SMB3.1.1 POSIX Protocol Extensions: Summary and Current Implementation Status

Steve French
Azure Storage – Microsoft

Jeremy Allison
Google
Legal Statement

- This work represents the views of the author(s) and does not necessarily reflect the views of Microsoft or Google.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.
Outline

- What is POSIX?
- Why do these extensions matter?
- Demo
- What if we don't have them?
  - What works?
  - Some history: CIFS Extensions
  - Alternatives
- Some details
- What if Linux continues to extend, to improve?
POSIX != Linux
(Linux API is much bigger)
Linux is BIG

- Currently 293 Linux syscalls!
- vs
- About 100 POSIX API calls
Motivations for Extensions

• Linux Apps work!
  - Case sensitivity e.g. is required for the kernel to build on Linux
  - (And Linux and other posix-like operating systems want posix behavior for files whether on premise or in cloud)

• Improve common situations where customers have Linux and Windows and Mac clients accessing the same data

• Deprecation of CIFS – make sure extensions work with most secure, most optimal SMB3.1.1 dialect
What could you try today?

• For obvious reasons these experimental changes not enabled by default:
  - With current mainline Linux (4.18 or later) must mount with “vers=3.1.1”
    AND also specify new mount option “posix” and turn off remapping of reserved characters (ie append “nomapposix”)
  - Only a few limited protocol features (posix open context request) can be tried but this small change VERY useful … enough to experiment with and test various apps

• JRA has a tree on samba.org (git.samba.org/jra/samba/.git in branch “master-smb2”) with prototype server code

• Other vendors testing experimental distinct implementations of POSIX extensions as well this week
Example

- On the client:
  - "mount -t smb3 //<address>/<share> /mnt -o username=<user>,password=<pass>, vers=3.1.1,posix,mfsymlinks,nomapposix,noperm"

- On the server add to smb.conf
  - "mangled names = no"
  - "directory mask = 07777"
  - "create mask = 07777"
Note the new mount option “posix” (vs “nounix”)
Mode bits on create and case sensitivity work!
Mode bits on mkdir works!

root@smf-Thinpad-P51:~/cifs-2.6# mount -t smb3 //127.0.0.1/scratch /mnt -o username=testuser,vers=3.11,posix
root@smf-Thinpad-P51:~/cifs-2.6# umask 0000
root@smf-Thinpad-P51:~/cifs-2.6# mkdir /mnt/0774 -m 0774
root@smf-Thinpad-P51:~/cifs-2.6# mkdir /mnt/0770 -m 0770
root@smf-Thinpad-P51:~/cifs-2.6# mkdir /mnt/0444 -m 0444
root@smf-Thinpad-P51:~/cifs-2.6# ls /scratch -la

```
total 20
drwxrwxrwx  5 root root  4096 Jun 15 20:42 ·
drwxr-xr-x  35 root root  4096 Jun 15 20:39 ..
dr--r--r--  2 testuser testuser 4096 Jun 15 20:42 0444
drwxrwxrwx  2 testuser testuser 4096 Jun 15 20:42 0774
drwxrwxrwx  2 testuser testuser 4096 Jun 15 20:42 0774
root@smf-Thinpad-P51:~/cifs-2.6# cat /proc/version
Linux version 4.17.0+ (sfrench@smf-Thinpad-P51) (gcc version 7.3.0 (Ubuntu 7.3.0-16ubuntu3))
```
Rename works with POSIX extensions!
Statfs (“stat –f”) without POSIX extensions:
Statfs ("stat –f") with POSIX extensions – works!

```
root@smf-Thinkpad-P51:~/cifs-2.6# cat /proc/mounts | grep smb3
//127.0.0.1/scratch /mnt1 smb3 rw,relatime,vers=3.1.1,cache=strict,username=testuser,ino=,uid=0,noforceuid,gid=0,noforcegid,addr=127.0.0.1,file_mode=0755,dir_mode=0755,squash,delete_if_noatime,hardlinks,serverino,mappossix,nopreserve,rsize=1048576,wsize=1048576,echo_interval=0,imeo=0 0
root@smf-Thinkpad-P51:~/cifs-2.6# stat -f /mnt1
File: "/mnt1"
   ID: 0     Name.len: 4096     Type: smb2
  Block size: 4096  Fundamental block size: 4096
 Blocks: Total: 58701044   Free: 10080249    Available: 7080966
 Inodes: Total: 14983168  Free: 13901538
root@smf-Thinkpad-P51:~/cifs-2.6# stat -f /scratch
File: "/scratch"
   ID: e94471edc7140504 Name.len: 255     Type: ext2/ext3
  Block size: 4096  Fundamental block size: 4096
 Blocks: Total: 58701044   Free: 10080127    Available: 7080844
 Inodes: Total: 14983168  Free: 13901536
```
Details – Negotiate Request (w/POSIX)

Filter: smb2

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Protocol Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>2.073582637</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>SMB2</td>
<td>254</td>
<td>Negotiate Protocol Request</td>
</tr>
<tr>
<td>23</td>
<td>2.080103274</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>SMB2</td>
<td>338</td>
<td>Negotiate Protocol Response</td>
</tr>
<tr>
<td>25</td>
<td>2.080184273</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>SMB2</td>
<td>199</td>
<td>Session Setup Request, NTLMSSP</td>
</tr>
<tr>
<td>26</td>
<td>2.080380294</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>SMB2</td>
<td>360</td>
<td>Session Setup Response, Error:</td>
</tr>
</tbody>
</table>

**NegotiateContextCount:** 3

Reserved: 0000

Dialect: 0x0311

- Negotiate Context: SMB2 PREAUTH INTEGRITY CAPABILITIES
- Negotiate Context: SMB2 ENCRYPTION CAPABILITIES
- Negotiate Context: Unknown Type: (0x100)

**Type:** Unknown (0x0100)

DataLength: 8

Reserved: 00000000

Unknown: 0000000000000000
Details (continued) – Neg response
Details continued – Create (POSIX) req
Details continued – create response
Summary of What works

• Without Extensions

• With Extensions
Other Alternatives: AAPL
Note that Apple create context (AAPL) can be used for some of this
And the response:
CIFS Unix/POSIX Extensions

• What was wrong with what we had?
  - Remember CIFS Deprecation?
  - And not just due to WannaCry …
    • SMB3 is really good …

• Apple SMB2/SMB3 create context does handle case sensitivity, but not all POSIX compatibility issues
Client Perspective

• What about the Linux Kernel?
  - What does it really need from SMB3 to be optimal…?
  - Not just to do 'cool' things: compile kernel on SMB3 mount, boot linux (show blazing performance …!)
  - For all key features: SMB3 >= CIFS with/Unix Extensions
    • We are not asking user to go backwards
  - Can we extend them as Linux API moves
    • (Did we mention that mount API and fsinfo/statfs BOTH are changing – see Al Viro’s git tree … and that statx was added last year and Linux continues to evolve …)
The challenges of Create/Rename/Delete
The challenges of POSIX inode metadata

• What do we need to be able to return?
• What about mode bits and ACLs?
The Challenges of POSIX locking
The Challenges of POSIX FS info
Remember JRA’s Server Perspective?

• Learn from the mistakes of SMB1 Unix extensions.
  - Security issues paramount.
  - Remove the possibility of server-followed symlinks
    • Break interoperability with NFS :-(, but necessary.

• Minimum Necessary Change (with apologies to Asimov’s “The End of Eternity”).
  - Fewer changes to the protocol the better.
  - Use the fact that we have experience with Samba in sharing between Windows and UNIX SMB connections.
Server Perspective Continued..

• Server-followed symlinks that the client can create have been a security disaster in Samba.

• Server-following symlinks is a useful holdover from ancient times, when admin-created symlinks gave great flexibility to setups.
  - As soon as clients gained the ability via UNIX extensions to create symlinks, disaster strikes.
  - Failed design decision to store these as real symlinks on the server filesystem.
    • Convenience for dual NFS / SMB1 servers.

• **THIS MUST NOT BE ALLOWED FOR SMB2+**
Server Perspective Continued..

- The key for SMB2 UNIX extensions is to allow simultaneous Windows and UNIX handles – using SMB2 create contexts.
  - Adding UNIX extension create context turns on POSIX behavior for this handle only.
  - Allows client code to probe for POSIX behavior – SMB2 specifies unknown create contexts are ignored.
  - The Samba server already has to handle this case in serving POSIX and non-POSIX client simultaneously.

- Leads to new Negotiate context requirement from the server.
  - That way a client can determine if a server could support POSIX behavior on a handle, but choses not to.
  - POSIX servers may expose POSIX behaviors or deny them depending on pathname (crossing mount points).
Server Perspective Continued..

- The rest of the changes are relatively small.
- One new info level needed to cope with POSIX stat returns.
- Keep protocol as close to “native” Windows as possible.
  - Map POSIX ‘mode’ into Windows ACL encoding.
  - No POSIX ACLs – return everything as Windows ACLs.
  - No POSIX uid/gids – return everything as Windows SIDs.
    * Client systems must cope with mapping SIDs anyway.
- Filename handling (POSIX specific, case sensitive) is the largest change. No access to Windows streams.
  - If you want a Windows stream handle, open a Windows stream handle.
  - Keep USC2 encoding (no change from Windows). UTF-8 would be nice, but not strictly required so drop it.
- Allow server to associate modified behavior on a per-handle basis.
Details
Proposed SMB3 POSIX Extensions

• Negotiate Protocol
  - SMB3.1.1 (or later required)
    • POSIX Negotiate Context 0x100
    • Version is implied by the context (in case extensions are revised in the future to a version 2 or 3 ...) but there is a reserved field that can be used in emergency
  - If POSIX open contexts not supported, negotiate context must be ignored
  - If POSIX open contexts supported for some files then negotiate context is returned, but server must fail opens with POSIX contexts for files where POSIX is not supported (rather than ignoring the POSIX context)

• Tree Connect – in future dialects tree connect contexts may allow more granularity in allowing servers to tell clients which shares they can't use POSIX opens on

• Case sensitivity yes/no can be exposed via existing QFS Info call
POSIX Extension Requirements

• If server returns a POSIX create context on an open:
  - It supports case sensitive names on this path
  - It supports POSIX unlink/rename semantics on this file
  - It supports advisory (POSIX) locking on this file.
    • Actually they are “OFD” not “POSIX” locks (see e.g. [link](https://gavv.github.io/blog/file-locks/#emulating-open-file-description-locks))
  - **NEED TO VERIFY**: PATH names are not remapped (no SFU remap needed for * and \ and > and < and : …). UCS2 converted directly to UTF-8 and server supports POSIX pathnames
We Leverage Existing SMB3 features

- Hardlinks use Windows setinfo call (long ago implemented)
- Symlinks, mkfifo, mknod use “nfs reparse point” (MS-FSCC 2.1.2.6)
- ACE with special SID (with mode at end) ala “NFS ACL” mapping can be used to set mode (SID: S-1-5-88-3) see http://people.redhat.com/steved/Bakeathon-2010/SDC2010-NFS-Windows.jbiseda.20100921.pdf
- Other linux extensions, e.g. fallocate are mapped to existing SMB3 operations where possible
Proposed POSIX Extensions

• Create/Open
  - New POSIX create context
    • If POSIX supported then context must be returned on all opens for which POSIX create context was sent (or open should be failed)
    • It is allowed to have POSIX and non-POSIX opens on the same file
    • It is allowed to have some files in a server which are POSIX and some which are not
POSIX open/create context resp.
SMB2/SMB3 Create Contexts

We define a new context name for this new CreateContext to distinguish it from others like MxAC and RqLs and a buffer to include POSIX Information in request and response

`SMB2_CREATE_TAG_POSIX = "\x93\xAD\x25\x50\x9C\xB4\x11\xE7\xB4\x23\x83\xDE\x96\x8B\xCD\x7C"`
Proposed POSIX Infolevels

- Query/SetInfo and Query_DIR
  - Level 0x64  SMB2_FIND_POSIX_INFORMATION
  - Payload variable (Max = 216 bytes)
    - Timestamps
    - File size
    - Dos attributes
    - U64 Inode number
    - U32 device id
    - U32 zero
    - Struct posix_create_context_response
Also need to support statfs ("stat -f")
currently using FS_INFO level 100

```c
+struct posix_v1_query_fs_info_response {
    /* Returned for context SMB2_POSIX_V1_STATFS_INFO */
    /* EXISTING posix extensions for fs info is good enough, note For undefined
    recommended transfer size return -1 in that field */
    __le32 OptimalTransferSize;  /* bsize on some os, iosize on other os */
    __le32 BlockSize; /* f_frsize, disk bytes avail based on this size */
    /* Next three fields are in terms of the block size above. If block size unknown, 4096 would
    be reasonable block size for a server to report. Note that returning blocks/blocksavail
    removes need to make second call (to QFSInfo level 0x103. UserBlockAvail is typically less
    than or equal to BlocksAvail, if no distinction is made return the same value in each */
    __le64 TotalBlocks;
    __le64 BlocksAvail; /* bfree */
    __le64 UserBlocksAvail; /* bavail */
    __le64 TotalFileNodes;
    __le64 FreeFileNodes;
    __le64 FileSysIdentifier; /* fsid */
    /* NB Namelen comes from FILE_SYSTEM_ATTRIBUTE_INFO call, and flags can come from
    FILE_SYSTEM_DEVICE_INFO call */
    /* In Linux f_type is always 0xFE 'S' 'M' 'B' since that is the fs, not the server's os - so
    server does not have to return it */
```
Wireshark

- See Aurelien’s dissector improvements
  - https://github.com/aaptel/wireshark/commits/smb3unix
  - And Pike sample test code
    - https://github.com/aaptel/pike/tree/smb3unix
POSIX Extensions – Where do we go from here?

- Continue debugging test implementations (cifs.ko and JRAs Samba POSIX test branch). Current focus on enhancing readdir this week
- Continue extending the wireshark dissectors (see Aurelien)
- Continue testing/prototyping here and additional testing in SMB3 plugfest here
- Continue updating the wiki with details: https://wiki.samba.org/index.php/SMB3-Linux
- Also questions/comments on samba-technical and linux-cifs mailing lists are welcome
Thank you for your time

- This is a very exciting time for ...