HP-UX/AdvFS On Disk Structure Scoping

Revision V1.0

SDM

October 22, 2002

Change History

Table of Contents

---

1 Introduction

1.1 Overview

This document started out as a look at what on disk structures (ODS) could be added or modified to allow more metadata robustness. The goal was to make the recovery tools able to do a better job of quickly restoring from a corrupted file system.

After the first revision it was decided to change this document into a place for all of the proposed changes to the AdvFS ODS for the Tru64 to HP-UX port.

While looking at ODSs I have tried to look towards the future and have added additional reserved fields to most ODSs. The recovery tools, or future improvements to AdvFS would be able to use them.

Some of these proposed changes are items that have been suggested over the years from many third parties. They were included to make sure they don't get lost during all the scoping work.

1.2 Assumptions

- All on disk structures (ODS) are fair game for changes.
- Do not want make change to the ODS which would require the file system to be converted to the new format, for 10 years after first release. Change to ODS which do not cause backwards compatibility problems, will be allowed.
- Physical storage is doubling in size every year. Based on the requirement of no changes to ODS for 10 years, all size (storage) related fields must be able handle values which are 2^10 larger than they are today.
- The end user model of multiple filesets per domain will be replaced with a single file system model. AdvFS will still internally deal with multiple filesets to help support snapshot functionality
- Will support up to multiple volumes, up to 256 volumes per file system.
- Variable on disk page sizes will happen in the future, make sure the structures could handle them. Metadata and user-data might use different size pages. It is possible that different user files will use different size pages.
- The frag file will go away.

1.3 Impacts

T.B.D.

1.4 Future Improvements

- Clones (Snapshots) will not be implemented in the first release. They will be added in at a later release. The current plan is to use the same model for clones we use today, with room for future improvements. This requires leaving in the structures for multiple filesets per file system. The multiple filesets will be hidden from the user.

- A new version of property lists (ACLs) is currently being designed. Property list changes herein are subject to change.
- Because the frag file is going away with the HP-UX port, this will cause file systems which contain mostly small files, to use as much as 8 times the current amount of disk space. A new solution to solve this problem must be developed, one method would be to use small variable size pages (1K).

1.5 References

1.6 Tru64 vs HP-UX name definitions

Some terminology from the Tru64 version of AdvFS will be changing with the HP-UX port. The following table will show you each side's equivalent.

Tru64	**HP-UX**
domain	file system
domain name	Fsname
clone	Snapshot
fixfdmn, verify	fsck_advfs
/etc/fdmns	/dev/advfs

1.7 Acknowledgments

Thanks to various members of the AdvFS recovery team, JC for her work on the Tru64 Metadata Robustness Project, and ND for his large file and disk work.

2 On Disk Structure Changes

2.1 Global Changes

2.1.1 Unique Magic Number for each Metadata page.

Structures changed: 3.2.3, 3.2.27, 3.2.29, 3.2.31

Magic numbers will be an unique uint32T field, which would be stored on each metadata page. This magic number would be stored at a known location, which then could be checked to verify what type of page is being worked on. The recommend format is 0xADFXXXXX, were XXXXX is a unique identifier per page type.

The following page types all have a header that the magic number could be stored in:

- BMT
- RBMT
- SBM
- Tag file
- Root Tag File
- Log

The following page types do not have headers, so they would not have the magic number:

- directories
- dir Indexes

2.1.1.1 Benefit

Magic numbers improve the reliability of the recovery tools in cases of severe corruptions. In these cases, the tools must sequentially search every page on each volume to locate the required Metadata.

2.1.1.2 Cost

An extra uint32T per page would need to be reserved for this value.

2.1.2 On Disk Structure Version Number (megaVersion)

2.1.2.1 Major and Minor number

Structure changes: 4.3

There have been times the AdvFS team have made minor changes to or have added new on disk structures (ODS), but have not wanted to make a change to the ODS version number due to the large affect this would have on the kernel. This makes it very hard to tell if data on disk is actually a corruption or a designed change. The proposed change is to split this value into a major and minor number.

The major number will be an uint16T which will be incremented when any changes are made to the ODS which will cause the file system not to be usable (backward compatibility) on machines running previously released kernels.

The minor number will be an uint16T which will be incremented when modifications, additions or usage changes are made to the ODS which previous kernels will be able to handle or ignore (i.e. The change does not break backward compatibility).

On mount time (both R/W and read only) if the current minor number is smaller that what the kernel expects the kernel should increment the minor number and be able to handle the new or changed structures from that point on. This reason this needs to be done for

read only mounts is that metadata can still change on a read only mount, only user data is read only.

If a filesystem with a higher minor number than expected is mounted, the kernel must be smart enough not to delete/corrupt this newly created or changed structures. If this can't be done, then the designer should increment the major number instead when doing the code changes.

Both the Major and Minor number will be stored in a single structure.

2.1.2.1.1 Benefit

Minor future changes to the ODS only need to increment the minor number, which will limit the changes required to the kernel and utilities.

Tools like salvage and fsck_advfs will have a better idea if a file system is corrupted or just has been upgraded.

2.1.2.2 Stored in fixed location

Structure changes: 3.2.12

The ODS version number, which defines the structures of the metadata, should be stored outside of the main metadata, in a place which can be described without reference to the metadata.

The one place which we know will always exist on every file system which is outside the normal metadata is the super-block. We already store an AdvFS Magic Number there, so it should be trivial to store the version number there as well.

The kernel will make all ODS version number checks based on the number stored in the super-block, but to help in recovery work a redundant copy will be saved in the RBMT. If the ODS version number changes the kernel will need to make sure both places are updated.

2.1.2.2.1 Benefit

Currently this value is stored on on each RBMT and BMT page in the page header. By moving this to the super-block, it allows easier upgrades of ODS via a convert tool.

2.1.3 Tag and Seq Number

2.1.3.1 Increase the size of tag.num

Structure changes: 3.1.1

Change the type of tag.num from uint32T to uint64T.

2.1.3.1.1 Benefit

AdvFS has the current limit of 2^31 files per fileset, this would increase the limit to 2^63 files. Note that even though this is an unsigned value; we do have a few special case reserved signed values.

2.1.3.1.2 Cost

This change affects multiple ODS structures (tag pages, mcells), including ones that do not belong to just AdvFS (directory structures).

Extra 4 bytes per tag will be used.

2.1.3.2 Change the usage model on tag.seq

The following changes do not change the ODS, but instead how the kernel deals with the values stored in the field.

2.1.3.2.1 Allow wrapping of seq numbers

Currently AdvFS uses some of the bit in the seq number to keep track of different instances of a tag over time. This is done to protect from corruption of data during a log replay. This causes a limit on the number of times a tag can be reused. This limit is currently ~32K times, once this limit is reach the tag is marked as dead and is no longer usable.

The kernel should also be changed to handle the wrap case, so that when the seq number wraps the tag is still usable.

2.1.3.2.2 Add a random element to the seq number.

NFS requires inodes to be unique. In UFS they use what is known as a generation number, for AdvFS we use the seq number. To improve security for NFS, a portion of the seq number will randomly generated.

It is suggested to use the lower 16 bits for the seq, and the upper 16 bits for the random portion.

2.1.3.2.3 Move all flag bits in seq number.

A flags field has been created in the bsTMapT structure (See section **2.5.5**). This is a better place for flags to be stored. One of the flags which should be moved is BS_TD_IN_USE.

2.1.4 Mcell Ids (bfMCIdT)

Structure changes: 3.1.2

2.1.4.1 Increase the maximum size of the BMT

Increase the maximum number of pages in the BMT from 2^27 to 2^40.

2.1.4.1.1 Benefits

This allows 1 trillion pages in the BMT. Assuming AdvFS someday supports an on disk page size of as small as 1KB, this allows the BMT to grow to be as large as a Petabyte. For large page size the BMT can be even larger.

2.1.4.2 Increase the maximum number of mcells per BMT page

Increase the maximum number of mcells on a BMT page from 2^5 to 2^16.

2.1.4.2.1 Benefits

This allows 64 thousand mcells on a single BMT page. This will allow AdvFS to supports an on disk page size as large as 16MB, using the current mcell size. If the mcell size increases we could support even larger on disk page sizes.

2.1.4.3 Add the volume Id

In all most all places we access the Mcell Id, we also need the Volume Id. So add a volume Id of 2^8, this is same size as the currently supported maximum number of volumes on Tru64.

2.1.4.3.1 Benefits

Gives us a maximum of 256 volumes per file system. As volume managers improve our belief is that more users will go to a single volume model, so there is no reason to increase it beyond this.

Currently we use two structures to locate a mcells location. This puts all the fields in one structure.

2.1.5 Multiple volumes

If the storage stack will most likely not support large (16TB+) volume sizes by the time of AdvFS's first release on HP-UX, AdvFS will need to support multiple volumes in a file system if we wish to keep Tru64s 16TB capability.

2.1.6 Variable Page Sizes (VPS)

Structure changes: 3.2.11

The page size will be set when the file system is first created. Once the size is set it will not be allowed to change.

The record which contains the VPS size must be stored in a location that the kernel will always be able to find, regardless of the page size. It is recommended that the first block of the first page in the RBMT (page 0, mcell 0) be used.

VPS Limits:

- Minimum page size: 1KB
- Maximum page size: 16MB

The 16M maximum is due to the number of mcells that will fit in a BMT page. If the mcell size grows, then this limit could be increased. Another way we could increase this limit is to cut down the maximum number of volumes in a file system.

2.1.6.1 User files and Metadata use different VPS

Structure changes: 3.2.11, 4.5

Separate the metadata and the user-file variable page sizes. This allows more flexibility for future improvements.

A few structures currently are defined based on the size of the page. With out major changes to the way the kernel handles these structures, this could only be done by changing a variable and rebuilding the kernel. To help in the process, two new global defines called ADVFS_METADATA_PGSZ and ADVFS_METADATA_PGSZ_IN_BLKS will be created.

2.1.6.1.1 Costs

The SBM will need to keep track of page sizes based on the smaller of the two variable page sizes.

2.1.6.2 Different volumes will use different VPSs

Each volume will have different variable page sizes (VPS), these sizes will be set on a per volume basis and once set will not be allowed to change. This allows more flexibility for future improvements

2.1.6.2.1 Benefit

If the customer wishes to change the variable page size in the file system, they would just need to add a new volume with the new page size, then remove the old volume which would migrate the data to the new volume.

2.1.6.3 Individual User files use different VPS

Structure changes: 3.2.8

The structures for this exist in Tru64, but the functionality was never done. Keeping the per file blks per page allows more flexibility for future improvements.

2.1.6.3.1 Costs

The SBM will need to keep track of page sizes based on the smallest possible variable page sizes.

2.1.7 Unique variable names

Structure changes: All structures

For most (not all) HP-UX structures they use a name convention in which all elements of a structure have a common prefix attached. All AdvFS structures should be changed to follow this practice.

2.1.7.1 Benefits

Improves the readability of structures when in the debugger.

Allows easier look up of variables in tools such as cscope.

2.1.7.2 Cost

Massive changes to variable names through out the AdvFS source code. This could be done by doing global replaces on all the source code.

2.1.8 Do not use the X/Open typedefs

Structure changes: 3.2.21, 3.2.26, 3.3.2

Replace all references to time_t with int64T.

2.1.8.1 Benefits

Allow us to remain independent of any compile time changes that happen in future releases.

2.1.9 Prealloced extents

Customers (mostly DB vendors) want a way to pre-alloc large amounts of space on the disk, with out the overhead of zeroing them all out at create time. The reason behind this is that they want all of there data on the disk to be sequential, and not fragmented.

The problem with this scheme is the classic data reuse problem, there might be sensitive data on those pages which then could be accessed by the people or application which asked for the pages. It has been decided that only root priveledged applications will be able to do use prealloced extents, and those tools are responsible for any data reuse issues.

No ODS changes required.

2.1.10 Create an AdvFS file system Cookie (fsCookie)

Structure changes: 4.4, 3.2.3, 3.2.12, 3.2.27, 3.2.29, 3.2.31, 3.2.32

At file system creation time a known value will be created based on the creation time, this value will be known as the fsCookie. The fsCookie will be the same number as the domainId. In the case of a split mirror being dual mounted, the fsCookie would not be updated, were the domainId would be. The fsCookie will be stored in the header of most metadata pages.

2.1.10.1 Benefit

Would improve the reliability of recovery tools when in cases of severe corruptions the tools must search the volumes to locate the Metadata. Current tools have a hard time determining if the page they find actually belongs to this file system, or is left over from an old file system on this volume.

2.1.10.2 Costs

Extra 8 bytes per page would need to be used.

2.1.11 Remove references to time_t in bsIdT

Structure changes: 3.1.4

This structure contents is actually a timeval, which contains a time_t. The problem we have is timeval is not a structure which AdvFS owns, so its size could change out from under us. So instead of using the timeval structure, create a structure which AdvFS has control over.

2.1.12 Stopping accidental reuse of AdvFS volumes

On Tru64 a field in the disklabel is marked when a volume is part of a domain, this allows AdvFS to stop the user from accidentally overwriting a partition which is used in another AdvFS domain. As HP-UX does not have the concept of disk labels, so a new way to stop this will need to be implemented.

One way that this could be done is check the super-block to see if the AdvFS magicnumber is valid. This would require that rmfdmn (or its equivalent) zero out the magic number when a volume is removed from a file system, or the file system is deleted.

2.1.13 Flag for multi-volume file systems

AdvFS needs a way to know if a file system has been setup for multi volume support. AdvFS has flags in some of the RBTM mcells which store information which could be used for this.

It would be useful to be able to check if this file system has been setup for multi volume support, with out having to activate the domain. To allow for this AdvFS will also store a value at a known location in the super-block which will signify if the filesystem has been setup for multi volume support.

2.2 Log File Changes

2.2.1 Unique magic number at a known location

Structure changes: 3.2.31

See Section **2.1.1**

2.2.2 New reserved fields

Add two uint32T reserved fields in the page header for future changes.

2.2.3 Replace domainId with fsCookie.

Structure changes: 3.2.31, 3.2.32

The domainId in the log can become out of sync after a mirror has been split, and both halves have been mounted. To solve this problem a global fsCookie has been created. (See section **2.1.10**)

The domainId is stored in both the header and trailer record on each log file page, both will be changed to the domainCookie.

2.2.3.1 Benefits

Useful for sanity checks in the recovery tools.

2.3 Storage Bit Map (SBM)

Structure changes: 3.2.27

2.3.1 Unique magic number at a known location

See Section **2.1.1**

2.3.2 Store the page number

Save the SBM page number in the header block of each SBM page.

2.3.2.1 Benefits

Useful for sanity checks in the recovery tools.

Once we allow the size of volumes to change, the SBM can become fragmented. This gives AdvFS an additional way to piece the SBM back together after a corruption.

2.3.2.2 Cost

Extra uint32T (4 bytes) per SBM page would need to be used.

2.3.3 Store the fsCookie.

See section **2.1.10** for more info.

2.3.3.1 Benefits

Useful for sanity checks in the recovery tools.

2.3.3.2 Cost

Extra uint32T (4 bytes) per SBM page would need to be used.

2.4 Frag file

Structure changes: 3.2.16, 3.3.2

2.4.1 Remove the frag file functionality

When a large file system was 1GB it made sense to save partial pages, but now when we have file systems which are in the hundreds of GBs if not TBs does not make sense to save frags on most file systems.

In Tru64 we now give the customer the ability to turn this functionality off. This was done, as customers are more concerned with speeding up their file systems versus saving of extra disk space.

2.5 Tag File

Structure changes: 3.2.28, 3.2.29

2.5.1 Unique magic number at a known location

See Section **2.1.1**

2.5.2 Fix off by 1 confusion

Fix the numbering in the tag file. Currently the free list in the tag file is off by 1, So when tmNextMap or nextFreeMap says the next free entry is '11', it is actually element '10' in the array.

2.5.2.1 Benefit

Easier to understand code.

2.5.2.2 Cost

Will require kernel logic changes, as will need to change the end case from '0' to '-1'.

2.5.3 Increase maximum number of tags

See section **2.1.3.1**.

2.5.4 Increase the size of seqNum

This structure is a uint16T, it needs to be the same size as the seq number stored in the bfTagT record. The seqNum only needs to be stored in the tm_s3 section of the union.

2.5.5 Add a bit flags field

Add a new field that can be used for bit flags for each tag. This needs to be stored as the first element of all 3 parts of the union.

The following fields will be moved from their current structure to here:

- deleteWithClones - Currently in bsBfAttrT
- outOfSyncClone - Currently in bsBfAttrT
- BS_TD_IN_USE - Currently stored in tag.seq
- dataSafety - Currently in bsBfClAttrT

2.5.5.1 Benefits

Will replace unit32T fields in other records with a single bit flag located in the tag file.

In the case of deleteWithClones, when a snapshot is deleted the kernel would not have to read a page for each tag to get it's attribute record. In the worse case were every tag's primary mcell is on a different page this would save 1000 disk reads per tag page.

2.5.5.2 Cost

Extra uint32T (4 bytes) per tag page would need to be used.

2.5.6 Store the fsCookie.

See section **2.1.10** for more info.

2.5.6.1 Benefits

Useful for sanity checks in the recovery tools.

2.5.6.2 Cost

Extra uint32T (4 bytes) per Tag page would need to be used.

2.5.7 Add new reserved field

To make the math work out right when computing the number of tags per tag page, fill the extra space with reserved fields. This works out to be either 1 or 5 reserved fields. Have decided to go with 1 in the page header, should be enough for future changes.

2.5.8 Store the fileset id

By storing the fileset id in the tag page header it gives a way to know which fileset a tag file page belongs to. For root tag page use -2 as the fileset id.

2.5.8.1 Benefits

Would improve the reliability of recovery tools when in cases of severe corruptions the tools must search the volumes to locate the Metadata.

Currently if the primary mcell for the fileset cannot be found we do not know which fileset the tag page belongs to.

Using fileset id '-2' will easily allow us to tell the difference between a root tag file and a fileset tag file.

2.5.8.2 Cost

Extra 8 bytes per tag page would need to be used.

2.6 BMT/RBMT page

Structure changes: 3.2.3

2.6.1 Unique magic number at a known location

See section **2.1.1**

2.6.2 Store the fsCookie.

See section **2.1.10** for more info.

2.6.2.1 Benefits

Would improve the reliability of recovery tools when in cases of severe corruptions the tools must search the volumes to located the Metadata. Current tools have a hard time determining if the page they find actually belongs to this file system, or is left over from an old file system on this volume.

2.6.2.2 Cost

Extra uint32T (4 bytes) per BMT page would need to be used.

2.6.3 Duplicate the ODS version number in the BSR_DMN_MATTR record

Structure changes: 3.2.19

Store a copy of the ODS version number(s) in the RBMT record BSR_DMN_MATTR. The main ODS version number will be stored in the super-block of the volume.

2.6.3.1 Benefit

This redundant copy will be used by the recovery tools to help check for corruptions.

2.6.4 Change the type of nextFreePg and freeMcellCnt

With the increase in the maximum number of pages in the BMT, nextFreePg needs to be increased. freeMcellCnt actually doesn't need to be as large as it currently is, so have them share the bits of a uint64T.

2.6.4.1 Benefit

Better use of the space on the disk.

Creates 8 bit reserved field for future changes.

2.6.5 Increase the size of pageId

Increase pageId from a uint32T to a uint64T. AdvFS only uses 40 bits of the field, so the other 24 bit can be used as a reserved field.

2.6.5.1 Benefit

Creates a 24 bit reserved field for future changes.

2.6.5.2 Costs

Extra 4 bytes per page would need to be used.

2.6.6 Add new reserved fields

Add two uint32T reserved fields in the page header for future changes.

2.6.7 Change the usage of BSR_XTNT record in the MISC mcell

The RBMT mcell for the MISC files currently uses a hack where it puts 3 elements in an array that is hard coded to handle 2 elements. This has always caused confusion, as well as creating complier warning messages.

This was originally done to save valuable space on BMT page 0. With the invention of the RBMT, AdvFS no longer has the space constraints that it once had. So instead of this hack, use the extent chain pointer to create a chain of extents for the misc extents.

2.6.7.1 Benefit

Will clear up the confusion on the MISC mcell in the RBMT.

Will get rid of compile time warning messages.

2.7 Shadow RBMT pages (aka Redundant RBMT)

Structure changes: 4.1

Store two additional copies of all RBMT pages. These pages will be stored at a known (or computable) location. Possible locations for the shadow RBMT pages would be at the last page of the volume, and a fixed location. (example page 1000 on the disk)

At the cost of recoverability, this could be made an option to be turned on or off at file system creation time to give a slight improvement to the performance.

2.7.1 Benefit

In the case of corruption the recovery tools would be able to use the shadow pages to check and repair corruptions of the primary RBMT page.

2.7.2 Cost

Keeping the RBMT shadow pages in sync will require some extra disk I/O and CPU cycles. This isn't a major concern as the RBMT is fairly static and doesn't change very often.

Puts unmovable islands on the disk. Vfast must work around these islands. Any database that would like to have the entire disk as contiguous space would need to fit between these islands.

2.8 Global Mcell changes

2.8.1 Mcell header (bsMCT)

Structure changes: 3.2.2

2.8.1.1 Store number of records in the mcell

When an mcell record is added or deleted from this mcell, modify the count. This will not add any extra I/O as AdvFS is already modify the page.

2.8.1.1.1 Benefit

Recovery tools would have a better idea of how many records they would need to find in a mcell.

2.8.1.2 Increase the size of linkSegment

The linkSegment field lets us know the order of mcells in the primary chain of mcells. It is currently uint16T, which could to wrap at its current size, so change it to an uint32T.

The one known way this field could wrap would be on a file with a large number of property lists.

2.8.1.2.1 Cost

Could cause byte alignment issues (See below).

2.8.1.3 Add reserved fields

Add an uint32T reserved fields at the end of the record for future changes.

If there is a byte alignment error, add an uint16T as well.

2.8.2 Mcell Record headers (bsMRT)

Structure changes: 3.2.1

8 bit field called version exists in this structure. Currently it is never used, so change this field to reserved.

2.8.3 Extent record (bsXtntT) to handle 64bit page and block size

Structure changes: 3.2.4

Change the type of vdBlk and bsPage from uint32T to uint64T.

2.8.3.1 Benefit

Increases the limit on the maximum size of a volume.

2.8.3.2 Costs

This is the most common piece of metadata, which with this change will use up twice as much space.

This will require large number of changes through out the kernel.

SBM caching could become complicated with this scheme.

2.8.4 Add reserved fields

Structure changes: Most structures

Add reserved fields to most (all) of the mcell records. More details will be included with each individual mcell record type.

2.8.4.1 Benefit

Right now if we ever need to make a change to any mcell record we need to jump through major hoops due to the changes to on disk structures. If we had some reserve fields we could use them as needed.

2.8.4.2 Cost

Increase the overall percentage of metadata to user-data.

2.8.5 Increase the amount of space in a mcell.

The size of a mcell cell should be large enough to handle all three of the following records: BSR_ATTR, BSR_XTNTS, BMTR_FS_STAT.

Based on current calculations 354 bytes will be needed per mcell. This will decrease the number of mcells per 8K page from 28 to 23 mcells.

2.8.5.1 Benefit

These 3 records are required for a basic user file. If all three records can fit in a single mcell, then it will cut in half the number of page reads required when the stat record needs to be accessed.

The size should be computed to limit the amount of wasted space on the default metadata page size. It should also take in to account that in the future we might have variable page sizes (VPS) and we don't want to wasted space on smaller pages.

2.8.5.2 Cost

Depending on how much additional space is added to the mcell, it will lower the number of mcells that will fit on a single BMT page. This will require a higher amount of disk space overhead for metadata.

2.9 Mcell Records

Structure changes: 3.2.7, 3.2.8

2.9.1 Primary extent mcell record (BSR_XTNTS - bsXtntRT)

2.9.1.1 Change mcellCnt to a uint32T

In firstXtnt (bfPrimXT) change mcellCnt from a uint16T to a uint32T.

2.9.1.1.1 Benefit

This is a counter of how many mcells are in the chain. This would stop overflows on this field for highly fragmented files.

2.9.1.1.2 Cost

Will cause byte alignment issues. (See **2.9.1.2** for solution)

2.9.1.2 Add a uint16T reserved field

In firstXtnt (bfPrimXT) add a uint16T reserved field.

2.9.1.2.1 Benefit

If we change the type of mcellCnt we will get an alignment error. This will fix the byte alignment problem as well as give us a reserved field to be used later.

2.9.1.3 Change types on reserved fields

This record has two reserved fields, a uint32T and a bfMCIdT. These should be changed to 3 uint32T.

2.9.1.3.1 Benefit

More flexibility in the use of the reserved fields for the future.

2.9.1.4 Combine bfPrimXt and bsXtntRT into a single record.

In earlier releases of Tru64 the bfPrimXt was a union of multiple structures. At the point it made sense to have it as a separate structure. Now that it is no longer a union, the fields in that record should be folded into the bsXtntRT.

2.9.1.4.1 Benefit

Easier to understand the structure.

2.9.1.5 Remove type field.

With the removal of the striped files, this field is not longer needed. As part of the clean up we can also remove the enum type bsXtntMapTypeT.

2.9.2 Bit file Attribute Record (BSR_ATTR - bsBfAttrT)

Structure changes: 3.2.10, 3.5.1

2.9.2.1 New reserved fields

Add four uint32T reserved fields at the end of the record for future changes.

2.9.2.2 Snapshot related fields

There are 5 different snapshot related fields in this record. Snapshots are not scheduled to be implemented in the first release, but will likely be added in the second release. So reserve the space in the structure for these fields, based on the current implementation of snapshots.

2.9.2.2.1 Rename clones to snapshots

All the references to clones in the ODS will be replaced with the term snapshot. This is to better reflect how this functionality will be used.

2.9.2.3 Move deleteWithClone flag to tag file

Remove the deleteWithClones flag from the tag's attribute record, instead store it in the bit flag field in the tag file.

2.9.2.3.1 Benefit

When a snapsnot is deleted the kernel would not have to read a page for each tag to get it's attribute record. (See section **2.5.5**)

2.9.2.4 Move outOfSyncClone flag to tag file

Remove the outOfSyncClone flag from the tag's attribute record, instead store it in the bit flag field in the tag file. (See section **2.5.5**)

2.9.2.4.1 Benefit

When the system is out of space and a COW is received, AdvFS might not be able to add the snapshot's primary mcell which is needed to store the outOfSyncClone flag.

2.9.2.5 Change ftxIdT record type

Change the ftxIdT type from a uint32T to uint64T, to improve CFS recovery.

2.9.3 Volume Attribute record (BSR_VD_ATTR - bsVdAttrT)

Structure changes: 3.2.11

2.9.3.1 Increase the size of vdBlkCnt

Change the type of vdBlkCnt from uint32T to uint64T.

2.9.3.1.1 Benefit

When AdvFS goes to larger volumes size we will need this.

2.9.3.2 New reserved field

Add five uint32T reserved field at the end of the record for future changes.

2.9.3.3 Rename reserved field

Rename jays_new_field (uint16T) to standard reserve name.

2.9.3.4 Store the volume's World Wide Id

Add a field to store the storage devices WWID.

2.9.3.4.1 Cost

On mount the kernel will need to check the WWID to see if matches the one stored one disk. If different the kernel will update the record with the new WWID. This will handle the case of dual mounts, or if the WWID changes for any other reason.

2.9.4 Domain Attribute (not mutable) record (BSR_DMN_ATTR - bsDmnAttrT)

Structure changes: 3.2.12

2.9.4.1 New reserved field

Add two uint32T reserved field at the end of the record for future changes.

2.9.4.2 Store the master fsCookie

Keep the master copy of the fsCookie, it will not change once it is created.

See section **2.1.10** for more info.

2.9.5 Extra Extent record (BSR_XTRA_XTNTS - bsXtraXtntRT)

Structure changes: 3.2.13

2.9.5.1 New reserved field

Add two uint32T reserved field at the end of the record for future changes.

2.9.5.1.1 Cost

BMT_XTRA_XTNTS will need to reflect the rsvd fields, as it will decrees the maximum number of extents per record.

2.9.6 Shadow Extent record (BSR_SHADOW_XTNTS - bsShadowXtntT)

Structure changes: 3.2.14

2.9.6.1 Remove striped file functionality

Hardware solutions do this faster and cleaner than kernel can.

2.9.6.1.1 Benefit

The ODS structure (bsShadowXtntT) for striped files can be confusing, and have been known to lead to programming errors which have caused corruptions and file system panics.

2.9.7 Free Mcell List Record (BSR_MCELL_FREE_LIST - bsMcellFreeListT)

Structure changes: 3.2.15

2.9.7.1 New reserved field

Add a uint32T reserved field at the end of the record for future changes.

2.9.8 Fileset Attribute record (BSR_BFS_ATTR - bsBfSetAttrT)

Structure changes: 3.2.16

2.9.8.1 Remove unused field

Remove oldQuotaStatus, in Tru64 this value is never set, but AdvFS still sets quotaAttr.quotaStatus from this field.

Remove fsContext[], is used today in a trivial way that can easily be eliminated. This is an array of 8 elements, so removing it save a fair amount of space.

2.9.8.2 New fields to track file usage

Add three new fields (uint64T) to keep track of how many extents, files and directories are in the fileset. It is acceptable if these numbers are slightly off, so some sort of async update will be used.

Fixfdmn (fsck_advfs) will be modified to add a flag which would reset these fields to the exact count.

2.9.8.2.1 Benefit

Would be used to give time estimates (time to completion) on tools such as verify and salvage.

2.9.8.2.2 Cost

Unless we do an async update, these fields would need to be update on every file creation or deletion that would add additional disk I/Os. So research is needed on how to implement a way to get this data out to disk with out adding a large number of writes to

the disk. The structure will added, but until a low cost method of updating the record, the fields will not be used.

2.9.8.3 New reserved field

Add two uint32T and two uint64T reserved field at the end of the record for future changes.

2.9.8.4 Snapshot related fields

There are 5 different snapshot related fields in this record. Snapshots are not scheduled to be implemented in the first release, but will likely be added in the second release. So reserve the space in the structure for these fields, based on the current implementation of snapshots.

All the references to clones in the ODS will be replaced with the term snapshot. This is to better reflect how this functionality will be used.

All the snapshot related fields will be clumped together, near the end of the record. This improves the readability of the data structure.

2.9.9 Volume IO Record (BSR_VD_IO_PARAMS - vdIoParamsT)

Structure changes: 3.2.17

2.9.9.1 Change unused fields into reserved field

The following fields currently are not used:

- qtoDev
- consolidate
- blockingFact
- lazyThresh

2.9.9.2 New reserved field

Add two uint32T reserved field at the end of the record for future changes.

2.9.10 Deferred Delete List Record (BSR_DEF_DEL_MCELL_LIST - delLinkT)

Structure changes: 3.2.5

2.9.10.1 Add an additional DDL for the RBMT

The main DDL which is stored in the BMT at page 0, mcell 0, is used to remove mcells from the BMT. It is not used to remove mcells from the RBMT.

For improvements such as log isolation, a way to remove and truncate mcells chain from the RBMT is needed. In the current prototype for log isolation on Tru64, a second DDL was added in the RBMT. Based on that prototype this seems to be the best place to store it.

2.9.10.2 New reserved field

Add a uint32T reserved field at the end of the record for future changes.

2.9.11 Domain Mutable Attribute record (BSR_DMN_MATTR - bsDmnMAttrT)

Structure changes: 3.2.19

2.9.11.1 New reserved field

Add two uint64T reserved field at the end of the record for future changes.

2.9.11.2 Store the ODS version number

See **2.6.3** for why this should be done.

This record must be the very first field in this record. This is to allow the record to change in the future, but still be able to find the version number on all versions of the ODS.

2.9.12 Bitfile inheritable attribute(BSR_BF_INHERIT_ATTR - bsBfInheritAttrT)

Structure changes: 3.2.9

AKA the Bitfile client attributes record (bsBfClAttrT).

This record type is used in two different ways, the first is it included in the bsBfAttrT record, it is also used for inheritable attributes.

2.9.12.1 Remove unused fields

The following fields are not used, so delete them.

- extendSize
- clientArea Array
- optServices

- rsvd_sec3

2.9.12.2 Change the name of the acl to plParent

The field acl is a currently not used, but the new implementation of property lists will need a way to keep track of which tag this property list belongs to. As this was most likely the original idea behind this field, change the name to better reflect its usage.

2.9.12.3 Use common naming scheme for reserved fields

Make all the reserved fields use the same style of variable names.

2.9.12.4 Move dataSafety field to the tag file

Remove the dataSafety flag from this record, instead store it in the bit flag field in the tag file. (See section **2.5.5**)

2.9.13 Per-fileset quota attributes (BSR_BFS_QUOTA_ATTR - bsQuotaAttrT)

Structure changes: 3.2.21

If we go to a single fileset model then these quotas are not needed.

Quota's per fileset may still have usefulness in a snapshot case.

Customers do use this functionality today, and will miss it if it goes away.

Based on the discussion during a review it was decided to leave the record in the kernel, but it will not be active. That way if it is needed for snapshots, or customers complain we will be able to re-activate this functionality with out changing the ODS.

2.9.13.1 Use common naming scheme for reserved fields

The 4 reserved fields in the structure use a different naming scheme than most of the other structures.

2.9.13.2 Replace the hi and low fields with single 64bit field

On Tru64 the quotas were implemented in such a way that they were broken in to the high 32 bits and the low 32 bits. There is no longer a reason to do this, and replacing them with a single 64 bit entry makes the structure easier to understand.

2.9.14 Property List Header Record (BSR_PROPLIST_HEAD - bsPropListHeadT)

Structure changes: 3.2.22

This section is TBD until the team has finalized the new Property List design.

2.9.14.1 Remove ODSv3 of structure

There are two versions of this record depending on which ODS version you are talking about. The ODSv3 version of the proplist is only supported on older AdvFS file systems, it should **not** be ported forward.

2.9.14.2 New reserved field

Add a uint32T reserved field at the end of the record for future changes.

2.9.14.3 Remove support for BSR_PL_PAGE

In earlier releases of Tru64 a special type of property list was designed which filled a complete BMT page with a single property list mcell. This type of PL has not been used since Tru64 v4.0 (vanilla).

2.9.15 Property List Data Record (BSR_PROPLIST_DATA - bsPropListPageT)

Structure changes: 3.2.23

This section is TBD until the team has finalized the new Property List design.

2.9.15.1 Rename structure to bsPropListDataT

In the PL/ACL design document the structure bsPropListPageT has been changed to bsPropListDataT. The original name has been a misnomer since its inception since the word "page" in bsPropListPageT really applies to mcells of record type BSR_PROPLIST_DATA. This change more closely aligns it with its real purpose in the kernel.

2.9.15.2 Remove ODSv3 of structure

There are two versions of this record depending on which ODS version you are talking about. The ODSv3 version of the proplist is only supported on older AdvFS file systems, it should **not** be ported forward.

2.9.15.3 New reserved field

Add a uint32T reserved field at the end of the record for future changes.

2.9.16 Domain Trans Attribute record (BSR_DMN_TRANS_ATTR - bsDmnTAttrT)

Structure changes: 3.2.24

No longer need the chainVdIndex, it is now part of the chainMCId.

2.9.17 Vfast on-disk record (BSR_DMN_SS_ATTR - bsSSDmnAttrT)

Structure changes: 3.2.25

2.9.17.1 Use common naming scheme for reserved fields

2.9.17.2 New reserved field

Add 2 uint64T reserved field at the end of the record for future changes.

2.9.17.3 Change some fields to uint64T

The following fields will be changed from uint32T to uint64T:

- ss_filesDefraged
- ss_pagesDefraged
- ss_pagesBalanced
- ss_filesIOBal
- ss_extentsConsol
- ss_pagesConsol

2.9.18 Domain Freeze Attributes (BSR_DMN_FREEZE_ATTR - bsDmnFreezeAttrT)

Structure changes: 3.2.26

2.9.18.1 Use common naming scheme for reserved fields

2.9.19 File Stat record (BMTR_FS_STAT - statT)

Structure changes: 3.3.2

StatT is a record that is a superset of the posix controlled structure stat (sys/stat.h). This structure is used by filesystems other than AdvFS. AdvFS uses most of the unused or reserve fields in the stat structure, and has added some additional ones at the end.

2.9.19.1 New reserved field

Add a uint32T and 2 uint64T reserved field at the end of the record for future changes.

2.9.19.2 Increase the size of st_nlink

Change the type of st_nlink from u_short to an uint32T.

2.9.19.2.1 Benefit

This would allow directories that have more than 64k files in them with out the number nlinks overflow.

2.9.19.3 Increase the size of st_mode

HP-UX uses a short for this value, Tru64 uses an uint32T. At minimum keep the size at uint32T. Bob Harris suggested that we might even consider changing its type to uint64T.

2.9.19.3.1 Benefit

Would give AdvFS more flags for future improvements.

2.9.20 Symbolic link data record (BMTR_FS_DATA - char *)

This is a variable length record that contains the path name of the symbolic link. If we use this record (known as FAST symlink) we are limited to 255 characters. If the symlink is longer than 255 (known as SLOW symlinks) we store the name in 1 or more pages and have extent records for those pages.

No changes suggested.

2.9.21 Trash can record (BMTR_FS_UNDEL_DIR - struct undel_dir_rec)

Structure changes: 3.3.1

This record is just a pointer to a tag number of the directory the deleted files will be moved to instead of deleted.

2.9.21.1 Redesign of trash-cans

It is believed that most users do not use the current design of trash-cans. A redesign on the current usage model might be a big win for customers, with out much change to the filesystem. This would be an overall design issue, not a ODS change.

The change would be to create a global trash-can (one per fileset) vs a trash-can per directory.

2.9.21.2 New reserved field

Add 2 uint32T reserved field at the end of the record for future changes.

2.9.21.3 Change the structure name

This structure does not follow the general naming convention used for other AdvFS structures. Change the name of the structure to bsUndelDirT.

2.9.22 fileset last sync time record (BMTR_FS_TIME - time_t)

Change the type from time_t to an int64T.

2.9.23 Directory index record (BMTR_FS_DIR_INDEX_FILE - bfTagT)

No proposed changes

2.9.24 Directory index data record (BMTR_FS_INDEX_FILE - bsIdxRecT)

Structure changes: 3.4.1

Directory indexes are stored in a separate tag. The parent directory has a pointer to the tag number of the directory index. If this pointer becomes corrupt there is no easy way to tell which directory the directory index belonged to.

2.9.24.1 Store parent directory's tag number

2.9.24.1.1 Costs

Extra tag number (8 bytes) must be stored for each DI.

2.9.24.2 Change long to unit64T

2.10 New Mcell Records

2.10.1 Utilities Run Time Stamp

Structure changes: 4.2

A record for each file system (maybe per fileset). The record will contain multiple time-stamps to keep track of when the last time certain tools have been run on the file system. For sure we want a time-stamp for salvage, and two for fsck_advfs (unmounted and active). I would recommend 8 time stamps in this record, this should handle any future tools that find this information helpful.

2.10.1.1 Benefits

Allow the user (or a tool) to know when the last time one of these programs has been run on the file system.

This could help us track down problems, or limit the amount of time tools take to run on file systems.

2.10.1.2 Cost

A new kernel interface would be required so online tools can read/write this field.

2.10.2 new file system name record

Structure changes: 4.7

Currently the file system name is not stored in the file system, but in the /dev/advfs directory. If the root file system becomes corrupted, or the volumes are moved to a new machine we have no way of knowing the file system name.

On mount the kernel will check the file system name with the one stored in the mattr record, if different the kernel will update the record with the new name. This will handle the case of dual mounts, or if the user changes the name of the entry in the /dev/advfs directory for any other reason.

This record will be stored in the RBMT, on the same chain as the MATTR record.

2.10.2.1 Benefit

Would be used by advscan to recreate the file system name.

2.10.2.2 Cost

A new syscall will be needed to allow this field to be modified.

2.11 Unused records

Following records are no longer needed and can be removed

- BSR_RSVD11 - bsrRsvd11T
- BSR_RSVD12 - bsrRsvd11T
- BSR_RSVD13 - bsrRsvd11T
- BSR_RSVD17 - bsrRsvd17T

2.12 Lost and Found directory

When a new fileset is created, create a directory called "lost+found" in the root of the directory. This directory will be assigned to tag 6 and will not be able to be removed. The equivalent is already done for UFS and other filesystems.

2.12.1 Benefits

This will simplify the code for tools such as fsck_advfs. If tools find files that are lost (do not have a parent directory), this gives them a standard place for the tools to place the lost files in.

2.12.2 Costs

Currently the kernel assumes that there are only 5 reserved tags. The kernel will need to be able to handle this additional reserved tag. This includes not allowing the tag to be deleted (same behavior as on the other reserved tags). Currently the define BS_BFTAG_MAX_META is stored inmsfs_kdm.h, this should be moved to bs_public.h. There might be other defines which need to be changed or looked into, for example GROUP_QUOTA_FILE_TAG.

2.13 Directory format changes

2.13.1 Remove trailing structure on each dir page

The structure dir_rec (aka dirRec) is not needed by AdvFS and just adds overhead when dealing with directories.

2.13.1.1 Benefits

Save 12 bytes per directory page.

This would allow the directory format to be the same between AdvFS and UFS.

3 Header File Changes

3.1 msfs/bs_public.h

3.1.1 bfTagT

3.1.1.1 Tru64 Version

```
typedef struct {
    uint32T num;
    uint32T seq;
} bfTagT;
```

size 8 bytes

3.1.1.2 Proposed HP-UX Version

```
typedef struct {
    uint64T tag_num;
    uint32T tag_seq;
} bfTagT;
```

size 12 bytes

3.1.2 bfMCIdT

3.1.2.1 Tru64 Version

```
typedef struct bfMCId {
    uint32T cell : 5;
    uint32T page : 27;
} bfMCIdT;
```

size 4 bytes

3.1.2.2 Proposed HP-UX Version

```
typedef struct bfMCId {
    uint64T mcid_volume  : 08;
    uint64T mcid_page    : 40;
    uint64T mcid_cell    : 16;
} bfMCIdT;
```

size 8 bytes

3.1.3 bfSetIdT

3.1.3.1 Tru64 Version

```
typedef struct {
    bfDomainIdT domainId;
    bfTagT      dirTag;
} bfSetIdT;
```

size 16 bytes

3.1.3.2 Proposed HP-UX Version

```
typedef struct {
    bfDomainIdT fset_domainId;
    bfTagT      fset_dirTag;
} bfSetIdT;
```

size 24 bytes

3.1.4 bfIdT

3.1.4.1 Tru64 Version

```
typedef struct timeval bsIdT;
```

size 8 bytes

3.1.4.2 Proposed HP-UX Version

~~**typedef struct timeval bsIdT;**~~

```
typedef struct bsId {
   int64T id_sec;
   int32T id_usec;
} bsIdT;
```

size 12 bytes

3.2 msfs/bs_ods.h

3.2.1 bsMRT

3.2.1.1 Tru64 Version

```
typedef struct bsMR {
    uint32T bCnt    : 16;
    uint32T type    :  8;
    uint32T version :  8;
} bsMRT;
```

size 4 bytes

3.2.1.2 Proposed HP-UX Version

```
typedef struct bsMR {
    uint32T mcr_bCnt    : 16;
    uint32T mcr_type    :  8;
    uint32T mcr_rsvd    :  8;
} bsMRT;
```

size 4 bytes

3.2.2 bsMCT

3.2.2.1 Tru64 Version

```
#define BSC_R_SZ \
    (292 - sizeof(bfMCIdT) - (2*sizeof(bfTagT)) - (2*sizeof(uint16T)))

#define BS_USABLE_MCELL_SPACE   (BSC_R_SZ - 2*sizeof( bsMRT ))

typedef struct bsMC {
    bfMCIdT nextMCId;
    uint16T nextVdIndex;
    uint16T linkSegment;
    bfTagT tag;
    bfTagT bfSetTag;
    char bsMR0[BSC_R_SZ];
} bsMCT;
```

BSC_R_SZ = 268
BS_USABLE_MCELL_SPACE = 260
size 24 + BSC_R_SZ bytes = 292

3.2.2.2 Proposed HP-UX Version

```
#define BSC_R_SZ \
    (354 - sizeof(bfMCIdT) - (2*sizeof(bfTagT)) - (2*sizeof(uint16T))-
((2*sizeof(uint32T))

#define BS_USABLE_MCELL_SPACE   (BSC_R_SZ - 2*sizeof( bsMRT ))

typedef struct bsMC {
    bfMCIdT mc_nextMCId;
    ~~uint16T nextVdIndex;~~
    uint32T mc_linkSegment;
    bfTagT mc_tag;
    bfTagT mc_bfSetTag;
    uint16T mc_numRecords;
    uint16T mc_rsvd1;      /* Needed for byte alignment */
    uint32T mc_rsvd2;
    char mc_bsMR0[BSC_R_SZ];
} bsMCT;
```

BSC_R_SZ = 310
BS_USABLE_MCELL_SPACE = 302
size 44 + BSC_R_SZ bytes = 354 bytes

3.2.3 bsMPgT

3.2.3.1 Tru64 Version

```
#define BSPG_CELLS \
  ((ADVFS_PGSZ - (3 * sizeof(uint32T) + sizeof(bfMCIdT))) \
  / sizeof(struct bsMC))

typedef struct bsMPg {
    bfMCIdT nextfreeMCId;
    uint32T nextFreePg;
```

```
    uint32T freeMcellCnt;
    uint32T pageId : 27;
    uint32T megaVersion: 5;
    struct bsMC bsMCA[BSPG_CELLS];
} bsMPgT;
```

BSPG_CELLS = 28
size 8 kbytes

3.2.3.2 Proposed HP-UX Version

```
#define BSPG_CELLS \
  ((ADVFS_METADATA_PGSZ - (3 * sizeof(uint32T) + 2 * sizeof(uint64T) +
\
                  sizeof(bfMCIdT) + sizeof(bfFsCookieT)) \
  / sizeof(struct bsMC)))

typedef struct bsMPg {
    uint32T bmt_magicNumber;
    bfFsCookieT bmt_fsCookie;
    bfMCIdT bmt_nextfreeMCId;
    uint64T bmt_nextFreePg   : 40;
    uint64T bmt_freeMcellCnt : 16;
    uint64T bmt_rsvd1        : 08;
    uint64T bmt_pageId : 40;
    uint64T bmt_rsvd2  : 24;
    uint32T bmt_rsvd3;
    uint32T bmt_rsvd4;
    struct bsMC bmt_bsMCA[BSPG_CELLS];
} bsMPgT;
```

BSPG_CELLS = 23
size 8190 bytes

3.2.4 bsXtntT

3.2.4.1 Tru64 Version

```
typedef struct bsXtnt {
    uint32T bsPage;
    uint32T vdBlk;
} bsXtntT;
```

size 8 bytes

3.2.4.2 Proposed HP-UX Version

```
typedef struct bsXtnt {
    uint64T xt_bsPage;
    uint64T xt_vdBlk;
} bsXtntT;
```

size 16 bytes

3.2.5 delLinkT (BSR_DEF_DEL_MCELL_LIST)

3.2.5.1 Tru64 Version

```
typedef struct {
    bfMCIdT nextMCId;
    bfMCIdT prevMCId;
} delLinkT;

typedef delLinkT delLinkRT;
```

size 8 bytes

3.2.5.2 Proposed HP-UX Version

```
typedef struct {
    bfMCIdT dl_nextMCId;
    bfMCIdT dl_prevMCId;
    uint32T pxt_rsvd1;
} delLinkT;

typedef delLinkT delLinkRT;
```

size 20 bytes

3.2.6 delRstT

3.2.6.1 Tru64 Version

```
typedef uint16T vdIndexT;

typedef struct {
    bfMCIdT mcid;
    vdIndexT vdIndex;
    uint32T xtntIndex;
    uint32T offset;
    uint32T blocks;
} delRstT;
```

size 18 bytes

3.2.6.2 Proposed HP-UX Version

```
typedef struct {
    bfMCIdT dr_mcid;
    vdIndexT vdIndex;
    uint32T dr_xtntIndex;
    uint32T dr_offset;
    uint32T dr_blocks;
```

```
} delRstT;
```

size 20 bytes

3.2.7 bfPrimXT

3.2.7.1 Tru64 Version

```
#define BMT_XTNTS 2

typedef struct {
    uint16T mcellCnt;
    uint16T xCnt;
    bsXtntT bsXA[BMT_XTNTS];
} bfPrimXT;
```

size 20 bytes

3.2.7.2 Proposed HP-UX Version

This record will be removed, all of its fields will be included in the bsXtntRT record.

3.2.8 bsXtntRT (BSR_XTNTS)

3.2.8.1 Tru64 Version

```
typedef enum {
    BSXMT_APPEND,    /* Append only bitfile */
    BSXMT_SHADOW_UNSUPPORTED, /* Shadowed file; not supported currently
*/
    BSXMT_STRIPE     /* Striped bitfile */
} bsXtntMapTypeT;

typedef struct bsXtntR {
    bsXtntMapTypeT type;
    uint32T chainVdIndex;
    bfMCIdT chainMCId;
    uint32T rsvd1;
    bfMCIdT rsvd2;
    uint32T blksPerPage;
    uint32T segmentSize;
    delLinkT delLink;
    delRstT delRst;
    bfPrimXT firstXtnt;
} bsXtntRT;
```

size 74 bytes

3.2.8.2 Proposed HP-UX Version

```
#define BMT_XTNTS 2
```

```
typedef enum {
    BSXMT_APPEND,     /* Append only bitfile */{
    BSXMT_SHADOW_UNSUPPORTED, /* Shadowed file; not supported currently
*/{
    BSXMT_STRIPE      /* Striped bitfile */{
} bsXtntMapTypeT;{

typedef struct bsXtntR {
    bsXtntMapTypeT type;
    uint32T chainVdIndex;
    bfMCIdT pxt_chainMCId;
    uint32T pxt_mcellCnt;
    uint32T rsvd1;
    bfMCIdT rsvd2;
    uint32T pxt_blksPerPage;
    uint32T pxt_segmentSize;
    delLinkT pxt_delLink;
    delRstT pxt_delRst;
    bfPrimXT firstXtnt;
    uint16T pxt_xCnt;
    uint32T pxt_rsvd1;
    uint32T pxt_rsvd2;
    uint32T pxt_rsvd3;
    bsXtntT pxt_bsXA[BMT_XTNTS];

} bsXtntRT;
```

size 106 bytes

3.2.9 bsBfClAttrT (AKA bsBfInheritAttrT) (BSR_BF_INHERIT_ATTR)

3.2.9.1 Tru64 Version

```
typedef struct bsBfClAttr {
    bfDataSafetyT dataSafety;
    serviceClassT reqServices;
    serviceClassT optServices;
    int32T extendSize;
    int32T clientArea[BS_CLIENT_AREA_SZ];
    int32T rsvd1;
    int32T rsvd2;
    bfTagT acl;
    int32T rsvd_sec1;
    int32T rsvd_sec2;
    int32T rsvd_sec3;
} bsBfClAttrT;

typedef bsBfClAttrT bsBfInheritAttrT;
```

size 60 bytes

3.2.9.2 Proposed HP-UX Version

```
typedef struct bsBfClAttr {
    bfDataSafetyT dataSafety;
    serviceClassT bcl_reqServices;
    bfTagT bcl_plParent;
    serviceClassT optServices;
    int32T extendSize;
    int32T clientArea[BS_CLIENT_AREA_SZ];
    int32T bcl_rsvd1;
    int32T bcl_rsvd2;
    bfTagT acl;
    int32T bcl_rsvd3;
    int32T bcl_rsvd4;
} bsBfClAttrT;

typedef bsBfClAttrT bsBfInheritAttrT;
```

size 32 bytes

3.2.10 bsBfAttrT (BSR_ATTR)

3.2.10.1 Tru64 Version

```
typedef struct bsBfAttr {
    bfStatesT state;
    uint32T bfPgSz;
    ftxIdT transitionId;
    uint32T cloneId;
    uint32T cloneCnt;
    uint32T maxClonePgs;
    int16T deleteWithClone;
    int16T outOfSyncClone;
    bsBfClAttrT cl;
} bsBfAttrT;
```

size 88 bytes

3.2.10.2 Proposed HP-UX Version

```
typedef struct bsBfAttr {
    bfStatesT bfa_state;
    uint32T bfa_bfPgSz;
    ftxIdT bfa_transitionId;
    uint32T bfa_snapShotId;
    uint32T bfa_snapShotCnt;
    uint32T bfa_maxSnapShotPgs;
    int16T  deleteWithClone;
    int16T  outOfSyncClone;
    bsBfClAttrT bfa_cl;
    uint32T bfa_rsvd1;
    uint32T bfa_rsvd2;
    uint32T bfa_rsvd3;
    uint32T bfa_rsvd4;
} bsBfAttrT;
```

size 72 bytes

3.2.11 bsVdAttrT (BSR_VD_ATTR)

3.2.11.1 Tru64 Version

```
typedef struct bsVdAttr {
    bsIdT vdMntId;
    bsVdStatesT state;
    vdIndexT vdIndex;
    uint16T jays_new_field;
    uint32T vdBlkCnt;
    uint32T stgCluster;
    uint32T maxPgSz;
    uint32T bmtXtntPgs;
    serviceClassT serviceClass;
} bsVdAttrT;
```

size 36 bytes

3.2.11.2 Proposed HP-UX Version

```
typedef struct bsVdAttr {
    bsIdT vda_vdMntId;
    bsVdStatesT vda_state;
    vdIndexT vda_vdIndex;
    uint16T vda_rsvd1;        /* Needed for byte alignment*/
    ~~uint32T vdBlkCnt;~~
    ~~uint32T stgCluster;~~
    ~~uint32T maxPgSz;~~
    uint32T vda_blkSz;        /* block size */
    uint64T vda_blkCnt;       /* number blocks on this volume */
    uint32T vda_sbmBlksBit;   /* SBM blocks to bit */
    uint32T vda_blksUserPage; /* Variable User Page Size */
    uint32T vda_blksMetaPage; /* Variable Meta Page Size */
    uint64T vda_worldWideId;  /* Storage Device WWID */
    uint32T vda_bmtXtntPgs;
    serviceClassT vda_serviceClass;
    uint32T vda_rsvd2;
    uint32T vda_rsvd3;
    uint32T vda_rsvd4;
    uint32T vda_rsvd5;
    uint32T vda_rsvd6;
} bsVdAttrT;
```

size 80 bytes

3.2.12 bsDmnAttrT (BSR_DMN_ATTR)

3.2.12.1 Tru64 Version

```
typedef struct bsDmnAttr {
```

```
    bsIdT bfDomainId;
    uint32T maxVds;
    bfTagT bfSetDirTag;
} bsDmnAttrT;
```

size 20 bytes

3.2.12.2 Proposed HP-UX Version

```
typedef struct bsDmnAttr {
    bsIdT dna_bfDomainId;
    bfFsCookieT dna_fsCookie;
    uint32T dna_maxVds;
    bfTagT dna_bfSetDirTag;
    uint32T dna_rsvd1;
    uint32T dna_rsvd2;
} bsDmnAttrT;
```

size 48 bytes

3.2.13 bsXtraXtntRT (BSR_XTRA_XTNTS)

3.2.13.1 Tru64 Version

```
#define BMT_XTRA_XTNTS ((BSC_R_SZ - 2*sizeof( bsMRT ) - \
                        (2 * sizeof( uint16T ))) / sizeof( bsXtntT ))

typedef struct bsXtraXtntR {
    uint16T blksPerPage;
    uint16T xCnt;
    bsXtntT bsXA[BMT_XTRA_XTNTS];
} bsXtraXtntRT;
```

BMT_XTRA_XTNTS = 32
size 260 bytes

3.2.13.2 Proposed HP-UX Version

```
#define BMT_XTRA_XTNTS ((BSC_R_SZ - 2*sizeof( bsMRT ) - \
                         (2 * sizeof( uint16T )) - (2 * sizeof (
uint32T ))) \
                        / sizeof( bsXtntT ))

typedef struct bsXtraXtntR {
    uint16T xxt_blksPerPage;
    uint16T xxt_xCnt;
    uint32T xxt_rsvd1;
    uint32T xxt_rsvd2;
    bsXtntT xxt_bsXA[BMT_XTRA_XTNTS];
} bsXtraXtntRT;
```

BMT_XTRA_XTNTS = 18
size 12 + (16 * BMT_XTRA_XTNTS) = 300 Bytes

3.2.14 bsShadowXtntT (BSR_SHADOW_XTNTS)

3.2.14.1 Tru64 Version

```
#define BMT_SHADOW_XTNTS \
    ((BSC_R_SZ - 2*sizeof( bsMRT ) - sizeof (vdIndexT) - \
      (3 * sizeof( uint16T ))) / sizeof( bsXtntT ))

typedef struct bsShadowXtnt {
    vdIndexT allocVdIndex;
    uint16T mcellCnt;
    uint16T blksPerPage;
    uint16T xCnt;
    bsXtntT bsXA[BMT_SHADOW_XTNTS];
} bsShadowXtntT;
```

BMT_SHADOW_XTNTS = 31
size 256 bytes

3.2.14.2 Proposed HP-UX Version

Record will be deleted, AdvFS will not support striped files.

3.2.15 bsMcellFreeListT (BSR_MCELL_FREE_LIST)

3.2.15.1 Tru64 Version

```
typedef struct {
    uint32T headPg;
} bsMcellFreeListT;
```

size 4 bytes

3.2.15.2 Proposed HP-UX Version

```
typedef struct {
    uint32T mfl_headPg;
    uint32T mfl_rsvd1;
} bsMcellFreeListT;
```

size 8 bytes

3.2.16 bsBfSetAttrT (BSR_BFS_ATTR)

3.2.16.1 Tru64 Version

```
#define BFS_FRAG_MAX 8
#define BS_SET_NAME_SZ        32
#define BS_FS_CONTEXT_SZ       8

typedef struct {
    bfSetIdT bfSetId;
    bfTagT fragBfTag;
    bfTagT nextCloneSetTag;
    bfTagT origSetTag;
    bfTagT nxtDelPendingBfSet;
    uint16T state;
    uint16T flags;
    uint32T cloneId;
    uint32T cloneCnt;
    uint32T numClones;
    uint32T fsDev;
    uint32T freeFragGrps;
    uint32T oldQuotaStatus;
    uid_t uid;
    gid_t gid;
    mode_t mode;
    char setName[BS_SET_NAME_SZ];
    uint32T fsContext[BS_FS_CONTEXT_SZ];
    fragGrpT fragGrps[BFS_FRAG_MAX];
} bsBfSetAttrT;
```

size 216 bytes

3.2.16.2 Proposed HP-UX Version

```
#define BFS_FRAG_MAX 8
#define BS_SET_NAME_SZ        32
#define BS_FS_CONTEXT_SZ       8

typedef struct {
    bfSetIdT bsa_bfSetId;
    bfTagT fragBfTag;
    bfTagT bsa_nxtDelPendingBfSet;
    uint16T bsa_state;
    uint16T bsa_flags;
    uint32T bsa_fsDev;
    uint32T freeFragGrps;
    uint32T oldQuotaStatus;
    uid_t bsa_uid;
    gid_t bsa_gid;
    mode_t bsa_mode;
    char bsa_setName[BS_SET_NAME_SZ];
    uint32T fsContext[BS_FS_CONTEXT_SZ];
    fragGrpT fragGrps[BFS_FRAG_MAX];
    bfTagT bsa_nextSnapShotTag;
    bfTagT bsa_origSnapShotTag;
    uint32T bsa_snapShotId;
    uint32T bsa_snapShotCnt;
    uint32T bsa_numSnapShots;
    uint64T bsa_numberFiles;   /* ~ Number of file in fileset */
    uint64T bsa_numberDirs;    /* ~ Number of dirs in fileset */
```

```
    uint64T bsa_numberExtents; /* ~ Number of extents in fileset */
    uint32T bsa_rsvd1;
    uint32T bsa_rsvd2;
    uint64T bsa_rsvd3;
    uint64T bsa_rsvd4;
} bsBfSetAttrT;
```

size 172 bytes

3.2.17 vdIoParamsT (BSR_VD_IO_PARAMS)

3.2.17.1 Tru64 Version

```
typedef struct vdIoParams {
    int rdMaxIo;
    int wrMaxIo;
    int qtoDev;
    int consolidate;
    int blockingFact;
    int lazyThresh;
} vdIoParamsT;
```

size 24 bytes

3.2.17.2 Proposed HP-UX Version

```
typedef struct vdIoParams {
    uint32T vio_rdMaxIo;
    uint32T vio_wrMaxIo;
    uint32T vio_rsvd1;
    uint32T vio_rsvd2;
    uint32T vio_rsvd3;
    uint32T vio_rsvd4;
    uint32T vio_rsvd5;
    uint32T vio_rsvd6;
} vdIoParamsT;
```

size 32 bytes

3.2.18 bsrRsvd11T

3.2.18.1 Tru64 Version

```
typedef struct {
    int32T          r1;
    int32T          r2;
    int32T          r3;
    int32T          r4;
    int32T          r5;
    int32T          r6;
    uint32T         r7;
} bsrRsvd11T;
```

size 28 bytes

3.2.18.2 Proposed HP-UX Version

This structure type will be deleted.

3.2.19 bsDmnMAttrT (BSR_DMN_MATTR)

3.2.19.1 Tru64 Version

```
typedef struct bsDmnMAttr {
    uint32T seqNum;
    bfTagT delPendingBfSet;
    uid_t uid;
    gid_t gid;
    mode_t mode;
    uint16T vdCnt;
    uint16T recoveryFailed;
    bfTagT bfSetDirTag;
    bfTagT ftxLogTag;
    uint32T ftxLogPgs;
} bsDmnMAttrT;
```

size 48 bytes

3.2.19.2 Proposed HP-UX Version

```
typedef struct bsDmnMAttr {
    onDiskVersionT dma_ODSVersion;   /* Must be first field in record
*/
    uint32T dma_seqNum;
    bfTagT dma_delPendingBfSet;
    uid_t dma_uid;
    gid_t dma_gid;
    mode_t dma_mode;
    uint16T dma_vdCnt;
    uint16T dma_recoveryFailed;
    bfTagT dma_bfSetDirTag;
    bfTagT dma_ftxLogTag;
    uint32T dma_ftxLogPgs;
    uint64T dma_rsvd1;
    uint64T dma_rsvd2;
} bsDmnMAttrT;
```

size 80 bytes

3.2.20 bsrRsvd17T (BSR_RSVD17)

3.2.20.1 Tru64 Version

```
typedef struct {
```

```
    int32T          r1;
    int32T          r2;
    int32T          r3;
    int32T          r4;
    int32T          r5;
    int32T          r6;
    uint32T         r7;
    uint32T         r8;
} bsrRsvd17T;
```

size 32 bytes

3.2.20.2 Proposed HP-UX Version

This structure type will be deleted.

3.2.21 bsQuotaAttrT (BSR_BFS_QUOTA_ATTR)

3.2.21.1 Tru64 Version

```
typedef struct bsQuotaAttr {
    uint32T blkHLimitLo;
    uint32T blkHLimitHi;
    uint32T blkSLimitLo;
    uint32T blkSLimitHi;
    uint32T fileHLimitLo;
    uint32T fileHLimitHi;
    uint32T fileSLimitLo;
    uint32T fileSLimitHi;
    time_t  blkTLimit;
    time_t  fileTLimit;
    uint32T quotaStatus;
    uint32T unused1;
    uint32T unused2;
    uint32T unused3;
    uint32T unused4;
} bsQuotaAttrT;
```

size 60 bytes

3.2.21.2 Proposed HP-UX Version

```
typedef struct bsQuotaAttr {
    ~~uint32T blkHLimitLo;~~
    ~~uint32T blkHLimitHi;~~
    ~~uint32T blkSLimitLo;~~
    ~~uint32T blkSLimitHi;~~
    ~~uint32T fileHLimitLo;~~
    ~~uint32T fileHLimitHi;~~
    ~~uint32T fileSLimitLo;~~
    ~~uint32T fileSLimitHi;~~
    uint64T quo_blkHLimit;
    uint64T quo_blkSLimit;
```

```
    uint64T quo_fileHLimit;
    uint64T quo_fileSLimit;
    int64T  quo_blkTLimit;
    int64T  quo_fileTLimit;
    uint32T quo_quotaStatus;
    uint32T quo_rsvd1;
    uint32T quo_rsvd2;
    uint32T quo_rsvd3;
    uint32T quo_rsvd4;
} bsQuotaAttrT;
```

size 68 bytes

3.2.22 bsPropListHeadT (BSR_PROPLIST_HEAD)

3.2.22.1 Tru64 Version

```
typedef struct bsPropListHead {
  uint64T  flags;
  uint32T  pl_num;
  uint32T  spare;
  uint32T  namelen;
  uint32T  valuelen;
  char buffer[1];
} bsPropListHeadT;
```

size 24 + Variable bytes

3.2.22.2 Proposed HP-UX Version

A new form of property lists might allow us to delete this record type.

```
typedef struct bsPropListHead {
  uint64T  plh_flags;
  uint32T  plh_num;
  uint32T  plh_spare;
  uint32T  plh_namelen;
  uint32T  plh_valuelen;
  uint32T  plh_rsvd1;
  char plh_buffer[1];
} bsPropListHeadT;
```

size 28 + Variable bytes

3.2.23 bsPropListPageT (BSR_PROPLIST_DATA)

3.2.23.1 Tru64 Version

```
#define BSR_PROPLIST_PAGE_SIZE ((BSPG_CELLS*sizeof(bsMCT)) - \
                                sizeof(bfMCIdT) - \
                                (2*sizeof(bfTagT)) - \
                                (2*sizeof(uint16T)) - \
```

```
                                (2*sizeof(bsMRT)))

typedef struct bsPropListPage {
  uint32T pl_num;
  uint32T pl_seg;
  char    buffer[BSR_PROPLIST_PAGE_SIZE];
} bsPropListPageT;
```

BSR_PROPLIST_PAGE_SIZE = 8144
size 8152 bytes - **Special Case Mcell**

3.2.23.2 Proposed HP-UX Version

A new form of property lists might allow us to delete this record type.

```
#define BSR_PROPLIST_DATA_SIZE ((BSPG_CELLS*sizeof(bsMCT)) - \
                                sizeof(bfMCIdT) - \
                                (2*sizeof(bfTagT)) - \
                                (2*sizeof(uint16T)) - \
                                (2*sizeof(uint32T)) - \
                                (2*sizeof(bsMRT)))

typedef struct bsPropListData {
  uint32T pld_num;
  uint32T pld_seg;
  uint32T pld_rsvd1;
  char    pld_buffer[BSR_PROPLIST_DATA_SIZE];
} bsPropListDataT;
```

BSR_PROPLIST_PAGE_SIZE = 8090
size 8102 bytes - **Special Case Mcell**

3.2.24 bsDmnTAttrT (BSR_DMN_TRANS_ATTR)

3.2.24.1 Tru64 Version

```
typedef enum {
    BSR_VD_NO_OP,
    BSR_VD_ADDVOL,
    BSR_VD_RMVOL
} bsVdOpT;

typedef struct bsDmnTAttr {
    uint32T chainVdIndex;
    bfMCIdT chainMCId;
    bsVdOpT op;
    dev_t   dev;
} bsDmnTAttrT;
```

size 20 bytes

3.2.24.2 Proposed HP-UX Version

```
typedef enum {
    BSR_VD_NO_OP,
    BSR_VD_ADDVOL,
    BSR_VD_RMVOL
} bsVdOpT;

typedef struct bsDmnTAttr {
    ~~uint32T chainVdIndex;~~
    bfMCIdT dta_chainMCId;
    bsVdOpT dta_op;
    dev_t   dta_dev;
} bsDmnTAttrT;
```

size 20 bytes

3.2.25 bsSSDmnAttrT (BSR_DMN_SS_ATTR)

3.2.25.1 Tru64 Version

```
typedef enum
{
    SS_DEACTIVATED,
    SS_ACTIVATED,
    SS_SUSPEND,
    SS_ERROR,
    SS_CFS_RELOC,
    SS_CFS_MOUNT,
    SS_CFS_UMOUNT
} ssDmnOpT;

typedef struct bsSSDmnAttr {
    ssDmnOpT   ssDmnState;
    uint16T    ssDmnDefragment;
    uint16T    ssDmnSmartPlace;
    uint16T    ssDmnBalance;
    uint16T    ssDmnVerbosity;
    uint16T    ssDmnDirectIo;
    uint16T    ssMaxPercentOfIoWhenBusy;
    uint32T    ssFilesDefraged;
    uint32T    ssPagesDefraged;
    uint32T    ssPagesBalanced;
    uint32T    ssFilesIOBal;
    uint32T    ssExtentsConsol;
    uint32T    ssPagesConsol;
    uint16T    ssAccessThreshHits;
    uint16T    ssSteadyState;
    uint16T    ssMinutesInHotList;
    uint16T    ssReserved0;
    uint32T    ssReserved1;
    uint32T    ssReserved2;
} bsSSDmnAttrT;
```

size 56 bytes

3.2.25.2 Proposed HP-UX Version

```
typedef enum
{
    SS_DEACTIVATED,
    SS_ACTIVATED,
    SS_SUSPEND,
    SS_ERROR,
    SS_CFS_RELOC,
    SS_CFS_MOUNT,
    SS_CFS_UMOUNT
} ssDmnOpT;

typedef struct bsSSDmnAttr {
    ssDmnOpT   ss_dmnState;
    uint16T    ss_dmnDefragment;
    uint16T    ss_dmnSmartPlace;
    uint16T    ss_dmnBalance;
    uint16T    ss_dmnVerbosity;
    uint16T    ss_dmnDirectIo;
    uint16T    ss_maxPercentOfIoWhenBusy;
    uint64T    ss_filesDefraged;
    uint64T    ss_pagesDefraged;
    uint64T    ss_pagesBalanced;
    uint64T    ss_filesIOBal;
    uint64T    ss_extentsConsol;
    uint64T    ss_pagesConsol;
    uint16T    ss_accessThreshHits;
    uint16T    ss_steadyState;
    uint16T    ss_minutesInHotList;
    uint16T    ss_rsvd0;
    uint32T    ss_rsvd1;
    uint32T    ss_rsvd2;
    uint64T    ss_rsvd3;
    uint64T    ss_rsvd4;
} bsSSDmnAttrT;
```

size 96 bytes

3.2.26 bsDmnFreezeAttrT (BSR_DMN_FREEZE_ATTR)

3.2.26.1 Tru64 Version

```
typedef struct bsDmnFreezeAttr {
    time_t  freezeBegin;
    time_t  freezeEnd;
    uint32T freezeCount;
    uint32T unused1;
    uint32T unused2;
    uint32T unused3;
    uint32T unused4;
} bsDmnFreezeAttrT;
```

size 28 bytes

3.2.26.2 Proposed HP-UX Version

```
typedef struct bsDmnFreezeAttr {
    int64T  dfa_freezeBegin;
    int64T  dfa_freezeEnd;
    uint32T dfa_freezeCount;
    uint32T dfa_rsvd1;
    uint32T dfa_rsvd2;
    uint32T dfa_rsvd3;
    uint32T dfa_rsvd4;
} bsDmnFreezeAttrT;
```

size 36 bytes

3.2.27 bsStgBmT

3.2.27.1 Tru64 Version

```
#define SBM_LONGS_PG  \
     ((ADVFS_PGSZ - 2 * sizeof(uint32T)) / sizeof(uint32T))

typedef struct bsStgBm {
    uint32T lgSqNm;
    uint32T xor;
    uint32T mapInt[SBM_LONGS_PG];
} bsStgBmT;
```

SBM_LONGS_PG = 2046
size 8 Kbytes

3.2.27.2 Proposed HP-UX Version

```
#define SBM_LONGS_PG  \
     ((ADVFS_METADATA_PGSZ - (3 * sizeof(uint32T))  -
(sizeof(bfFsCookieT))) / sizeof(uint32T))

typedef struct bsStgBm {
    uint32T sbm_magicNumber;
    bfFsCookieT sbm_fsCookie;
    uint32T sbm_pageNumber;
    uint32T sbm_xor;
    uint32T sbm_mapInt[SBM_LONGS_PG];
} bsStgBmT;
```

SBM_LONGS_PG = 2042
size 8 Kbytes

3.2.28 bsTMapT

3.2.28.1 Tru64 Version

```
typedef struct bsTMap {
    union {
        struct {
            uint32T freeList;
            uint32T unInitPg;
        } tm_s1;

        struct {
            uint16T seqNo;
            uint16T unused;
            uint32T nextMap;
        } tm_s2;

        struct {
            uint16T seqNo;
            uint16T vdIndex;
            bfMCIdT bfMCId;
        } tm_s3;
    } tm_u;
} bsTMapT;
```

size 8 bytes

3.2.28.2 Proposed HP-UX Version

```
typedef struct bsTMap {
    union {
        struct {
            uint32T s1_flags;
            uint32T s1_freeList;
            uint32T s1_unInitPg;
            uint32T s1_rsvd;
        } tm_s1;

        struct {
            uint32T s2_flags;
            ~~uint16T seqNo;~~
            ~~uint16T unused;~~
            uint32T s2_nextMap;
            uint64T s2_rsvd;
        } tm_s2;

        struct {
            uint32T s3_flags;
            uint32T s3_seqNo;
            ~~uint16T vdIndex;~~
            bfMCIdT s3_bfMCId;
        } tm_s3;
    } tm_u;
} bsTMapT;
```

size 16 bytes

3.2.29 bsTDirPgHdrT

3.2.29.1 Tru64 Version

```
typedef struct bsTDirPgHdr {
    uint32T currPage;
    uint32T nextFreePage;
    uint16T nextFreeMap;
    uint16T numAllocTMaps;
    uint16T numDeadTMaps;
    uint16T padding;
} bsTDirPgHdrT;
```

size 20 bytes

3.2.29.2 Proposed HP-UX Version

```
typedef struct bsTDirPgHdr {
    uint32T tdh_magicNumber;
    bfFsCookieT tdh_fsCookie;
    uint32T tdh_currPage;
    uint32T tdh_nextFreePage;
    uint32T tdh_nextFreeMap;
    uint32T tdh_numAllocTMaps;
    uint32T tdh_numDeadTMaps;
    ~~uint16T padding;~~
    uint32T tdh_rsvd1;
} bsTDirPgHdrT;
```

size 32 bytes

3.2.30 bsTDirPgT

3.2.30.1 Tru64 Version

```
#define BS_TD_TAGS_PG ((ADVFS_PGSZ - \
                        (sizeof(bsTDirPgHdrT))) / sizeof(bsTMapT))

typedef struct bsTDirPg {
    struct bsTDirPgHdr tpPgHdr;
    struct bsTMap tMapA[BS_TD_TAGS_PG];
} bsTDirPgT;
```

BS_TD_TAGS_PG = 1021
size 8 Kbytes

3.2.30.2 Proposed HP-UX Version

```
#define BS_TD_TAGS_PG ((ADVFS_METADATA_PGSZ - \
                        (sizeof(bsTDirPgHdrT))) / sizeof(bsTMapT))

typedef struct bsTDirPg {
    struct bsTDirPgHdr tp_pgHdr;
    struct bsTMap tp_tMapA[BS_TD_TAGS_PG];
```

```
} bsTDirPgT;
```

BS_TD_TAGS_PG = 510
size 8 Kbytes

3.2.31 logPgHdrT

3.2.31.1 Tru64 Version

```
typedef struct logPgHdr {
    lsnT thisPageLSN;
    bfdBfMetaT pgType;
    bfDomainIdT dmnId;
    uint16T pgSafe;
    uint16T chkBit;
    int16T curLastRec;
    int16T prevLastRec;
    logRecAddrT firstLogRec;
} logPgHdrT;
```

size 32 bytes

3.2.31.2 Proposed HP-UX Version

```
typedef struct logPgHdr {
    uint32T lph_magicNumber;
    lsnT lph_thisPageLSN;
    bfdBfMetaT lph_pgType;
    bfFsCookieT lph_fsCookie;
    bfDomainIdT lph_dmnId;
    uint16T lph_pgSafe;
    uint16T lph_chkBit;
    int16T lph_curLastRec;
    int16T lph_prevLastRec;
    uint32T lph_rsvd1;
    uint32T lph_rsvd2;
    logRecAddrT lph_firstLogRec;
} logPgHdrT;
```

size 60 bytes

3.2.32 logPgTrlrT

3.2.32.1 Tru64 Version

```
#define ADVFS_PGSZ_IN_BLKS     (ADVFS_PGSZ / BS_BLKSIZE)

typedef struct logPgTrlr {
    lsnT lsnOverwriteVal[ADVFS_PGSZ_IN_BLKS - 1];
    bfdBfMetaT pgType;
    bfDomainIdT dmnId;
} logPgTrlrT;
```

ADVFS_PGSZ_IN_BLKS = 16
size 72 bytes

3.2.32.2 Proposed HP-UX Version

```
#define ADVFS_METADATA_PGSZ_IN_BLKS   (ADVFS_METADATA_PGSZ /
BS_BLKSIZE)

typedef struct logPgTrlr {
    lsnT lpt_lsnOverwriteVal[ADVFS_METADATA_PGSZ_IN_BLKS - 1];
    bfdBfMetaT lpt_pgType;
    bfDomainCookieT lpt_domainCookie;
} logPgTrlrT;
```

ADVFS_METADATA_PGSZ_IN_BLKS = 16
size 76 bytes

3.2.33 logPgT

3.2.33.1 Tru64 Version

```
#define DATA_WORDS_PG \
    howmany( (ADVFS_PGSZ - sizeof(logPgHdrT) - sizeof(logPgTrlrT)), \
             sizeof(uint32T) )

typedef struct logPg {
    logPgHdrT  hdr;
    uint32T    data[DATA_WORDS_PG];
    logPgTrlrT trlr;
} logPgT;
```

DATA_WORDS_PG = 2022
size 8 Kbytes

3.2.33.2 Proposed HP-UX Version

```
#define DATA_WORDS_PG \
    howmany( (ADVFS_METDATA_PGSZ - sizeof(logPgHdrT) -
sizeof(logPgTrlrT)), \
             sizeof(uint32T) )

typedef struct logPg {
    logPgHdrT  log_hdr;
    uint32T    log_data[DATA_WORDS_PG];
    logPgTrlrT log_trlr;
} logPgT;
```

DATA_WORDS_PG = 2011
size 8 Kbytes

3.3 msfs/fs_dir.h

3.3.1 undel_dir_rec (BMTR_FS_UNDEL_DIR)

3.3.1.1 Tru64 Version

```
struct undel_dir_rec {
    bfTagT dir_tag;
};
```

size 8 bytes

3.3.1.2 Proposed HP-UX Version

```
bsUndelDirT.

typedef struct bsUndelDir {
    bfTagT udd_tag;
    uint32T udd_rsvd1;
    uint32T udd_rsvd2;
} bsUndelDirT;
```

size 20 bytes

3.3.2 fs_stat

3.3.2.1 Tru64 Version

```
struct fs_stat
{
    bfTagT  st_ino;
    mode_t  st_mode;
    uid_t   st_uid;
    gid_t   st_gid;
    dev_t   st_rdev;
    off_t   st_size;
#ifdef __arch32__
    uint32T st_size_filler;
#endif
    time_t  st_atime;
    int     st_uatime;
    time_t  st_mtime;
    int     st_umtime;
    time_t  st_ctime;
    int     st_uctime;
    uint_t  st_flags;
    bfTagT  dir_tag;
    bfFragIdT fragId;
    u_short st_nlink;
    short   st_unused_1;
    uint32T fragPageOffset;
    uint32T st_unused_2;
};

typedef struct fs_stat statT;
```

size 88 bytes

3.3.2.2 Proposed HP-UX Version

```
struct fs_stat
{
    bfTagT  st_ino;
    uint64T st_mode;
    uid_t   st_uid;
    gid_t   st_gid;
    dev_t   st_rdev;
    off_t   st_size;
#ifdef __arch32__
    uint32T st_size_filler;
#endif
    int64T  st_atime;
    int     st_uatime;
    int64T  st_mtime;
    int     st_umtime;
    int64T  st_ctime;
    int     st_uctime;
    uint_t  st_flags;
    bfTagT  dir_tag;
    bfFragIdT fragId;
    uint32T  st_nlink;
    short   st_unused_1;
    uint32T fragPageOffset;
    uint32T st_unused_2;
    uint32T st_rsvd1;
    uint32T st_rsvd2;
    uint64T st_rsvd3;
    uint64T st_rsvd4;
};

typedef struct fs_stat statT;
```

size 120 bytes

3.4 msfs/fs_index.h

3.4.1 bsIdxRec (BMTR_FS_INDEX_FILE)

3.4.1.1 Tru64 Version

```
typedef struct bsIdxRec
{
    ulong flags;          /* This field must not move ! */
    bsIdxBmtRecT bmt;
    bfAccessT *dir_bfap;
    ulong prune_key_ffree;
    ulong prune_key_fname;
}bsIdxRecT;
```

size 44 bytes

3.4.1.2 Proposed HP-UX Version

```
typedef struct bsIdxRec
{
    uint64T idx_flags;          /* This field must not move ! */
    bsIdxBmtRecT idx_bmt;
    bfAccessT *idx_bfap;
    uint64T idx_pruneKeyFfree;
    uint64T idx_pruneKeyFname;
    bfTagT  idx_parentDirTag;
}bsIdxRecT;
```

size 56 bytes

3.5 msfs/ftx_public.h

3.5.1 ftxIdT

3.5.1.1 Tru64 Version

```
typedef uint32T ftxIdT;
```

size 4 bytes

3.5.1.2 Proposed HP-UX Version

```
typedef uint64T ftxIdT;
```

size 8 bytes

4 New HPUX ODS

4.1 RBMT Shadow record

```
#define RBMT_SHADOWS 2

typedef struct sdw_bsRbmtShadowR {
    uint16T sdw_blksPerPage;
    uint16T sdw_seqno;
    uint16T sdw_actualNumCurPageShadows;
    uint16T sdw_actualNumNextPageShadows;
    bsXtntT sdw_curPageShadows[RBMT_SHADOWS];
    bsXtntT sdw_nextPageShadows[RBMT_SHADOWS];
} bsRbmtShadowRT;
```

size 72 bytes

4.2 Utility run times

```
typedef struct bsRunTimesR {
    int64T  rt_active_fsck;
    int64T  rt_unmounted_fsck;
    int64T  rt_salvage;
    int64T  rt_rsvd1;
    int64T  rt_rsvd2;
    int64T  rt_rsvd3;
    int64T  rt_rsvd4;
    int64T  rt_rsvd5;
} bsRunTimesRT;
```

size 64 bytes

4.3 On Disk Structure Version Number

```
typedef struct onDiskVersion {
    uint16T odv_major;
    uint16T odv_minor;
} onDiskVersionT;
```

size 4 bytes

4.4 File System Cookie

```
typedef  bfDomainIdT bfFsCookieT;
```

size 12 bytes

4.5 Deletion Management Record

Need more info on how this record would be used.

```
typedef struct delMgmt {
   delLinkT dm_delLink;  /* doubly linked global deferred delete list
*/
   delRstT dm_delRst;    /* large bitfile deletion management */
} delMgmtT;
```

size 40 bytes

4.6 Metadata page size

```
#define ADVFS_METADATA_PGSZ             ADVFS_PGSZ
#define ADVFS_METADATA_PGSZ_IN_BLKS     (ADVFS_METADATA_PGSZ /
BS_BLKSIZE)
```

4.7 file system name (BSR_DMN_FSNAME)

```
#define BS_DOMAIN_NAME_SZ (NAME_MAX + 1)

typedef struct bsFsName {
   char bdf_name[BS_DOMAIN_NAME_SZ];
} bsFsNameT;
```

size 256 bytes

5 Open Issues

5.1 Property List (ACLs) (Sections 2.9.14 and 2.9.15)

These sections will need to be updated after the new property list design is completed.

5.2 World Wide Id (Section 2.9.3.4)

Not sure if this field is needed, it might be needed for dual mounts. If later it is determined that it is not needed, we can change the field to reserved or remove completely.