CAPABILITIES IN CEPHFS

When a client wants to operate on an inode, it will query the MDS in various ways, which will then grant the client a set of **capabilities**. These grant the client permissions to operate on the inode in various ways. One of the major differences from other network filesystems (e.g NFS or SMB) is that the capabilities granted are quite granular, and it’s possible that multiple clients can hold different capabilities on the same inodes.

TYPES OF CAPABILITIES

There are several “generic” capability bits. These denote what sort of ability the capability grants.

```
/* generic cap bits */
#define CEPH_CAP_GSHARED     1  /* client can reads (s) */
#define CEPH_CAP_GEXCL       2  /* client can read and update (x) */
#define CEPH_CAP_GCACHE      4  /* (file) client can cache reads (c) */
#define CEPH_CAP_GRD         8  /* (file) client can read (r) */
#define CEPH_CAP_GWR        16  /* (file) client can write (w) */
#define CEPH_CAP_GBUFFER    32  /* (file) client can buffer writes (b) */
#define CEPH_CAP_GWREXTEND  64  /* (file) client can extend EOF (a) */
#define CEPH_CAP_GLAZYIO   128  /* (file) client can perform lazy io (l) */
```

These are then shifted by a particular number of bits. These denote a part of the inode’s data or metadata on which the capability is being granted:

```
/* per-lock shift */
#define CEPH_CAP_SAUTH      2 /* A */
#define CEPH_CAP_SLINK      4 /* L */
#define CEPH_CAP_SXATTR     6 /* X */
#define CEPH_CAP_SFILE      8 /* F */
```

Only certain generic cap types are ever granted for some of those “shifts”, however. In particular, only the FILE shift ever has more than the first two bits.

```
| AUTH | LINK | XATTR | FILE
2      4      6       8
```

From the above, we get a number of constants, that are generated by taking each bit value and shifting to the correct bit in the word:

```
#define CEPH_CAP_AUTH_SHARED  (CEPH_CAP_GSHARED  << CEPH_CAP_SAUTH)
```

These bits can then be or’ed together to make a bitmask denoting a set of capabilities.

There is one exception:

```
#define CEPH_CAP_PIN  1  /* no specific capabilities beyond the pin */
```

The “pin” just pins the inode into memory, without granting any other caps.

Graphically:

```
+---+---+---+---+---+---+---+---+
| p | _ |As   x |Ls   x |Xs   x |
+---+---+---+---+---+---+---+---+
|Fs   x   c   r   w   b   a   l |
+---+---+---+---+---+---+---+---+
```

The second bit is currently unused.

ABILITIES GRANTED BY EACH CAP:

While that is how capabilities are granted (and communicated), the important bit is what they actually allow the client to do:

- PIN: this just pins the inode into memory. This is sufficient to allow the client to get to the inode number, as well as other immutable things like major or minor numbers in a device inode, or symlink contents.
- AUTH: this grants the ability to get to the authentication-related metadata. In particular, the owner, group and mode. Note that doing a full permission check may require getting at ACLs as well, which are stored in xattrs.
- LINK: the link count of the inode
- XATTR: ability to access or manipulate xattrs. Note that since ACLs are stored in xattrs, it’s also sometimes necessary to access them when checking permissions.
- FILE: this is the big one. These allow the client to access and manipulate file data. It also covers certain metadata relating to file data – the size, mtime, atime and ctime, in particular.

SHORTHAND:

Note that the client logging can also present a compact representation of the capabilities. For example:

::

```
pAsLsXsFs
```

The ‘p’ represents the pin. Each capital letter corresponds to the shift values, and the lowercase letters after each shift are for the actual capabilities granted in each shift.