SAM-QFS x.x Planning - Functional Specification Document

==========

Updated: 04/08/2008

FEATURE DESCRIPTION:

There will be a change in the tar format used by SAM-QFS when archiving from the current GNU based format to a posix pax compliant format. This is being done in order to support future extensions to SAM-QFS like storing ACLs and extended attributes, and to provide customers with a tar format that can be extracted by third party tar implemenations that support the posix/pax standard format.

As part of this change, we will make changes in the way the offset of the beginning of the file is stored in the inode. Currently the offset in the inode points to the beginning of the file data. For the new implementation, we will make use of a bit in the inode to indicate that the offset points to the beginning of the tar header data. This will work better with the variable length header data that will be generated in the pax format.

RFE 6686296 – Update SAM-QFS tar format

SCOPE OF THE PROJECT:

- The following are withing the scope of the project:
- changes to the archiver (notable arcopy) to create the new posix pax archive format
- changes to the stager to read and validate the new posix pax archive format
- changes to sls to support diagnostic output appropriate to the new format
- a library used to abstract the creation and reading of the posix pax format headers from the archiver and stager implementations.
- changes to support reading tar format related configuration from the defaults.conf file

FUNCTIONAL REQUIREMENTS:

Backward compatibility:

To enable users to opt to continue using the legacy tar format, we will add an option to defaults.conf:

```
tar_format = [ pax | legacy ]
```

Selecting the legacy option will force arcopy to write tar archives in the same format as it does currently. Selecting the pax option will force arcopy to write tar archives in the new, posix compliant, pax format.

For SAM-QFS x.x, the default will be 'legacy'. The archiver and stager will fully support the pax format. We will expose the pax functionality to customers. Having a standard archive format is expected to be a strong selling point

for new customers.

Existing customers may make the change if they wish, but will need to be advised that rollback from SAM x.x to pre-SAM x.x will not be supported if the archive format is pax. This will be documented.

In a future SAM release, we will change the default to pax. If a customer needs to roll back to SAM x.x after creating pax archives, SAM x.x will fully support reading and writing the archives.

Legacy stager incompatibility:

Currently, the stager validates the magic and version fields in the tar header before it tries to extract the file from the archive. Unfortunately, the magic has changed from the old Gnu format which combines magic and version into "ustar[sp][sp][null]". The pax format requires that the magic is set to "ustar[null]" and the version to "00" (which is not null terminated). Trying to stage a pax format archive with the legacy stager results in the archive being marked damaged.

The situation where this becomes a problem is if a customer needs to roll back a SAM x.x system to pre-SAM x.x, and they have tapes in the posix archive format. Because of the above incompatibility, they would be unable to stage data from tape on the pre-SAM x.x system.

Resolution:

No patch to legacy SAM systems is planned to support the posix format, either partially or fully. This incompatibility will be documented for the SAM x.x release. Customers with pre-SAM x.x systems will be advised that they need to test SAM 5.0 with posix archives thoroughly in their environment before upgrading their primary systems if rollback is a concern. There is no rollback issue if the customer continues to use the legacy format (which is the default).

Pax extended format:

The pax extended format provides an extension to the ustar standard that allows the storage of arbitrary metadata. A pax extended header for a file consists of a ustar header block with type field 'x' followed by a pax extended header data section padded to a 512 byte block length. After this comes the a header block (512 byte ustar header block) and the file data (0 padded to a 512 byte block length) for the file the extended header data should be associated with. This enables non-pax capable archivers to extract the data as two separate files (one metadata, one data).

In the ustar header block for a pax extended header, the type field is set to 'x' as described above, the size field should be set to the unpadded length of the extended header data. The name field should be set to a value that allows the pax data to be associated with the file it belongs to.

The format of the pax extended header data is as follows (quoted from the standard):

> "An extended header shall consist of one or more records, each constructed as follows:
> `"%d %s=%s\n", <length>, <keyword>, <value>`
>
> The extended header records shall be encoded according to the ISO/IEC 10646-1:2000 standard (UTF-8). The <*length*> field, <blank>, equals sign, and <newline> shown shall be limited to the portable character set, as encoded in UTF-8. The <*keyword*> and <*value*> fields can be any UTF-8 characters. The <*length*> field shall be the decimal length of the extended header record in octets, including the trailing <newline>."
>
> [http://www.opengroup.org/onlinepubs/009695399/utilities/pax.html#tag_04_100_13_03]

For SAM x.x, we will use the pax extended format to store filepath (name), uid, gid, uname, gname, linkpath (linkname), and mtime without restrictions on length or size. The formats required by the pax standard are listed in the

table below:

Field	Format	Used when
filepath (name)	UTF-8	name >99 characters (1)
size	unpadded decimal integer	size > 077777777777 (8GB)
uid	unpadded decimal integer	uid >07777777
gid	unpadded decimal integer	gid >07777777
uname	UTF-8	uname >31 characters
gname	UTF-8	gname >31 characters
linkpath (linkname)	UTF-8	linkname >99 characters
mtime	unpadded decimal float	mtime > 077777777777 (2)

Notes:

(1) Strictly speaking, you can use the prefix as well. We will not use it since it requires being able to split the name on a slash '/'.

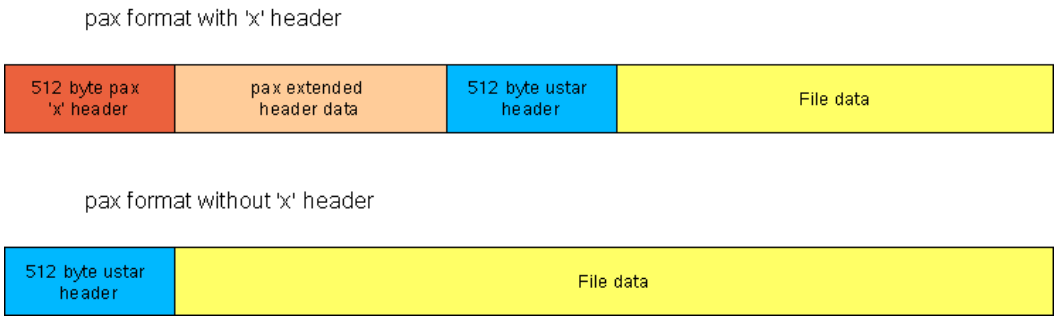
(2) Y2K38 bug – will be needed for 64 bit time_t. Also can be used for file systems that support sub-second resolution.

The values in the pax extended header override the values in the ustar header that follows. The values in the ustar header block following the pax extended header data should be set to values that allow a non-pax extractor to create a file with the pax attributes in it, and another file containing the file data.

For regular files smaller than 8GB, this does not present a problem – the complete pax data and the complete file data can be extracted based upon the name specified in the header blocks. Ownership can be assigned appropriately (owner or root) depending on whether or not the uid/gid uname/gname fit in the header block. Filenames that don't fit can be assigned systematically in the header blocks and stored in full in the pax data.

Symlinks cannot be recovered if the linkname won't fit in the linkname field in the header block. The data in files larger than 8GB cannot be recovered because there is no standards compliant way to represent the size of the file in the size field.

The only way to retrieve data if these issues come up is to use a pax capable tool to extract the archives.



For more information on the pax format:

http://www.opengroup.org/onlinepubs/009695399/utilities/pax.html#tag_04_100_13_01

Staging and legacy archive compatibility:

A SAM-FS system with legacy archives on tape must be fully compatible with the posix-capable stager.

Resolution:

1. The stager will read a 512 byte block from tape
2. The stager will inspect the header block before beginning processing:

```
if (header.type = 'x') {
        //POSIX archive with pax extended headers
} else {
        if (header.magic == “ustar  “) //legacy archive
        else if (header.magic == “ustar\000” //POSIX archive
        else //damaged archive
}
```

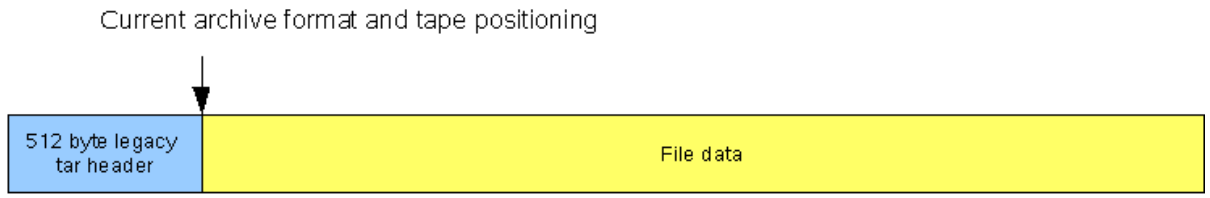
3. For a posix archive with pax extended headers, the stager will load the pax headers, and check for a file size.
4. If a file size is found in the extended headers, that will be the file size that is verified against the inode
5. For legacy archives or posix archives without extended headers, the file size will be read from the posix or legacy header block.
6. The stager validates the file size against the inode
7. The stager determines how far it needs to seek from the inode offset to the beginning of the file data. This is:

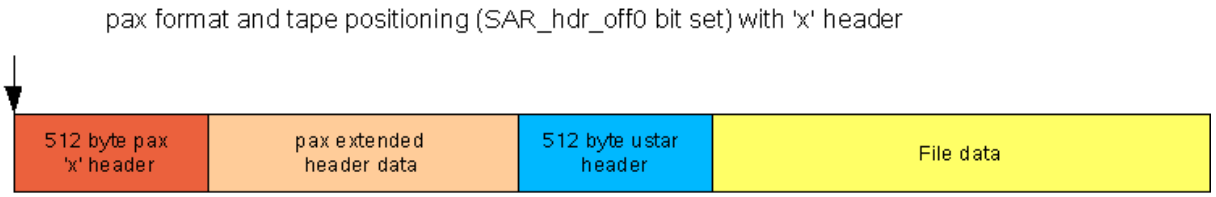
   512 (pax 'x' header block)
   + size of pax extended header data rounded up to a 512 byte boundary
   + 512 (file data ustar header block))
8. The stager seeks to the beginning of the file data.
9. Staging the file proceeds as normal.

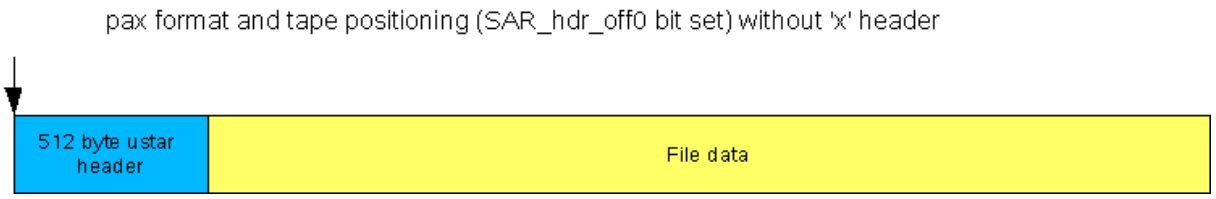
Tape positioning:

Tape positioning for legacy format archives will be maintained at the end of the GNU tar header block/beginning of file data. For archives in the new format, the file_offset stored in the inode will be changed to point to the beginning of the first tar header block (regular ustar, or pax 'x' header). The stager will position the tape based upon whether the SAR_hdr_off0 flag is set for the archive copy.

Currently the stager subtracts the size of the header block (512 bytes) from the offset to position the tape. If the SAR_hdr_off0 flag is set, the stager will not subtract the header block, and position the tape directly at the offset in the inode.







Diagnostic information:

For diagnostic purposes (sls -D, for example), we need to be able to determine the type of the archive from the inode information. Because of limitations in the number of free bits in the inode, we will not store archive format in the inode. Instead we will use the SAR_hdr_off0 bit to implicitly determine the archive type.

Since we are maintaining the current tape positioning for legacy format archives, this bit will not be set. For posix format archives, this bit will be set. Based upon this, we can determine the archive type without inspecting the actual data on tape.

For the foreseeable future, this relationship will be maintained between the status of the SAR_hdr_off0 bit and the archive format. The only possible reasons for this to change is if we were to change the tape positioning for legacy format archives (unlikely), or if we need to change the format again in the future. In either of these cases, we will need to revisit the way we handle diagnostic information.

DELIVERABLES:

- arcopy: Changes to tar header generation and how tape positioning is stored in the inode.
- stager: Changes to tar header reading for validation of files, changes to tape positioning logic.
- sls: Changes to reading the inode and what is displayed if the tape positioning is set to the beginning of the tar header (p for posix, L for legacy)
- pax_xhdr library: Provides a convenient way of reading and writing the pax headers, abstracted from the archiver and stager.

FUNCTIONAL DEPENDENCIES:

- libiconv (if needed to convert filenames, usernames, and groupnames to UTF-8)