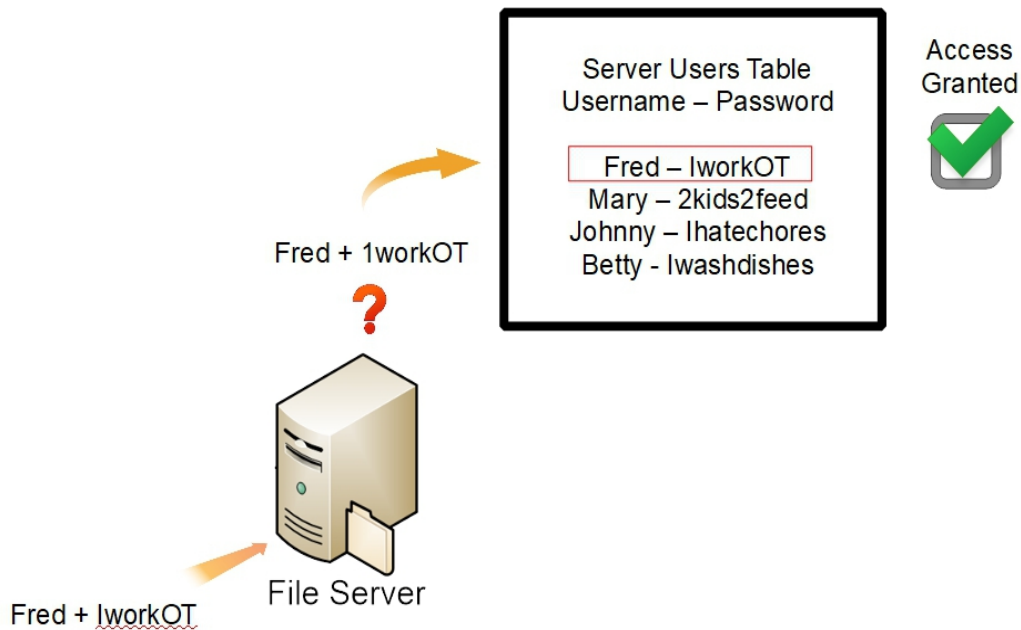
The same is true, indirectly, of network share access. When access to a network share is requested by a LAN client, there's a background authentication process taking place that is not visible to the user. When a user is logged onto a workstation, the workstation acts as an authentication “proxy”, offering the logged on user's credentials (username + password) to the server.



If the authentication process fails, the user may be prompted for alternate credentials (another username and password) or access is denied.



If the authentication process completes successfully, access is granted and the share is opened.



This is the basis for setting up transparent access for users, who are logged on at workstations, when they attempt to access NAS network shares.

---

In a workgroup environment, it's relatively easy to grant access to shares by **username** and **password**, but it requires some setup. As indicated in the above example, the Openmediavault server (hereafter referred to as OMV) needs to be aware of usernames and passwords for users who may attempt to access shares with permissions enabled.

(Continued)

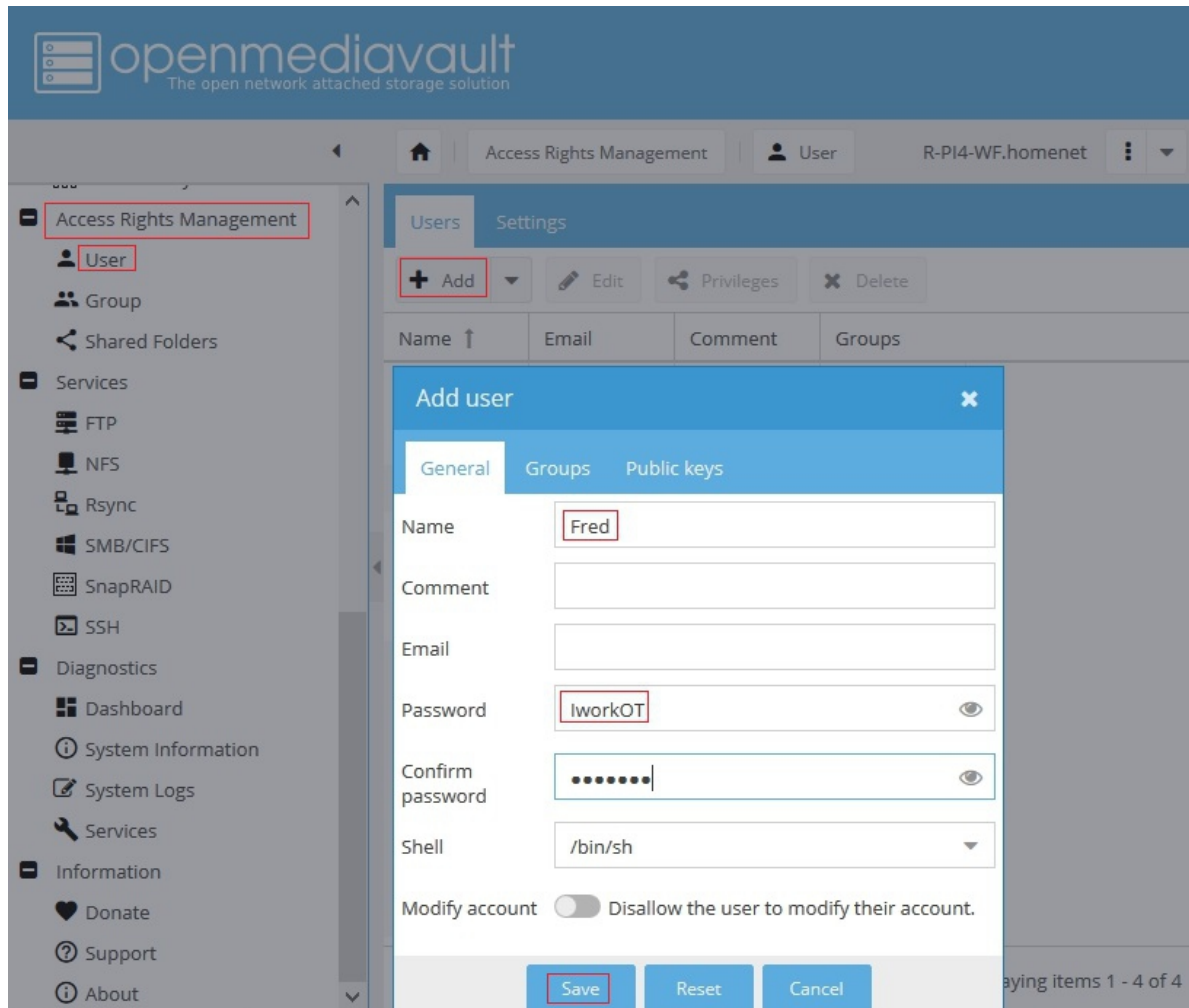
## Adding LAN Users to OMV

To enable transparent access, the first step is to add workstation usernames and their passwords to OMV.

Under, **Access Rights Management**, **User**, in the **Users** tab, click on the **+Add** button.

**Name:** Add the user name **exactly** as it is entered in the workstation logon, with capitalized letters if used.

**Password:** Add the username's password exactly as it is entered at the workstation.  
(In this example, the eye icon was used to show the password unmasked.)

The screenshot shows the OpenMediaVault (OMV) web interface. On the left sidebar, the 'Access Rights Management' menu is highlighted with a red box, and the 'User' sub-menu is also highlighted. In the main content area, the 'Users' tab is active, and the '+ Add' button is highlighted with a red box. An 'Add user' dialog box is open, showing the 'General' tab. The 'Name' field contains 'Fred', 'Comment' is empty, 'Email' is empty, 'Password' contains 'lworkOT', 'Confirm password' is masked with dots, and 'Shell' is set to '/bin/sh'. The 'Save' button at the bottom of the dialog is highlighted with a red box. The background shows the OMV logo and navigation tabs for 'Access Rights Management', 'User', and 'R-PI4-WF.homenet'.

**Save** the entry.

Repeat the process, adding all LAN users that will need access to the server's shares where permissions are applied.

The End Result

openmediavault  
The open network attached storage solution

Access Rights Management | User | R-PI4-

Access Rights Management

User

Group

Shared Folders

Services

FTP

NFS

Rsync

SMB/CIFS

SnapRAID

SSH

Diagnostics

Dashboard

System Information

System Logs

Services

Users Settings

+ Add Edit Privileges Delete

Name ↑	Email	Comment	Groups
Betty			users
Fred			users
Johnny			users
Mary			users
pi		""	pi, adm, dialout, cdrom, sudo, audio, video, plugdev, games, users, input, netdev, ssh, spi, i2c, gpio

All users have been entered in OMV, by the exact username and password they use to log into their workstations, laptops, etc. Notice that all usernames are in the Group **users** by default. (\*The **pi** user is a default system user installed on a Raspberry PI for administrative purposes. For the purpose of this document, the pi user can be ignored.\*)

(Continued)

## Shared Folder Permissions

By default, the majority of files and folders on the OMV file server are owned and accessed solely by the **root** user account. Since that is not useful in a networked environment, user access to a NAS server storage location is changed by the creation of a “**Shared Folder**”. Creating a shared folder is covered in the New User's Guide under [Setting up a Shared Folder](#). This process physically creates the folder and assigns usable permissions to the folder, that allow regular user access.

The default permissions assigned to a new Shared Folder, in OMV's GUI, are:

**Administrator: read/write,**

**Users: read/write,**

**Others: read-only**

These permissions directly correlate to the following:

The screenshot shows the 'Modify shared folder ACL' dialog. The 'Directory' is 'Test'. The 'User/Group permissions' table lists users Betty, Fred, Johnny, and Mary, each with checkboxes for Read/W..., Read-o..., and No acce... (likely Read-only, Read-only, and No access). The 'Extra options' section shows 'Owner' as 'root' with 'Read/Write/Execute' permissions, 'Group' as 'users' with 'Read/Write/Execute' permissions, and 'Others' with 'Read/Execute' permissions. The 'Replace' checkbox is checked, indicating 'Replace all existing permissions'.

Type	Name ↑	Read/W...	Read-o...	No acce...
User accounts				
Person icon	Betty	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person icon	Fred	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person icon	Johnny	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person icon	Mary	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Extra options

Owner: root | Read/Write/Execute  
Permissions of owner.

Group: users | Read/Write/Execute  
Permissions of group.

Others: Read/Execute  
Permissions of others (e.g. anonymous FTP users).

Replace: ☒ Replace all existing permissions

Apply Close

As previously noted and illustrated, all users are added to the Group **users** by default. In the example provided above, **Fred**, **Mary**, **Johnny** and **Betty** will be able to “**write**” to the folder “**Test**”.

**In the interests of clarity:**

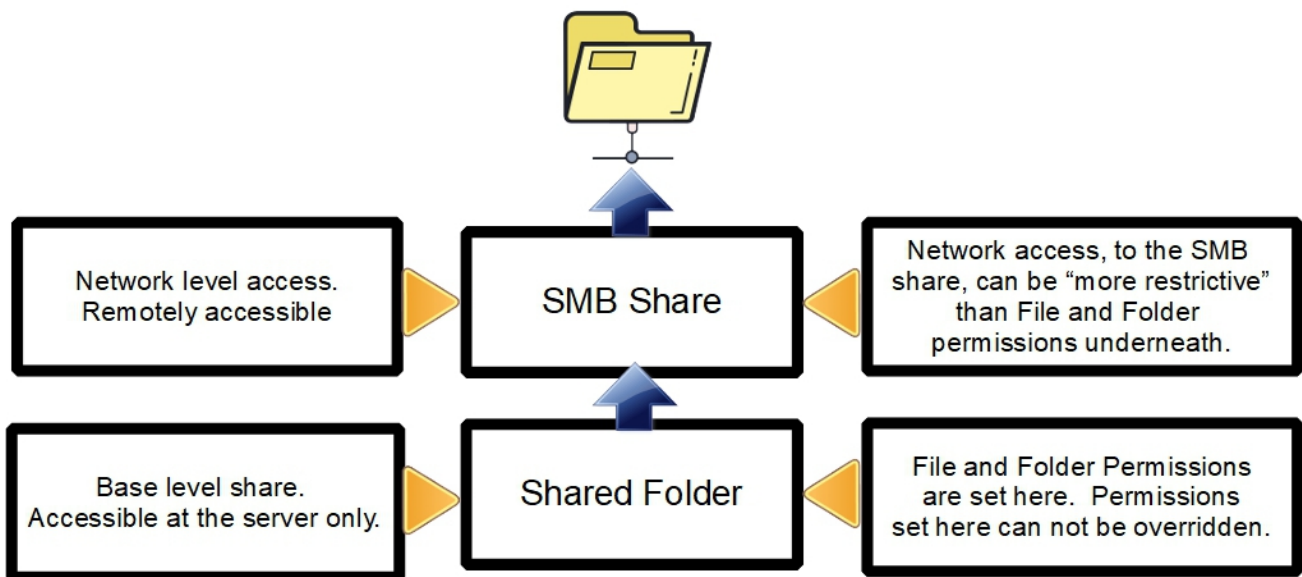
- What is shown as **Extra options** (above) are **standard Linux permissions**.
- For home server use and to keep server permissions simple, **use standard Linux permissions**.
- What is labeled as **User/Group permissions** (above) are **ACL's (Access Control List)**. **Do not mix ACL's** with standard Linux permissions, without understanding the *exact* effects.
- “**Others**” means any user that is not **root** or any user that is NOT in the Group **users**. This includes members of other Groups and **anonymous** login's. **Others**, in this example, have **Read**.



## Samba (SMB) Network Shares

While a **Shared Folder** is a “base” for sharing files, it is only one part of sharing data. A Shared Folder allows for local access, at the server, but it doesn't allow for network sharing. Network sharing requires a Samba share known as “**SMB/CIF**” in OMV's GUI.

(There are other network sharing techniques, such as **NFS** shares, that are not covered in this document.)



As noted in the illustration, a SMB share is layered onto a Shared folder to allow network access to LAN clients.

---

(Continued)

In the following; Samba, under Services, SMB/CIF, in the Settings tab is assumed to be Enabled.

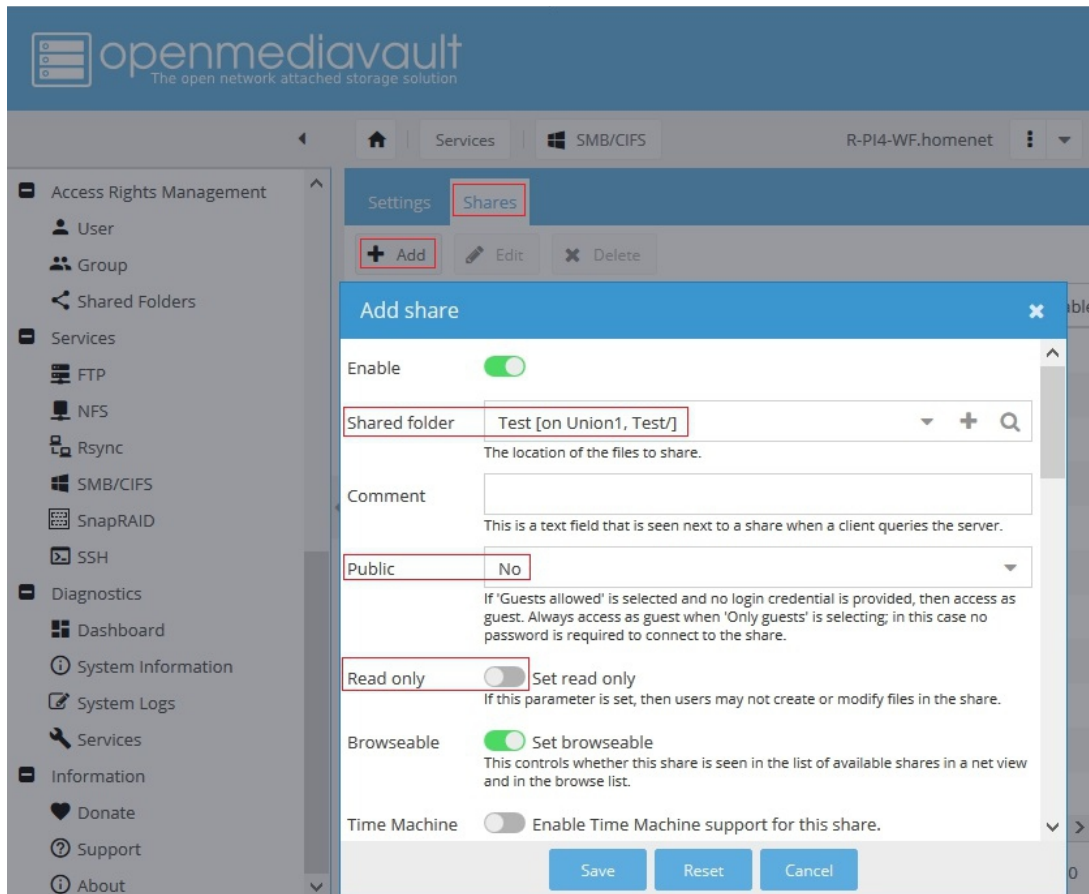
Under Services, SMB/CIF, in the Shares tab, click on the +Add button.

- **Shared Folder:**

In this case, we're layering a Samba network share on top of the “Test” Shared Folder, previously created.

- **Public:**

In this case, the entry selected is No.



- In the **Test** Shared Folder, we allowed **Others** “**Read**” access. The SMB (Samba) network share is layered on top of the “**Test**” Shared Folder. **Others** with **Read** access, in the Shared folder, equates to “**Guests Allowed**” in Samba. However, the SMB setting “**Public - No**” will stop anonymous or unknown users at the network share. This is what was meant by, “Samba can be more restrictive” than base level Shared Folder permissions.
- If the SMB **Public** field is set to “**Guests Allowed**”, that would combine with the “**Test**” Shared Folder permission, **Others** – **Read** to allow network guests **Read** access. (These permissions; **Others** – **Read** in the Shared Folder and **Guests Allowed** in Samba are appropriate for a media share. Network guests would have read access to media, music, movies, etc.)
- Beyond **Public** access choices, Samba assumes that appropriate user permissions have been assigned to the bottom layer, at the Shared Folder level.
- As shown below, if **Read only** is **ON** (green), **users** with **write** access to the Shared Folder, will not be able to add (write) or delete files. (There are exceptions. More on that later.)

Scroll to the bottom of the **Add share** dialog box, using the slider bar, on the right, or the down cursor key.

The **Hosts allow**'ed and **Hosts deny**'ed fields are workstation level access control options. While these options may fit some use cases, they can make permissions excessively “complicated” for some of the reasons following:

Understanding permissions effects, specifically the combination of various settings, is important. Again, Samba can further restrict but it can't override and “increase” access. Some examples:

- If a “host is allowed” but the username doesn't have access, the result is **denied**.
- If a host is denied but the username has access, the result is still **denied**.
- Consumer router behavior is not always consistent. If a host is specified by IP address, but the client uses DHCP, the IP address may change.
- Many consumer routers do not consistently map host names to IP address which may make “allow” or “deny” by host name inconsistent.

For these reasons and more, host entries should not be used without closely considering their effects.

**Add share**

or directory.

**Hosts allow**

This option is a comma, space, or tab delimited set of hosts which are permitted to access this share. You can specify the hosts by name or IP number. Leave this field empty to use default settings.

**Hosts deny**

This option is a comma, space, or tab delimited set of host which are NOT permitted to access this share. Where the lists conflict, the allow list takes precedence. In the event that it is necessary to deny all by default, use the keyword ALL (or the netmask 0.0.0.0/0) and then explicitly specify to the hosts allow parameter those hosts that should be permitted access. Leave this field empty to use default settings.

**Audit** ☐ Audit file operations.

**Extra options** \*

Please check the [manual page](#) for more details.

Save Reset Cancel

**Extra options:** This field presents home and small business administrators with some interesting options for share administration. For example, in the upper half of this Samba dialog box, there is the option for **Read only**. In a Samba share, the **Read only switch** will further restrict the group **users** to **read only** access, even if the Shared Folder below allows **write** access.

However a “**write list**” will allow an administrator to selectively bypass the Samba **Read only** switch. In this case if the statement **write list=Fred** is added to **Extra Options** field, the user **Fred** will have **write** access while the rest of the group users will still be restricted to **Read only**, enforced by Samba's **Read only switch**.

The same could be done for the entire users group with **write list=@users**. Adding this statement would allow the entire users group, over the network, **write** access while restricting **Others** with the **Read only switch**.

## ACL's - Extended Permissions

### General

Extended Permissions are not native to Linux. They are “add-on's” that are stored with a file or folder in their extended attributes. They are referred to as “Extended permissions” or “ACL's” (Access Control List), interchangeably. Extended permissions grant or deny file/folder access based on user or group “names”.

Again, note the following:

**Modify shared folder ACL**

Directory: Test

User/Group permissions		ACL's		
Type	Name ↑	Read/Write	Read-only	No access
User accounts				
Person icon	Betty	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person icon	Fred	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person icon	Johnny	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person icon	Mary	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Extra options** | **Standard Permissions**

Owner: root | Read/Write/Execute

Group: users | Read/Write/Execute

Others: Read/Execute

Replace: ☒ Replace all existing permissions

Recursive: ☐ Apply permissions to files and subfolders

Apply Close

Where possible, use **Standard Permissions** (labeled as **Extra Options**).

In the context of a NAS, used as a home server, ACL's should be avoided. Mixing Standard and Extended permissions can cause inexplicable effects, if not done carefully. However, ACL's can be used, if necessary, to explicitly “deny” access to one or more users in the Group **users**.

For instance, in the example group **users** we have two adults **Fred** and **Mary**, and their two children **Johnny** and **Betty**. It's easy to envision a scenario where adults may need a network share that their children couldn't access, that may contain medical information, letters to school officials, etc.

Following is a potential use of ACL's that would allow parents access to a share while denying their children access:

Type	Name ↑	Read/Write	Read-only	No access
<b>User accounts</b>				
Person	Betty	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Person	Fred	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person	Johnny	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Person	Mary	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Owner	root	Read/Write/Execute
Permissions of owner.		
Group	users	Read/Write/Execute
Permissions of group.		
Others	Read/Execute	
Permissions of others (e.g. anonymous FTP users).		
Replace	<input checked="" type="checkbox"/> Replace all existing permissions	
Recursive	<input checked="" type="checkbox"/> Apply permissions to files and subfolders	

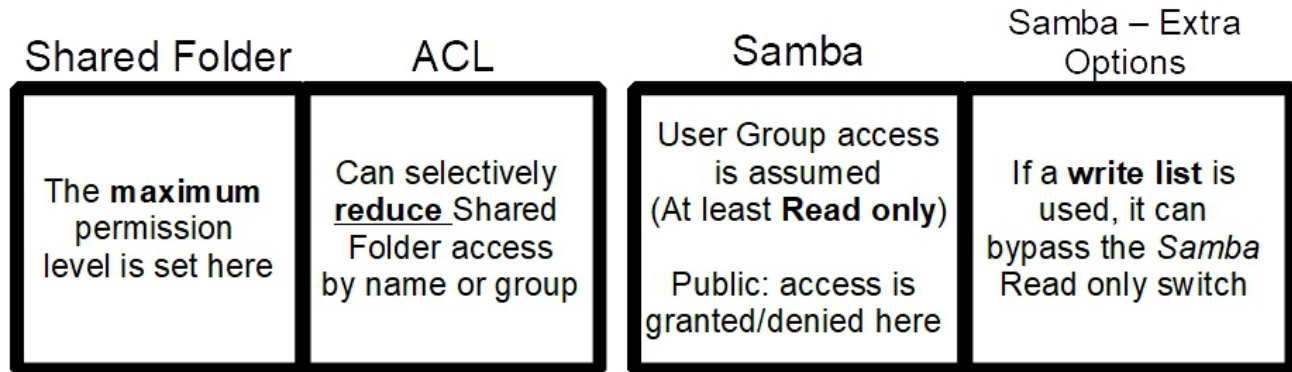
Apply Close

Note the check marks under **No access** for **Johnny** and **Betty**. To be sure that all files and folders in the share are reset with the appropriate permissions, the **Recursive** switch should be **ON** (green), before Clicking on **Apply**.

**Johnny** and **Betty** will have no access to the **Test** share, while the remaining users in the Group **users** will have **Write**. Using ACL's in this way allows a home administrator to selectively set individual users to **Read-only** or **deny access**. However, note that ACL's can not grant increased access that does not exist in Standard permissions.

## Permissions Overview

The following, moving left to right shows the hierarchy of Standard Linux permissions and the network permissions that are layered onto it, with Samba. Once Standard permissions are set in the Shared Folder, follow on permission layers can only *reduce* access. They cannot, for example, grant a user or a group **Write** access to a Shared Folder, if **Read only** is specified at the Shared Folder level.



### ***Practical Permissions Examples***

(In the following examples, root as the owner is assumed.)

In the examples, the list of users are as follows:

Fred – lworkOT

Mary – 2kids2feed

Johnny – lhatechores

Betty – lwashdishes

All users are in the Group **users**. Fred is the server admin.

---

(Continued)

## A Media Share

Shared Folder	<u>ACL's</u>	Samba	SMB Write List
Users: Read/Write Others: Read	Not Used	Public: Guests Allowed <b>Read Only is ON</b>	<b>write list=Fred</b>

- In the Shared Folder, the group **users** have **write**. This is necessary so that **Fred**, who is the family server administrator, can **write** to the share.
- Samba Public access is set to **Guests allowed** which works with the Shared Folder permission **Others: Read**. These permissions and Samba settings will allow visitors **read** access to media shares such as music or movies.
- **Read Only** is **ON**. This will further restrict the Group **users** down from **Write** to **Read only** access. With young children accessing a share, **Read only** is a good idea to prevent the possibility of an accidental deletion of media files.
- The Samba **Write list** bypasses the Samba **Read Only** setting for one user, allowing **Fred** to write to the share for admin purposes.

## A Group Share

(A location for sharing files among all family members or members of a group.)

Shared Folder	<u>ACL's</u>	Samba	SMB Write List
Users: Read/Write Others: Read	Not Used	Public: <b>NO</b> <b>Read Only is OFF</b>	<b>N/A</b>

- The Group **users** have **write**.
- While **Others** have **read**, SMB **Public** is set to “**NO**” which stops all users who are not in the Group **users**. Guests are not allowed. (The same effect, no Guest users, could be achieved at the shared folder level with **Others – None**.)
- **Read only** is **OFF** so Shared Folder permissions allow **all** members of the Group **users** to **write** to the share.



## A Restricted Share

This share is for private information, for select members of the Group **users**. ACL's can be used to remove access for users that should not see the contents of the applicable share. In this example, Parents have access while household children are set to **No Access**.

A significant point to be made about this example is that one or more users can be set to **Read only** or **No Access** without disturbing the access of the remaining members of the Group **users**. This might be convenient and expedient for employers who might want to restrict an employee to **No access** or **Read-only** access quickly, when “notice” has been given or received.

Shared Folder	ACL's	Samba	SMB Write List
Users: Read/Write Others: None	Johnny – No Access Betty – No Access	Public: <b>NO</b> Read Only is <b>OFF</b>	N/A

Shared Folder settings are as shown below:

Modify shared folder ACL

Directory: Test

Type	Name ↑	Read/Write	Read-only	No access
User accounts				
Person	Betty	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Person	Fred	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Person	Johnny	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Person	Mary	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Extra options

Owner: root (Permissions of owner: Read/Write/Execute)

Group: users (Permissions of group: Read/Write/Execute)

Others: None (Permissions of others (e.g. anonymous FTP users):)

Replace: ☒ Replace all existing permissions

Recursive: ☒ Apply permissions to files and subfolders

Apply Close

After selecting group usernames for **No access** (or **Read-only**) it's important to turn **Recursive ON** (green), before clicking the **Apply** button. This insures that new permissions are written to all files and folders within the share.



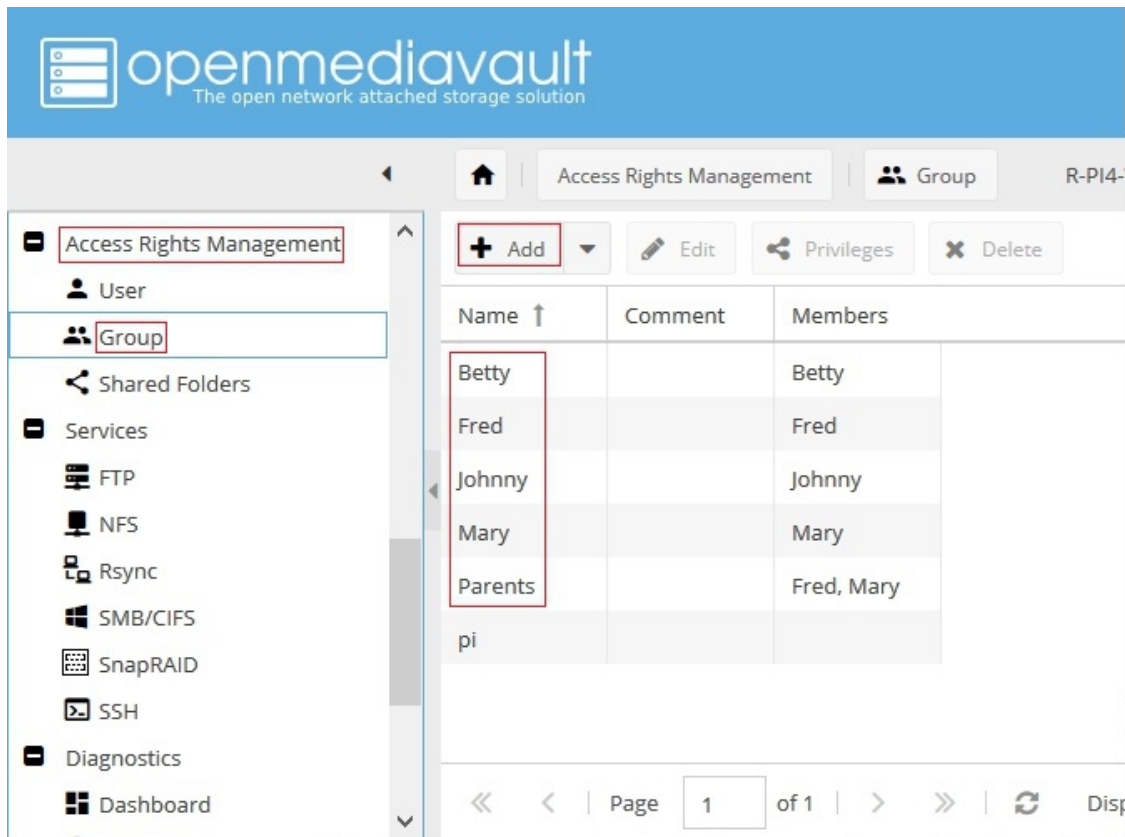
### Note:

The above could also be achieved by creating a new group created under; **Access Rights Management, Group** and clicking on the **+Add** button. A group named **parents** could contain the users **Fred** and **Mary**. If the Group **parents** is used above, in the **Group** field, ACL entries to deny access to children would not be required.

## A Personal / Private Share

A private share for an individual user could be created using ACL's and setting all users, but one, to **No access**. However, creating a group with one user might be the best approach.

Note the names of the newly create groups, below, and the usernames in each group. The naming scheme keeps it simple.



In this case, the Group **Fred** (with a single user **Fred**) has **Write**. The Samba settings noted are appropriate for this type of share. Only **Fred** can access and **write** this share.

Shared Folder	ACL's	Samba	SMB Write List
Fred: Read/Write Others: None	N/A	Public: NO Read Only is OFF	N/A

## The Bottom Line

If all data is stored in a single share, assigning appropriate permissions might range from difficult to impossible. On the other hand, if careful thought is given to segregating data into logical sets (shared folders) with user access and permissions in mind, assigning the appropriate permissions becomes a much easier task.

## Permissions Notes:

- Additions of new users or changes to existing user accounts, such as password changes, will have to be replicated at the server.
- Some use cases may benefit from using the [Credential Manager](#) built into Win10.
- Win10 workstations may need some network configuration changes to access a server in a peer-to-peer network. If a workstation can't connect to an OMV server, see this [document](#) for settings and work arounds.