Configuration Information

MAR\_reserved\_xattrs

MARobjid (object-creation template with many fields)

MARpost (post object-creation updatable info)

MARrestart (restart flag for pftool restart)

MARobjid fields

Object-IDs (object-names) are generated from some of these fields, though there may also be fields here (e.g. repo) that are not used in the object name. The highest-order field in the object-name is a reversed-order copy of the mdfilectime, because AWS docs suggest this helps maximize distribution of hashes, or something.

|  |  |
| --- | --- |
| bucket | Conceptually, this is “<namespace>.<repo>”. Bucket names in S3 are required to be no more than 63 chars, and can’t include slashes. Config should enforce these constraints so this always works. Slashes in namespace will be escaped somehow. |
| Reverseorder time stamp | This is a reverse order time stamp of some kind to make sorting easy when listing objects. |
| recordversion | Version number for this record |
| mdfilecreatetime | In the case of uni and multi files, this is the creation time from the mdfile. In the case of packed, this is the creation time of the first file being packed into the object. |
| Objcreatetime | This field is used to put current time stamp (in addition to mdfile creat time. This is used to “version” objects. For example on a truncate to zero, which would put all the objects for that file into the trash, but the names will be the same as create time and inode remain unchanged. This makes a unique name for the new objects for that file but yet they are still related by all fields except this one. |
| objtype | Packed if many files are being packed into the object or Not packed if not. |
| comptype | Compression type |
| sectype | Security type (encryption) |
| correcttype | Correctness type (crc/checksum) |
| objchnksz | Records chunk size for multi or striped file, this value is initially populated from the repo configuration table for the file based on namespace/path, chunk size is picked based on if this is interactive or batch and for batch, it is based on the size of the file being moved, (large, xlarge) derived from the configuration file. |
| objchnknumber | chunknumber this object is in the multi chunk file (this is always zero) in every case. For uni and packed there is only one object involved. For multi, the object name only changes by chunk number which is calculated based on chunksize So this is really just a placeholder value. |
| mdinode | Inode of the metadata file (for packed, it’s the first file in the object) |

MARpost fields

This is metadata that can’t be known until after an object has been written.

|  |  |
| --- | --- |
| recordversion | Version number for this record |
| objtype | Records how the file data is stored in an external object repo, there are currently 4 types:  Uni – one object stores the entire file  Multi – a file is spread across multiple objects using chunk sized objects, object id’s are recorded in the metadata file  Packed – multiple files in each object which requires using the objoffset field.  Striped – a file is striped across multiple objects using chunksize from the configuration file and round robin, object id’s are recorded in metadata file |
| spaceused | Space used in the object system for the entire file (may have to sum multiple object space used for multi) |
| objoffset | Records offset into object where file data lives, this is only used in Packed (multiple files in one object) |
| correctnessvalue | Checksum or CRC for the entire file (may have to sum multiple checksums or CRC’s for multi) |
| numobjects | For multi-object files, records the number of objects in the metadata file that contain chunk information (the rest of the file is just a hole to make the size of the metadata file == the size of the file |
| Chunkinfobytes | Number of bytes of chunkinfo in the file |

MARrestart

This is just a flag to indicate the file is in restart mode -- if not present then not in restart mode. Restart means creation of the file (e.g., by pftool) is not yet complete.

MAR\_mnttop top level mount point for fuse/pftool

Fuse/pftool will append MAR\_mnttop on the front of all the name space segments below to construct a namespace tree

Example

MAR\_mnttop = /redsea

MAR\_namespace.mnt.mntpath = /projecta

MAR\_namespace.mnt.mdpath = /md/projecta

MAR\_namespace.mnt.mntpath = /projectb

MAR\_namespace.mnt.mdpath = /md/project

User refers to /redsea/projecta and that refers to files metadata file system/namespace in /md/projecta

MAR\_namespace { (one stanza for each namespace segment) {

(one for each namespace)

|  |  |
| --- | --- |
| name | name that refers to this namespace used in the objid that gets store as the name of the object in the object system |
| mntpath | Specifies the path for this namespace, which is appended to the MAR\_mnttop (described above) (specified as a path with slashes) |
| bperms | Specifies permissions for batch programs. These permissions are above and beyond the POSIX permissions (rwx/ugo). This is because external repositories may have special permissions that don’t map exactly to POSIX permissions. The values are rmwmrdwdudtd:  rm – read metadata  wm – write metadata  rd – read data  wd – write data  ud – unlink data  td – truncate data  An example of interesting use is to allow read and write in POSIX permissions, allow metadata changes but not allow writing of data. This value is not stored with the file, it is interpreted real time, so this is a fast way to shut of write of data or metadata etc. This item can change based on allowed activity against this namespace and the data/space it represents. |
| Iperms | Specifies permissions for interactive (FUSE) programs. Same as bperms above. |
| Mdpath | Specifies the path for the POSIX file system that is to hold the metadata and potentially data for this namespace. If a file is to be written to an external repo, then only metadata is stored in this file sytem, but if data is to be stored into this file system then both data and metadata are used. Controlling whether data is written into the metadata file system is done in the repo configuration table using the repomethodinfo field so the repo you are writing to or reading from will be DIRECT (use the metadata file system) or some other external method like CDMI, S3, etc. This is specified using path notation using slashes. This can change if you have moved the metadata file system path for some reason. It might be hard to change on the fly though. |
| Iwrite\_repo | Specifies what repo interactive (FUSE) applications will write new files to, points at a name in the repo table. This can be changed as it just controls where new files are written to. |
| Iwrite\_chunksize | Chunksize for interactive |
| Swrite\_repo | Specifies what repo batch applications will write new small files to, points at a name in the repo table. This can be changed as it just controls where new files are written to. |
| Swrite\_size | Size below which is considered a small file. This can be changed as it just controls where new files are written to. |
| Swrite\_packsize | Size of object to pack multiple small files into  If this value is zero then packing will not occur |
| Mwrite\_repo | Specifies what repo batch applications will write new medium files to, points at a name in the repo table. This can be changed as it just controls where new files are written to. |
| Mwrite\_size | Size below which is considered a medium file. This can be changed as it just controls where new files are written to. |
| Lwrite\_repo | Specifies what repo batch applications will write new large files to, points at a name in the repo table. This can be changed as it just controls where new files are written to. |
| Lwrite\_size | Size below which is considered a large file. This can be changed as it just controls where new files are written to. |
| Lwrite\_chunksize | Chunksize for large files |
| xlwrite\_repo | Specifies what repo batch applications will write new xlarge files to, points at a name in the repo table. This can be changed as it just controls where new files are written to. |
| Xlwrite\_chunksize | Chunk size for xlarge files |
| trashmdpath | Specifies where in the namespace, information is stored on unlink and trunc/ftrunc operations, which could provide a trashcan function but is used by batch process for reclaiming space, repacking, reconciliation of space which is needed for external repo’s. All permanent deletion of data (both unlink and trunc) are done in batch for external repos. For “DIRECT” repos where the data is stored directly in the metadata file, unlink operations go to this path, but trunc’d space is not preserved. This is specified as a path with slashes. It is assumed that this is in the same metadata file system as the metadata file system for this name space, as rename is used for unlink operations. This value could change but much care would have to be taken because entries into this path can be occurring all the time and information about reclaimable space lives in this path. |
| fsinfopath | This is a path name specified with slashes to a file that contains the values one would get in a statfs/statvfs call like how much space is in the file system, how much space is used, etc. This file must be updated in a lazy way via periodic batch scans of inode space etc. Since the space for the files in a namespace may not be in the metadata file system associated with a name space, it is required that this info be provided in some way to be chosen by the site. It could involve walking the metadata tree or inode space and adding up spaced used or it could involve querying an external repository for space etc. This value could be change but care needs to be taken as statfs/statvfs calls will look in this file for providing information. |
| quota\_space | Specifies the space quota for this name space. This value is compared to information in the fsinfopath file above about how much space has been used which is populated via lazy batch runs to determine and record space used. This can be changed at any time, but will not take effect immediately as quota’s are done in a lazy way based on batch runs to update the fsinfopath file |
| quota\_name | Specifies the inode quota for this name space. This value is compared to information in the fsinfopath file above about how many inodes have been used which is populated via lazy batch runs to determine and record inodes used. This can be changed at any time, but will not take effect immediately as quota’s are done in a lazy way based on batch runs to update the fsinfopath file |

}

MAR\_datarepo { (one stanza for each data repo) {

|  |  |
| --- | --- |
| name | Name for this repo, this name is used in the namespace table above in the config file and it is also used stored with the file in xattr, so this can not be changed easily. It follows the same rules as deleting a repo in this list. |
| URLprefix | This is a string associated with the repo used to access the repo  Object names will be repo URLprefex/bucket/objid  Or really URLprefex/namespace.repo/objid (which is formed and stored in the MAR\_objid xattr |
| updateinplace | Updates in place for files in this repo are allowed. This lets you decide if a file is in a repo that can do update in place then the FUSE and batch programs can allow update in place. If a repo doesn’t allow this easily then you can forbid it. It is probably good practice to not allow this for all repo’s used in a namespace but you don’t have to do that. Update in place means that if you open for write, you have to overwrite the entire file from the beginning. It also means that you cannot truncate the file to any other value than zero. It also means that you cannot open with append and append to the file , (although this capability might be changed at a later date). The software can use update in place for DIRECT as the repomethodinfo (which tells the software to put the file data in the metadata file). (yes/no). This can be changed but it is not recommended. |
| repomethodinfo | Info about method for accessing the object repo, like S3 or CDMI |
| securitymethod | Specifies a method for how security works on this repo (authentication/authorization), this can change as it is not recorded anywhere other than in this file but any backend storage system must be kept in sync with this method |
| sectype | Specifies a method for encryption for data for the repo. This can NOT change as all files that have data in this repo are encrypted with this type. |
| comptype | Specifies a method for compression for data for the repo. This can NOT change as all files that have data in this repo are compressed with this type. |
| correcttype | Specifies a method for correctness for data for the repo. This can NOT change as all files that have data in this repo have this correction information calculated and stored with this type. |
| onoffline | Specifies a method for bringing online a repo if the repo is of the type that allows it to be offline. This value can change as it is dynamic. |
| latency | Specifies a time it might take to bring a repo online. This value can change as it is dynamic. |

}