XML Guide for Dynamic Storage Tiering (DST)

by

Marianne Lent
Symantec Advisory Engineer,
Data Center Management Group

February 6, 2007
# Table of Contents

Table of Contents ................................................................. 2  
OVERVIEW ................................................................................ 4  
WHAT IS DYNAMIC STORAGE TIERING?.................................. 4  
WHAT IS A TIER?...................................................................... 4  
A BRIEF XML TUTORIAL ............................................................ 6  
HOW TO TELL DST WHICH FILES TO MANAGE...................... 8  
  Basic File Selection .............................................................. 8  
    EXAMPLE: Files owned by a single user ................................ 9  
    EXAMPLE: Files associated with a single group.................. 9  
    EXAMPLE: Files with a specific name............................... 9  
    EXAMPLE: Files with a particular pattern in their names ...... 10  
    EXAMPLE: Files directly in a particular directory............. 11  
    EXAMPLE: Files anywhere in a directory hierarchy............ 11  
  Compound File Selection ......................................................... 12  
    EXAMPLE: Files owned by either of two users.................. 12  
    EXAMPLE: Files owned by a specific user AND also associated with a specific group ................................................................. 13  
    EXAMPLE: Files owned by a specific user OR associated with a specific group ................................................................. 13  
    EXAMPLE: Files owned by user A in directory B, OR files owned by user C in directory D ................................................................. 14  
An Alternate Method of Selecting Files ...................................... 15  
    EXAMPLE: All files under all directories with a specific name ... 15  
File Selection Rules .................................................................... 16  
    EXAMPLE: Rule for files owned by a single user ................ 16  
    EXAMPLE: Rule for files owned by two users, in different directories .... 16  
    EXAMPLE: Rule for all files in the file system ..................... 17  
HOW TO TELL DST WHICH TIERS TO USE.............................. 18  
  DST Placement Actions .......................................................... 18  
    Describing Tiers to DST ....................................................... 18  
      EXAMPLE: Allocate new files on tier "RAID" .................. 20  
      EXAMPLE: Allocate new files on tiers "RAID1" or "RAID2" or wherever they fit ................................................................. 20  
      EXAMPLE: Move existing files to tier "MAID" ................... 21  
      EXAMPLE: Move existing files to tier "MAID" if they are currently on tier "JBOD" ................................................................. 21  
      EXAMPLE: Delete existing files if they are currently on tier "obsolete" ................................................................. 22  
Balancing Across the Volumes in a Tier ................................. 22  
    EXAMPLE: Distribute new files across tier "array1" .............. 23  
Encoding Tiers in File Selection Rules ....................................... 23  
    EXAMPLE: Rule for files owned by a single user, with CREATE tier ................................. 23  
HOW TO TELL DST WHEN TO TAKE ACTION......................... 24  
  Action Triggers ...................................................................... 24  
    EXAMPLE: Files larger than a given size .......................... 26
EXAMPLE: Files smaller than a given size .......................................................... 27
EXAMPLE: Files equal to a given size ................................................................. 27
EXAMPLE: Files within a size range ................................................................. 27
EXAMPLE: Files last accessed more than 30 days ago................................. 28
EXAMPLE: Files last accessed fewer than 120 days ago........................... 28
EXAMPLE: Files last accessed within a time window ................................. 28
EXAMPLE: Files modified within a time window ......................................... 29
EXAMPLE: Files with read I/O intensity above a threshold .......................... 30
EXAMPLE: Files with write I/O intensity below a threshold .......................... 30
EXAMPLE: Files within a range of I/O intensity .......................................... 31
EXAMPLE: Files within a range of access intensity ...................................... 31

Encoding Action Triggers in File Selection Rules ........................................... 32
EXAMPLE: Relocate when files grow past a certain size .............................. 33
EXAMPLE: Delete when files are not accessed for a long time .................... 33
EXAMPLE: Relocate when files have high I/O activity AND a large size .......... 34
EXAMPLE: Relocate when files have infrequent access OR an old modification
time .................................................................................................................. 35
EXAMPLE: Relocate all files from "array3" to "array7" unconditionally........... 36

DST ADMINISTRATIVE COMMAND: fsppadm ................................................. 37

MISCELLANEOUS NOTES ................................................................................... 37
Interaction Between DIRECTORY, PATTERN, and fsppadm, for File Creation .... 37
Storage Allocation When Appending to a File ................................................. 37
CREATE Does Not RELOCATE ........................................................................ 38
Order of Evaluation for RULEs ...................................................................... 38
Scope of PERIOD ............................................................................................ 38
MIN Flags and MAX Flags ............................................................................. 39

COMPLETE DST PLACEMENT POLICY FILE EXAMPLES .............................. 40
EXAMPLE: Evacuate files from tier "array3" to tier "array7" .............................. 40
EXAMPLE: Place files in "archive" directories on tier "archive" .................... 41
EXAMPLE: Place active files on tier "active" and inactive files on tier "inactive" ... 42
EXAMPLE: Place old files on tier "inactive" and very old files on tier "archive" ... 44
EXAMPLE: Place selected data on special tiers and all other data on generic tier... 46

About the Author ............................................................................................ 49
About the Editor ............................................................................................... 49
OVERVIEW

This document discusses Veritas Storage Foundation 5.0 Dynamic Storage Tiering (DST). It primarily describes how to write XML placement policy rules.

Examples in later sections build on examples in earlier sections, so we recommend that you review the entire document before studying sections of interest.

WHAT IS DYNAMIC STORAGE TIERING?

Dynamic Storage Tiering (DST) is a VxFS feature that allows you to allocate file storage space from different storage tiers according to rules you create.

First you must tag volumes with tier names (as described in “What Is a Tier?”); then you can create placement policy rules that define which files go in which tier and when.

When writing placement policy rules, you must identify the files you want managed, identify the storage tiers you will use, and identify the circumstances in which you want DST to relocate file data.

DST XML provides a language for describing the files, tiers, and circumstances. DST allows you to specify target storage volumes for file creation (where to allocate blocks for new files) and for file relocation (where to allocate replacement blocks when migrating a file from one storage tier to another).

WHAT IS A TIER?

A DST “tier” is a collection of volumes with the same volume tag. The tag must be in the form “vxfs.placement_class.tttt”, where “tttt” is the tier name.

For example, if you want to create three storage tiers on your volumes, “gold”, “silver”, and “bronze”, you must tag the volumes with tags of the form

vxfs.placement_class.gold
vxfs.placement_class.silver
vxfs.placement_class.bronze

To use DST, your VxFS file system must be built on a VxVM volume set. A VxVM volume set can have its component volumes tagged as members of different tiers. All volumes that share the same tag are treated as part of the same tier.

You can classify volumes into whatever tiers make sense to you. For example, you might want to tag high-speed online storage as “gold” and lower-speed nearline storage as “silver”. Or, you might want to tag one group of volumes as “Mail”, and another group as “Documents”.

vxfs.placement_class.gold
vxfs.placement_class.silver
vxfs.placement_class.bronze
For more information on volume tags, refer to the `vxassist(1M)` man page.

**EXAMPLE:**

```
vxassist settag myvol1 vxfs.placement_class.gold
```
A BRIEF XML TUTORIAL

DST rules are stored in XML format. This section gives a very brief summary of XML.

You can view XML as a fancy set of parentheses. Each XML section gets its own open parenthesis and close parenthesis. Each type of XML element gets a tag name—for example “SELECT”. That tag name is embedded in the open parenthesis <SELECT> and close parenthesis </SELECT>. The open parenthesis always embeds the tag name in angle brackets “<>”, and the close parenthesis always adds a backslash “\” in front of the tag name.

The open parenthesis is called the “start tag”, and the close parenthesis is called the “end tag”.

XML elements nest within their start and end tags. For example, if a “RULE” contains several “SELECT” statements, and a “SELECT” statement contains several “USER” statements, the nesting looks like this:

```xml
<RULE Name="my_rule_name" Flags="data">
  <SELECT>
    <USER> user1 </USER>
    <USER> user2 </USER>
    <USER> user3 </USER>
  </SELECT>
  <SELECT>
    <USER> user4 </USER>
    <USER> user5 </USER>
  </SELECT>
</RULE>
```

Sometimes a start tag contains extra information:

```xml
<RULE Name="my_rule_name" Flags="data">
  <DIRECTORY Flags="recursive">
    <SELECT>
      <USER> user1 </USER>
      <USER> user2 </USER>
    </SELECT>
  </DIRECTORY>
</RULE>
```

You also need to add some housekeeping items at the beginning and end:

```xml
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSfspro/config/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0">
  <RULE Name="my_rule_name" Flags="data">
    <SELECT>
      <USER> user1 </USER>
      <USER> user2 </USER>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS> active </CLASS>
        </DESTINATION>
      </ON>
    </CREATE>
  </RULE>
</PLACEMENT_POLICY>
```
Note:

- The `<placement_policy>` start tag gets a matching end tag
- The leading `<?xml` and `!DOCTYPE` tags do not get matching end tags
- These items can be cut and pasted around any DST rule set

XML doesn't care about whitespace; you can add or omit tabs, extra spaces, and newlines.
## HOW TO TELL DST WHICH FILES TO MANAGE

DST allows you to select different file categories for tier placement. You can select files based on user, group, name, or directory location attributes.

### Basic File Selection

The table below shows the basic file selection criteria that tell DST which files to manage.

<table>
<thead>
<tr>
<th>Selection criterion</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User</strong></td>
<td>The name of the user who owns the file (as reported by <code>ls -l</code>). (Note: it must be the name, not the uid.)</td>
<td><code>&lt;USER&gt; &lt;/USER&gt;</code></td>
</tr>
<tr>
<td><strong>Group</strong></td>
<td>The name of the group associated with the file (as reported by <code>ls -l</code>). (Note: it must be the name, not the gid.)</td>
<td><code>&lt;GROUP&gt; &lt;/GROUP&gt;</code></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name of the file (as reported by <code>basename</code>). (Note: this is allowed to contain a single <code>*</code> wildcard character. For example, <code>*.mp3</code> selects all files that end with <code>.mp3</code>. Only the first <code>*</code> is treated as a wildcard; any more <code>*</code>s in the name are treated as literal <code>*</code>s.)</td>
<td><code>&lt;PATTERN&gt; &lt;/PATTERN&gt;</code></td>
</tr>
<tr>
<td><strong>Directory Location</strong></td>
<td>The name of the directory under which the file resides. Use the “recursive” flag to select files anywhere under the given directory; use the “nonrecursive” flag to select just the files directly in that directory. The directory name is relative to the file system's mount point. Note: A given directory must have the same flag anywhere that it is specified within one set of DST rules. (That is, you can mix “recursive” and “nonrecursive” when referring to different directories in the same file system, but you can’t use both on a single directory.) If a selection statement does not specify a <code>DIRECTORY</code>, then the selection applies to the whole file system.</td>
<td><code>&lt;DIRECTORY Flags=&quot;recursive&quot;&gt; &lt;/DIRECTORY&gt;</code></td>
</tr>
</tbody>
</table>

Different file selectors can be combined to create compound selection criteria.
**EXAMPLE:** Files owned by a single user

To tell DST to place files owned by a particular user, embed that user’s name into the **USER** start and end tags:

```xml
<User>
  jsmith
</User>
```

**EXAMPLE:** Files associated with a single group

To tell DST to place files associated with a particular group, embed the group name into the **GROUP** start and end tags:

```xml
<Group>
  accounting
</Group>
```

**EXAMPLE:** Files with a specific name

To tell DST to place all files that have a certain name, embed that name into the **PATTERN** start and end tags:

```xml
<PATTERN>
  mailbox
</PATTERN>
```

This statement designates any file that has the name “mailbox”.
EXAMPLE: Files with a particular pattern in their names

To tell DST to place files that match a certain pattern, embed the name plus a wildcard into the PATTERN start and end tags:

```xml
<PATTERN>
  mail *
</PATTERN>
```

The above statement designates any file whose name starts with the characters “mail”.

```xml
<PATTERN>
  * mail
</PATTERN>
```

The above statement designates any file whose name ends with the characters “mail”.

```xml
<PATTERN>
  GEO*. db
</PATTERN>
```

The above statement designates any file whose name starts with the characters “GEO” and ends with the characters “.db”.

```xml
<PATTERN>
  GEO*. d* b
</PATTERN>
```

The above statement designates any file whose name starts with the characters “GEO” and ends with the characters “.d*b”. (Note that the second “*” is not treated as a wildcard.)
EXAMPLE: Files directly in a particular directory

To tell DST to place files that are directly in a particular directory, embed the directory path name into the `<DIRECTORY>` start and end tags and specify the “nonrecursive” flag:

```
<DIRECTORY Flags="nonrecursive">  
  2006/data/geophysical  
</DIRECTORY>
```

The above statement designates files that are directly in the directory with the path name “2006/data/geophysical”.

Note that this is relative to the mount point of the file system for which you are creating your DST rules. If you mount your file system at mount point “/mnt1”, the above statement applies to anything in the directory “/mnt1/2006/data/geophysical”. If your file system gets unmounted and then mounted again as “/mnt2”, the statement then automatically applies to anything in the directory “/mnt2/2006/data/geophysical”.

EXAMPLE: Files anywhere in a directory hierarchy

To tell DST to place files that are anywhere under a directory hierarchy, embed the directory path name into the `<DIRECTORY>` start and end tags, using the “recursive” flag:

```
<DIRECTORY Flags="recursive">  
  2006/data  
</DIRECTORY>
```

The above statement designates files that are anywhere under the path name “2006/data”. Note that this is relative to the mount point of the file system, as described in the previous example.
Compound File Selection

You can combine basic file selectors into a compound statement, if you want to create a more complex selector. For example, to manage all files owned by user "jsmith" OR user "mjones", you can combine two sets of USER tags. Selector tags are grouped inside an additional set of DST XML tags called “SELECT”.

<table>
<thead>
<tr>
<th>Selection criterion</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping basic selector tags together</td>
<td>Use the SELECT tags to group USER, GROUP, PATTERN, and DIRECTORY selection criteria into compound selection statements. Within one set of SELECT tags, multiples of the same selection type (e.g. two USERS) are ORed together; multiples of different selection types (e.g., a USER plus a GROUP) are ANDed together. Multiple SELECTs within a rule are ORed together. Note: SELECT tags are required around all selection statements, even simple ones with only one selector.</td>
<td>&lt;SELECT&gt; &lt;/SELECT&gt;</td>
</tr>
</tbody>
</table>

EXAMPLE: Files owned by either of two users

To tell DST to place files owned by either of two different users, embed those users’ names into two sets of USER tags:

```xml
<SELECT>
  <USER>
    jsmith
  </USER>
  <USER>
    mjones
  </USER>
</SELECT>
```

The above statement selects all files owned by user "jsmith" and also selects all files owned by user "mjones".
EXAMPLE: Files owned by a specific user AND also associated with a specific group

To tell DST to place files that are owned by a particular user and are also associated with a particular group, use USER tags plus GROUP tags:

```
<SELECT>
  <USER>
    jsmith
  </USER>
  <GROUP>
    accounting
  </GROUP>
</SELECT>
```

The above statement selects all files that are owned by “jsmith” and are also associated with group “accounting”. It ignores files owned by “jsmith” but associated with some other group; it also ignores files associated with group “accounting” but owned by other users. Compare this with the next example.

EXAMPLE: Files owned by a specific user OR associated with a specific group

To tell DST to place files that are owned by a particular user or are associated with a particular group, use USER tags plus GROUP tags in separate SELECT tags:

```
<SELECT>
  <USER>
    jsmith
  </USER>
</SELECT>

<SELECT>
  <GROUP>
    accounting
  </GROUP>
</SELECT>
```

The above statement selects all files owned by “jsmith” and also selects all files associated with group “accounting”. Compare this with the example above—this example shows how extra SELECT tags affect USER tag and GROUP tag interactions.
EXAMPLE: Files owned by user A in directory B, OR files owned by user C in directory D

To tell DST to place files meeting either of two different selection criteria, specify both sets of selection criteria, surround each set with SELECT tags, and group both sets together:

```xml
<SELECT>
  <USER>
    jsmith
  </USER>
  <DIRECTORY Flags="recursive">
    august/sales/northeast
  </DIRECTORY>
</SELECT>

<SELECT>
  <USER>
    mjones
  </USER>
  <DIRECTORY Flags="recursive">
    august/sales/southwest
  </DIRECTORY>
</SELECT>
```

The above statement selects all of jsmith’s files under the directory “august/sales/northeast”, and it also selects all of mjones’s files under the directory “august/sales/southwest”.

Note how the SELECT tags allow us to group each user with the right directory.
An Alternate Method of Selecting Files

DST offers an alternate file selection method that is somewhat different from the previously described method.

<table>
<thead>
<tr>
<th>Selection criterion</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File and Directory Name</strong></td>
<td>Use this alternate selection method to apply storage tier placement to all files under a directory with a specified name, and also to all files with that same name. (The name is allowed to contain a single &quot;*&quot; wildcard character.)</td>
<td><code>&lt;PATTERN Flags=&quot;recursive&quot;&gt;trash&lt;/PATTERN&gt;</code></td>
</tr>
</tbody>
</table>

When used in combination with a recursive DIRECTORY selector, the recursive PATTERN applies to a sub-tree in the file system. Otherwise, it applies to the whole file system.

USER, GROUP, a nonrecursive PATTERN, or a nonrecursive DIRECTORY all override a recursive PATTERN, so avoid using these within the same directory hierarchy as your recursive PATTERN.

**EXAMPLE: All files under all directories with a specific name**

To tell DST to place files that are anywhere under any directory with a particular name, embed that name into PATTERN start and end tags, using the “recursive” flag:

```
<PATTERN Flags="recursive">
  trash
</PATTERN>
```

The above statement designates all files anywhere under any directory named “trash”, and also all files with the name “trash”.

The wildcard pattern matching rules are the same as those for the file name patterns described earlier.

Note that this method of selection should be combined only with a recursive DIRECTORY selector, and not with any other selector type.
File Selection Rules

After writing your DST file selection criteria, you must embed your selection statements into SELECT tags, and then embed the SELECT tags inside RULE tags.

<table>
<thead>
<tr>
<th>Delimiter</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
</table>
| Rule      | This delimiter gives a name to a collection of files, and associates action statements that describe how DST should manage those files. Each rule in the placement policy file must have its own unique name. | `<RULE Name="" Flags="data">
" </RULE>` |

The RULE start tag has a place for a name. Use this name as a mnemonic to describe the collection of files that you've just defined. (The Flags="data" is required, but doesn't have any effect in this release of DST.)

EXAMPLE: Rule for files owned by a single user

```xml
<RULE Name="John_Smith_files" Flags="data">
  <SELECT>
    <USER>
      jsmith
    </USER>
  </SELECT>
</RULE>
```

EXAMPLE: Rule for files owned by two users, in different directories

```xml
<RULE Name="Smith_Jones_August_Sales" Flags="data">
  <SELECT>
    <USER>
      jsmith
    </USER>
    <DIRECTORY Flags="recursive">
      august/sales/northeast
    </DIRECTORY>
  </SELECT>

  <SELECT>
    <USER>
      mjones
    </USER>
    <DIRECTORY Flags="recursive">
      august/sales/southwest
    </DIRECTORY>
  </SELECT>
</RULE>
```
EXAMPLE: Rule for all files in the file system

To tell DST to place all files that are anywhere in the file system, use PATTERN with just a single "*" wildcard:

```xml
<RULE Name="All_Files" Flags="data">
  <SELECT>
    <PATTERN>
      *
    </PATTERN>
  </SELECT>
</RULE>
```

The above statement designates all files that are anywhere in the file system.
HOW TO TELL DST WHICH TIERS TO USE

DST allows you to specify the tiers on which files should be allocated at file creation time, and also allows you to specify the tiers to which your files should be moved during their lifecycle. In addition, DST allows you to specify when to delete a file.

DST Placement Actions

Each of the three DST placement actions (create, delete, relocate) may be defined for each RULE. By defining a set of actions for each RULE, you can specify your desired file placement for each collection of files that you want to manage. (Refer to “File Selections Rules” for examples of RULES.)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation of new file</td>
<td>DST allows you to specify the tier on which new files are allocated when they are created.</td>
<td>&lt;CREATE&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/CREATE&gt;</td>
</tr>
<tr>
<td>Deletion of obsolete file</td>
<td>DST allows you to specify conditions (such as age of file) under which files are deleted.</td>
<td>&lt;DELETE&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/DELETE&gt;</td>
</tr>
<tr>
<td>Relocation of existing file</td>
<td>DST allows you to specify conditions under which files are moved to another tier.</td>
<td>&lt;RELOCATE&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/RELOCATE&gt;</td>
</tr>
</tbody>
</table>

Describing Tiers to DST

When telling DST which tier to use, give DST the same tier name that was used to set the volume tag (refer to “What Is a Tier?” for information on volume tags). Use only the tier name part of the volume tag (you don’t need to give the "vxfs.placement_class." prefix; DST knows to add that).
<table>
<thead>
<tr>
<th>Purpose of tier</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination tier for file creation</td>
<td>This tells DST which tier (or tiers) you want to use when allocating space for new files. You may list multiple DESTINATIONS within one set of ON tags; the order in which they are given indicates the order of preference. Flags=&quot;any&quot; is optional; if given, that means VxFS should fall back to other tiers if the ON tiers are out of space; if not given, that means VxFS should return ENOSPC if the ON tiers are full. BALANCE_SIZE is optional; it distributes allocations across volumes in a tier. Units can be &quot;KB&quot;, &quot;MB&quot;, or &quot;GB&quot;.</td>
<td>&lt;ON Flags=&quot;any&quot;&gt; &lt;DESTINATION&gt; &lt;CLASS&gt; &lt;/CLASS&gt; &lt;BALANCE_SIZE Units=&quot; &quot;&gt; &lt;/BALANCE_SIZE&gt; &lt;/DESTINATION&gt; &lt;/ON&gt;</td>
</tr>
<tr>
<td>Source tier for file deletion</td>
<td>This restricts DST automatic file deletion; the file is deleted only if it has blocks somewhere on the source tier(s). You may list multiple SOURCES within one set of FROM tags.</td>
<td>&lt;FROM&gt; &lt;SOURCE&gt; &lt;CLASS&gt; &lt;/CLASS&gt; &lt;/SOURCE&gt; &lt;/FROM&gt;</td>
</tr>
<tr>
<td>Destination tier for file relocation</td>
<td>This tells DST which tier (or tiers) you want to use when moving a file to a different tier. You may list multiple DESTINATIONS within one set of TO tags; the order in which they are given indicates the order of preference. BALANCE_SIZE is optional; it distributes allocations across volumes in a tier. Units can be &quot;KB&quot;, &quot;MB&quot;, or &quot;GB&quot;.</td>
<td>&lt;TO&gt; &lt;DESTINATION&gt; &lt;CLASS&gt; &lt;/CLASS&gt; &lt;BALANCE_SIZE Units=&quot; &quot;&gt; &lt;/BALANCE_SIZE&gt; &lt;/DESTINATION&gt; &lt;/TO&gt;</td>
</tr>
<tr>
<td>Source tier for file relocation</td>
<td>This restricts DST file relocation; the file is relocated only if it has blocks somewhere on the source tier(s). You may list multiple SOURCES within one set of FROM tags.</td>
<td>&lt;FROM&gt; &lt;SOURCE&gt; &lt;CLASS&gt; &lt;/CLASS&gt; &lt;/SOURCE&gt; &lt;/FROM&gt;</td>
</tr>
</tbody>
</table>

(Notice the "CLASS" tag doesn’t add any extra information. It’s required by the XML but doesn’t convey any meaning.)
EXAMPLE: Allocate new files on tier "RAID"

To tell DST to allocate new files on the tier named "RAID", embed the tier name within a quadruple-nested set of `CREATE+ON+DESTINATION+CLASS` tags:

```
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>
        RAID
      </CLASS>
    </DESTINATION>
  </ON>
</CREATE>
```

The above statement places new files into volumes belonging to the "RAID" tier (that is, volumes which have a tag in the form "vxfs.placement_class.RAID"). If all the volumes tagged as part of "RAID" are full, VxFS returns ENOSPC.

EXAMPLE: Allocate new files on tiers "RAID1" or "RAID2" or wherever they fit

To tell DST to allocate new files on the tiers named "RAID1" or "RAID2", embed the tier names within a quadruple-nested set of `CREATE+ON+DESTINATION+CLASS` tags. Each tier gets its own set of `DESTINATION` and `CLASS` tags. To allow new files to spill over to any available tier, use the "any" flag:

```
<CREATE>
  <ON Flags="any">
    <DESTINATION>
      <CLASS>
        RAID1
      </CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>
        RAID2
      </CLASS>
    </DESTINATION>
  </ON>
</CREATE>
```

The above statement places new files into the "RAID1" tier as long as there is room in "RAID1". When "RAID1" fills up, files go to "RAID2". When both "RAID1" and "RAID2" are full, files go to any tier that has space. Note that the order of preference for tiers is the order they are given in the XML.
EXAMPLE: Move existing files to tier "MAID"

To tell DST to move existing files to the tier named "MAID", embed the tier name within a quadruple-nested set of `RELOCATE+TO+DESTINATION+CLASS` tags:

```xml
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>MAID</CLASS>
    </DESTINATION>
  </TO>
</RELOCATE>
```

The above statement places existing files into volumes that belong to the "MAID" tier (that is, volumes which have a tag in the form "vxfs.placement_class.MAID"). If all the volumes tagged as part of "MAID" fill up, relocation stops and any remaining file blocks stay in their old tier (note that this could leave a file split across two tiers).

EXAMPLE: Move existing files to tier "MAID" if they are currently on tier "JBOD"

To tell DST to move existing files from the tier named "JBOD" to the tier named "MAID", embed the destination tier name ("MAID") within a quadruple-nested set of `RELOCATE+TO+DESTINATION+CLASS` tags, and embed the source tier name ("JBOD") within a quadruple-nested set of `RELOCATE+FROM+SOURCE+CLASS` tags:

```xml
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>MAID</CLASS>
    </DESTINATION>
  </TO>
  <FROM>
    <SOURCE>
      <CLASS>JBOD</CLASS>
    </SOURCE>
  </FROM>
</RELOCATE>
```

The above statement places files currently in the "JBOD" tier into volumes that belong to the "MAID" tier. If all volumes tagged as part of "MAID" fill up, relocation stops and any remaining file blocks stay in "JBOD" (note that this could leave a file partly in "JBOD" and partly in "MAID").
**EXAMPLE:** Delete existing files if they are currently on tier "obsolete"

To tell DST to delete files on the tier named "obsolete", embed the tier name within a quadruple-nested set of `DELETE+FROM+SOURCE+CLASS` tags:

```xml
<DELETE>
  <FROM>
    <SOURCE>
      <CLASS>
        obsolete
      </CLASS>
    </SOURCE>
  </FROM>
</DELETE>
```

The above statement deletes files that are currently in the "obsolete" tier.

**Balancing Across the Volumes in a Tier**

DST allows you to distribute allocations across the volumes in a tier, by using `BALANCE_SIZE` tags nested within `DESTINATION` tags. The `BALANCE_SIZE` tells VxFS to allocate file blocks in uniform chunks across all the volumes in the tier.

(Note that the allocated chunks might not always be exactly the size specified, if free extents of the desired size are not available. VxFS makes a best effort.)

If no `BALANCE_SIZE` is given, VxFS makes no effort to distribute allocations across the volumes in the tier.

To add a new volume to a tier that is using `BALANCE_SIZE`, tag the volume with the appropriate tier name, add the volume to the volume set and the file system, and then run `fsppadm` with the `assign` keyword. DST will then begin balancing new allocations across all the volumes, including the new one.

Note that DST balancing is separate from VxVM volume striping. If the volumes making up a tier are VxVM striped volumes, the volume striping takes place independently of any VxFS balancing across tier volumes.
EXAMPLE: Distribute new files across tier "array1"

To tell DST to allocate new files evenly across the tier named "array1", embed the tier name within a quadruple-nested set of CREATE+ON+DESTINATION+CLASS tags, and add BALANCE_SIZE tags:

```
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>
        array1
      </CLASS>
      <BALANCE_SIZE Units="MB">
        8
      </BALANCE_SIZE>
    </DESTINATION>
  </ON>
</CREATE>
```

The above statement places new files into the "array1" tier; allocations are in 8-megabyte chunks across all tier volumes. (Note that chunk size may vary if required free space is not available.)

Encoding Tiers in File Selection Rules

Once you’ve described the tiers that DST should use, embed those tier statements inside the RULE tags.

EXAMPLE: Rule for files owned by a single user, with CREATE tier

```
<RULE Name="John_Smith_files" Flags="data">
  <SELECT>
    <USER>jsmith</USER>
  </SELECT>
  <CREATE>
    <ON>
      <DESTINATION>
        <CLASS>RAID</CLASS>
      </DESTINATION>
    </ON>
  </CREATE>
</RULE>
```

The above statement places all new files belonging to user "jsmith" on tier "RAID".
HOW TO TELL DST WHEN TO TAKE ACTION

DST allows you to describe when you want it to take relocate or delete actions. (create always happens when the file is first created.) relocate or delete actions can be triggered by file age, file size, or frequency of access.

Action Triggers

DST supports five action triggers.

<table>
<thead>
<tr>
<th>Action Trigger</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
</table>
| File size            | Minimum size, maximum size, exact size, or size range at which a relocation (or deletion) should occur. Units can be: "bytes", "KB", "MB", "GB". MIN Flags can be: "gt", "gteq", or "eq". ("eq" is treated as an exact size match rather than as a minimum.) MAX Flags can be: "lt" or "lteq". The MIN/MAX sizes are given as decimal numbers (e.g. "200" or "2.5"). They are interpreted according to the "Units". | <SIZE Units=" ">
                      |                                                                              | <MIN Flags=" ">
                      |                                                                              | </MIN>
                      |                                                                              | <MAX Flags=" ">
                      |                                                                              | </MAX>
                      |                                                                              | </SIZE>                                                                                   |
| File access time     | Length of time since last file access ("access age") at which a relocation (or deletion) should take place. Calculation is based on POSIX atime. Units can be: "hours" or "days". Use "gt" for MIN Flags and "lt" for MAX Flags. (Refer to "MIN Flags and MAX Flags"). The MIN/MAX length-of-time values are given as decimal numbers. They are interpreted according to the "Units". | <ACCAGE Units=" ">
                      | ("Access Age")                                                                | <MIN Flags="gt">                                                                                   |
                      |                                                                              | </MIN>                                                                                               |
                      |                                                                              | <MAX Flags="lt">                                                                                   |
                      |                                                                              | </MAX>                                                                                               |
                      |                                                                              | </ACCAGE>                                                                                             |
| **File modification time**
| **("Modification Age")** | Length of time since last file modification ("modification age") at which a relocation (or deletion) should take place. Calculation is based on POSIX `mtime`.  
Units can be: "hours" or "days".
Use "gt" for MIN Flags and "lt" for MAX Flags. (Refer to "MIN Flags and MAX Flags".)
The MIN/MAX length-of-time values are given as decimal numbers. They are interpreted according to the "Units". |
| <MODAGE Units=" " >  
| | &lt;MIN Flags="gt">  
| | &lt;MAX Flags="lt">  
| | &lt;/MAX>  
| | &lt;/MODAGE>  
| **File I/O intensity**
| **("I/O Temperature")** | Level of I/O activity ("I/O temperature") at which a relocation (or deletion) should take place.  
Type can be: "nrbytes" (for read), "nwbytes" (for write), or "nrwbytes" (for read/write).
Use "gt" for MIN Flags and "lt" for MAX Flags. (Refer to "MIN Flags and MAX Flags".)
The MIN/MAX temperatures are given as decimal numbers (e.g. "1" or "2.5").  
One "degree" of I/O temperature is equivalent to one file-size-worth of I/O per day. E.g., a 50MB file that gets 150MB of I/O transfer per day has a temperature of 3. Use the Type value to specify whether you want to consider only read I/O, only write I/O, or both. Note that reading the whole 50MB once, or reading the same 1MB over and over 50 times, counts as the same amount of I/O.  
The PERIOD is given as a whole integer (e.g. "3" or "5"). Units must be "days". The PERIOD is the number of days over which the average temperature should be calculated. |
| <IOTEMP Type=" ">  
| | &lt;MIN Flags="gt">  
| | &lt;MAX Flags="lt">  
| | &lt;/PERIOD>  
| | &lt;/IOTEMP>  


**File access intensity**

("Access Temperature")

Level of file access activity ("access temperature") at which a relocation (or deletion) should take place.

Type can be: "nreads" (for read), "nwrites" (for write), or "nrws" (for read/write).

Use "gt" for MIN Flags and "lt" for MAX Flags. (Refer to "MIN Flags and MAX Flags").

The MIN/MAX temperatures are given as decimal numbers (e.g. "1000" or ".5").

One "degree" of access temperature is equivalent to one file access per day. E.g., a file that is accessed 150 times per day has a temperature of 150. Use the Type value to specify whether you want to consider only read access, only write access, or both. Note that file size is not part of the access temperature calculation. An "access" is an I/O request (not a file open).

The PERIOD is given as a whole integer (e.g. "3" or "5"). Units must be "days". The PERIOD is the number of days over which the average temperature should be calculated.

**EXAMPLE: Files larger than a given size**

To tell DST to act when files are larger than a minimum size, enclose the target size in a dual-nested set of SIZE+MIN tags:

```
<SIZE Units="MB">
    <MIN Flags="gt">16</MIN>
</SIZE>
```

The above statement triggers DST action on files greater than 16 megabytes.
**EXAMPLE: Files smaller than a given size**

To tell DST to act when files are smaller than a maximum size, enclose the target size in a dual-nested set of `SIZE+MAX` tags:

```xml
<SIZE Units="MB">
  <MAX Flags="lt">16</MAX>
</SIZE>
```

The above statement triggers DST action on files less than 16 megabytes.

**EXAMPLE: Files equal to a given size**

To tell DST to act when files are an exact size, enclose the target size in a dual-nested set of `SIZE+MIN` tags:

```xml
<SIZE Units="MB">
  <MIN Flags="eq">16</MIN>
</SIZE>
```

The above statement triggers DST action on files that are exactly 16 megabytes.

**EXAMPLE: Files within a size range**

To tell DST to act when files are between two sizes, enclose the target sizes in a dual-nested set of `SIZE+MIN+MAX` tags:

```xml
<SIZE Units="MB">
  <MIN Flags="gteq">16</MIN>
  <MAX Flags="lt">512</MAX>
</SIZE>
```

The above statement triggers DST action on files greater than or equal to 16 megabytes and less than 512 megabytes.
**EXAMPLE: Files last accessed more than 30 days ago**

To tell DST to act on files last accessed more than some number of days (or hours) ago, enclose the minimum access age in a dual-nested set of `ACCAGE+MIN` tags:

```xml
<ACCAGE Units="days">
  <MIN Flags="gt">
    30
  </MIN>
</ACCAGE>
```

The above statement triggers DST action on files last accessed more than 30 days ago. (Files accessed recently, within the past 30 days, do not count.)

**EXAMPLE: Files last accessed fewer than 120 days ago**

To tell DST to act on files last accessed fewer than some number of days (or hours) ago, enclose the maximum access age in a dual-nested set of `ACCAGE+MAX` tags:

```xml
<ACCAGE Units="days">
  <MAX Flags="lt">
    120
  </MAX>
</ACCAGE>
```

The above statement triggers DST action on files last accessed fewer than 120 days ago. (Files that are inactive, and have not been accessed in over 120 days, do not count.)

**EXAMPLE: Files last accessed within a time window**

To tell DST to act on files last accessed within a time window, enclose the lower and upper bound in a dual-nested set of `ACCAGE+MIN+MAX` tags:

```xml
<ACCAGE Units="hours">
  <MIN Flags="gt">
    30
  </MIN>
  <MAX Flags="lt">
    120
  </MAX>
</ACCAGE>
```

The above statement triggers DST action on files last accessed more than 30 hours ago but less than 120 hours ago.
EXAMPLE: Files modified within a time window

To tell DST to act on files last modified within a time window, enclose the lower and upper bound in a dual-nested set of MODAGE+MIN+MAX tags:

```xml
<MODAGE Units="days">
  <MIN Flags="gt">10</MIN>
  <MAX Flags="lt">20</MAX>
</MODAGE>
```

The above statement triggers DST action on files last modified more than 10 days ago but less than 20 days ago.
EXAMPLE: Files with read I/O intensity above a threshold

To tell DST to act on files subject to I/O activity above a certain level, enclose the target minimum I/O temperature in a dual-nested set of IOTEMP+MIN tags, plus a PERIOD tag:

```
<IOTEMP Type="nrbytes">
  <MIN Flags="gt">
    4
  </MIN>
  <PERIOD Units="days">
    30
  </PERIOD>
</IOTEMP>
```

The above statement triggers DST action on files having read I/O activity equivalent to more than 4 times the size of the file each day (averaged over a period of 30 days). (E.g., a 50MB file that averages more than 200MB of read activity per day, or a 12MB file that averages more than 48MB of read activity per day.)

EXAMPLE: Files with write I/O intensity below a threshold

To tell DST to act on files subject to I/O activity below a certain level, enclose the target maximum I/O temperature in a dual-nested set of IOTEMP+MAX tags, plus a PERIOD tag:

```
<IOTEMP Type="nwbytes">
  <MAX Flags="lt">
    .5
  </MAX>
  <PERIOD Units="days">
    30
  </PERIOD>
</IOTEMP>
```

The above statement triggers DST action on files with write I/O activity equivalent to at most .5 (one-half) times the size of the file each day (averaged over a period of 30 days). (E.g., a 50MB file that averages at most 25MB of write activity per day, or a 12MB file that averages at most 6MB of write activity per day.)
EXAMPLE: Files within a range of I/O intensity

To tell DST to act on files falling within an I/O intensity range, enclose the lower and upper bound in a dual-nested set of IOTEMP+MIN+MAX tags, plus a PERIOD tag:

```xml
<IOTEMP Type="nrwbytes">
  <MIN Flags="gt">5</MIN>
  <MAX Flags="lt">10</MAX>
  <PERIOD Units="days">20</PERIOD>
</IOTEMP>
```

The above statement triggers DST action on files having total (read and write) I/O activity equivalent to at least 5 and at most 10 times the size of the file each day (averaged over a period of 20 days).

EXAMPLE: Files within a range of access intensity

To tell DST to act on files fall within an access intensity range, enclose the lower and upper bound in a dual-nested set of ACCESSTEMP+MIN+MAX tags, plus a PERIOD tag:

```xml
<ACCESSTEMP Type="nrws">
  <MIN Flags="gt">5</MIN>
  <MAX Flags="lt">10</MAX>
  <PERIOD Units="days">20</PERIOD>
</ACCESSTEMP>
```

The above statement triggers DST action on files having at least 5 and at most 10 file accesses each day (averaged over a period of 20 days). Both read and write accesses are counted toward the total. (To count only read accesses, use "nreads"; to count only write accesses, use "nwrites".)


**Encoding Action Triggers in File Selection Rules**

Embed all `SIZE`, `ACCAGE`, `MODAGE`, `IOTEMP`, and `ACCESSTEMP` tags within `WHEN` tags, and embed `WHEN` tags within `RELOCATE` or `DELETE` tags.

<table>
<thead>
<tr>
<th>Delimiter</th>
<th>Description</th>
<th>XML tags</th>
</tr>
</thead>
</table>
| **When**  | Encloses the `SIZE`, `ACCAGE`, `MODAGE`, `IOTEMP`, and/or `ACCESSTEMP` specifiers. Each `RELOCATE` or `DELETE` may have one `WHEN`; each `WHEN` may have zero or one each of `SIZE`, `ACCAGE`, `MODAGE`, `IOTEMP`, and `ACCESSTEMP`. The items inside one set of `WHEN` tags are ANDed together. To OR together multiple `RELOCATE` or `DELETE` conditions, use multiple sets of `RELOCATE` or `DELETE` statements. To take a `RELOCATE` or `DELETE` action unconditionally, omit any `WHEN` clause from that action. | `<WHEN>       
WHEN         
WHEN         
</WHEN>`       |


**EXAMPLE: Relocate when files grow past a certain size**

To tell DST to relocate files larger than a specific size, enclose `SIZE` tags within `WHEN` tags and add them to your `RELOCATE` tags:

```xml
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>
        MAID
      </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="GB">16</SIZE>
  </WHEN>
</RELOCATE>
```

The above statement relocates files larger than 16 gigabytes to tier "MAID".

**EXAMPLE: Delete when files are not accessed for a long time**

Use this example very carefully, because it actually deletes files.

To tell DST to delete files that have not been accessed for a specified period, enclose `ACCAGE` tags within `WHEN` tags and add them to your `DELETE` tags:

```xml
<DELETE>
  <WHEN>
    <ACCAGE Units="days">730</ACCAGE>
  </WHEN>
</DELETE>
```

The above statement deletes files that have not been accessed for two or more years.

Again, `DELETE` actions will delete selected files. Be very careful when using this.
EXAMPLE: Relocate when files have high I/O activity AND a large size

To tell DST to relocate files having both high I/O activity and a minimum size, enclose SIZE tags and IOTEMP tags within WHEN tags and add them to your RELOCATE tags:

```xml
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>
        array1
      </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="GB">
      <MIN Flags="gt">16</MIN>
    </SIZE>
    <IOTEMP Type="nrwbytes">
      <MIN Flags="gt">3</MIN>
      <PERIOD Units="days">15</PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>
```

The above statement relocates files to tier "array1" if they are larger than 16 gigabytes and have total average daily I/O activity equivalent to more than 3 times the file size (averaged over 15 days).

Placing both of the WHEN items in the same RELOCATE statement tells DST to act when both the first condition AND the second condition are true. (Compare this to the next example.)
**EXAMPLE: Relocate when files have infrequent access OR an old modification time**

To tell DST to relocate files that have infrequent access or have not been modified in a long time, enclose ACCESSTEMP tags and MODAGE tags within WHEN tags and add them to separate RELOCATE tags:

```xml
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>
        MAID
      </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <MODAGE Units="days">
      <MIN Flags="gt">60</MIN>
    </MODAGE>
  </WHEN>
</RELOCATE>

<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>
        MAID
      </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <ACCESSTEMP Type="nrws">
      <MAX Flags="lt">.5</MAX>
      <PERIOD Units="days">30</PERIOD>
    </ACCESSTEMP>
  </WHEN>
</RELOCATE>
```

The above statement relocates files to tier "MAID" if they have not been modified in over 60 days or if they average less than one access every two days (averaged over 30 days).

Using two separate `RELOCATE` statements tells DST to act when either the first condition OR the second condition is true. (Compare this to the previous example.)
**EXAMPLE: Relocate all files from "array3" to "array7" unconditionally**

To tell DST to relocate files from tier "array3" to tier "array7" unconditionally, omit any **WHEN** from your **RELOCATE** tags:

```xml
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>array3</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>array7</CLASS>
    </DESTINATION>
  </TO>
</RELOCATE>
```

The above statement relocates files from the "array3" tier to the "array7" tier, regardless of size, age, I/O, or access. This is potentially useful during hardware upgrades.
DST ADMINISTRATIVE COMMAND: fsppadm

Once the XML placement policy file is finished, use the fsppadm command to apply the file to your file system. Refer to the fsppadm(1M) man page for details.

To initialize the tiers and install the XML placement policy, run fsppadm with the assign keyword. This needs to be done when:

- making changes to the XML
- adding new volumes to tiers
- setting up a new DIRECTORY selection (refer to "Interaction Between DIRECTORY, PATTERN, and fsppadm, for File Creation").

Once fsppadm assign has run, file creation is according to tiers specified by the CREATE actions.

To perform RELOCATE and DELETE actions according to the XML rules, run fsppadm with the enforce keyword. This should be done regularly (e.g., once every night).

MISCELLANEOUS NOTES

Interaction Between DIRECTORY, PATTERN, and fsppadm, for File Creation

A directory specified with the DIRECTORY selector must be created before running fsppadm assign. That is, after creating the directory with mkdir, you must run fsppadm assign at least once for the CREATE tiers to take effect. (Note that fsppadm enforce does not initialize CREATE tiers; only fsppadm assign does this.)

In contrast, a directory specified with PATTERN Flags="recursive" can be created after running fsppadm assign. Whenever a directory with the specified name gets created, the associated CREATE action automatically takes effect, without running fsppadm assign again. (However, if PATTERN Flags="recursive" is combined with DIRECTORY, then you must run fsppadm assign at least once after creating the directory specified with the DIRECTORY selector.)

Storage Allocation When Appending to a File

When an existing file is extended by appending to the end (or by filling in a hole), the new space is allocated from the same tier used to place the file in its current location. (If multiple tiers were specified in the original placement rule, they are all available for append as well.)
If a new file is placed by way of the CREATE action, later file appends follow the setting of the "any" flag. That is, if "any" is set, appends are placed on any tier if the designated tiers are full, and if "any" is not set, appends fail with ENOSPC if the designated tiers are full. However, once the file moves via a RELOCATE action, the "any" flag is lost. All appends after a RELOCATE succeed only if there is space on the designated tier (or tiers); if there is no space on those tiers, the append fails with ENOSPC even if space exists on another tier.

**CREATE Does Not RELOCATE**

Specifying a CREATE tier in a RULE affects only files newly created after the rule is put in place. CREATE does not automatically move existing files. To move existing files, the rule must specify a RELOCATE action.

**Order of Evaluation for RULEs**

RULES are evaluated in the order given in the XML. If a file can match more than one rule, it is governed by the first rule that it matches.

If a file doesn't match any rule, its data blocks are allocated from any tier that has space. To ensure that all files match at least one rule, include a final rule that matches everything (refer to "EXAMPLE: Rule for all files in the file system", "EXAMPLE: Place files in 'archive' directories on tier 'archive'", and "EXAMPLE: Place selected data on special tiers and all other data on generic tier").

Similarly, RELOCATE and DELETE actions within a RULE are evaluated in the order they are given, and DST takes the first action triggered by its WHEN tag. Therefore, actions must be ordered in the reverse of when they would trigger. For example, an action for files not accessed in the past 120 days should come before an action for files not accessed in the past 30 days (because if DST sees the 30-day rule first, it does not notice the other 120-day rule). Similarly, DELETE actions should always precede RELOCATE actions (because the RELOCATE actions trigger before the DELETE actions do, and the ordering needs to be in reverse).

**Scope of PERIOD**

The scope of the PERIOD for ITEMP and ACCESSTEMP is global: the biggest PERIOD in the placement policy file is used for all ITEMP and ACCESSTEMP calculations within a given file system.
MIN Flags and MAX Flags

The MIN Flags and MAX Flags for SIZE, ACCAGE, MODAGE, IOTEMP, and ACCESSTEMP may all be set to "gt"|"gteq"|"eq" or "lt"|"lteq". However, "gteq", "eq", and "lteq" are meaningful only for SIZE. For the action triggers other than SIZE, "gteq", and "lteq" are not perceptibly different from "gt" and "lt", and "eq" could prevent any action from occurring.

For ACCAGE and MODAGE, the time is measured in timestamp units, so there's only one second of difference between "greater than 30 hours" vs. "greater than or equal to 30 hours". For IOTEMP and ACCESSTEMP, the temperature is calculated as an average over a period of days, and the calculation produces fractional precision that similarly does not result in a meaningful difference between "lt" vs. "lteq" or "gt" vs. "gteq".

For ACCAGE and MODAGE, "eq" is true only for the one second when the age is exactly equal. For IOTEMP and ACCESSTEMP, "eq" is true only when the average divides out to exactly the given temperature, to the smallest precision supported by the hardware. Neither target is likely to be hit in a real production environment, so "eq" will most likely do nothing.

If you need to distinguish an extra second or .0000000001 of precision, the XML will accept "lteq" or "gteq", but in most cases this difference is not visible.

For SIZE, a one-byte difference is visible and meaningful, so "gt"|"gteq"|"eq" and "lt"|"lteq" are all potentially useful with SIZE.
COMPLETE DST PLACEMENT POLICY FILE
EXAMPLES

This section gives a few complete examples of DST placement policy files.

EXAMPLE: Evacuate files from tier "array3" to tier "array7"

```xml
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSfspro/config/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0">
  <RULE Name="All_Files" Flags="data">
    <SELECT>
      <PATTERN>*</PATTERN>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS>array7</CLASS>
        </DESTINATION>
      </ON>
    </CREATE>
    <RELOCATE>
      <TO>
        <DESTINATION>
          <CLASS>array7</CLASS>
        </DESTINATION>
      </TO>
      <FROM>
        <SOURCE>
          <CLASS>array3</CLASS>
        </SOURCE>
      </FROM>
    </RELOCATE>
  </RULE>
</PLACEMENT_POLICY>
```

This policy unconditionally moves files from the "array3" tier to the "array7" tier. This is useful following hardware upgrades. In this example, new hardware was added to the volume set and tagged as tier "array7". After DST relocates all files from "array3" to "array7", the old hardware underlying "array3" can be removed from the volume set and then physically detached. The file system remains mounted and accessible throughout. (The CREATE action ensures that new files go directly to tier "array7").
XML Guide for Dynamic Storage Tiering (DST)

EXAMPLE: Place files in "archive" directories on tier "archive"

```xml
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM
"/opt/VRTSfspro/config/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0">
  <RULE Name="Archive_Files" Flags="data">
    <SELECT>
      <PATTERN Flags="recursive">archive</PATTERN>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS>archive</CLASS>
        </DESTINATION>
      </ON>
    </CREATE>
    <RELOCATE>
      <TO>
        <DESTINATION>
          <CLASS>archive</CLASS>
        </DESTINATION>
      </TO>
    </RELOCATE>
  </RULE>
</PLACEMENT_POLICY>
```

This policy places files named "archive" in the "archive" tier, and also places any files under "archive" directories in the "archive" tier.

Note: In this example, files not matching the "archive" pattern might also be randomly placed in the "archive" tier, because there is no rule explicitly defining placement for other files. To keep other files out of "archive", add a rule at the end that matches all other files. Refer to "EXAMPLE: Place selected data on special tiers and all other data on generic tier" for an illustration.
EXAMPLE: Place active files on tier "active" and inactive files on tier "inactive"

```xml
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM
"/opt/VRTSfspro/config/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0">
  <RULE Name="All_Files" Flags="data">
    <SELECT>
      <PATTERN>*</PATTERN>
    </SELECT>
    <CREATE>
      <ON><DESTINATION><CLASS>active</CLASS></DESTINATION></ON>
    </CREATE>
    <RELOCATE>
      <TO><DESTINATION><CLASS>inactive</CLASS></DESTINATION></TO>
      <WHEN><ACCAGE Units="days"><MIN Flags="gt">120</MIN></ACCAGE></WHEN>
    </RELOCATE>
    <RELOCATE>
      <TO><DESTINATION><CLASS>active</CLASS></DESTINATION><FROM><SOURCE><CLASS>inactive</CLASS></SOURCE></FROM>
      <WHEN><ACCAGE Units="days"><MIN Flags="gt">120</MIN></ACCAGE></WHEN>
    </RELOCATE>
  </RULE>
</PLACEMENT_POLICY>
```
<MAX Flags="lt">
  30
</MAX>
</ACCAGE>
</WHEN>
</RELOCATE>
</RULE>
</PLACEMENT_POLICY>

This policy pushes files to tier "inactive" if they have not been accessed in over 120 days. It pulls newly-reactivated files from tier "inactive" back to tier "active" if they have been accessed within the past 30 days.
EXAMPLE: Place old files on tier "inactive" and very old files on tier "archive"

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM
"/optVRTSfspro/config/placement_policy.dtd">  
<PLACEMENT_POLICY Version="5.0">  
  <RULE Name="All_Files" Flags="data">  
    <SELECT>  
      <PATTERN>  
        *  
      </PATTERN>  
    </SELECT>  
    <RELOCATE>  
      <DESTINATION>  
        <CLASS>archive</CLASS>  
      </DESTINATION>  
      <WHEN>  
        <ACCAGE Units="days">  
          <MIN Flags="gt">120</MIN>  
        </ACCAGE>  
      </WHEN>  
    </RELOCATE>  
    <RELOCATE>  
      <DESTINATION>  
        <CLASS>inactive</CLASS>  
      </DESTINATION>  
      <WHEN>  
        <ACCAGE Units="days">  
          <MIN Flags="gt">30</MIN>  
        </ACCAGE>  
      </WHEN>  
    </RELOCATE>  
  </RULE>  
</PLACEMENT_POLICY>
```

This policy pushes files to tier "inactive" if they have not been accessed in over 30 days. It pushes files to tier "archive" if they have not been accessed in over 120 days. Note how the 120-day action must come first. DST performs the first action matching a `WHEN` trigger; if the 30-day action is first, files beyond 120 days
will match the 30-day action (since anything beyond 120 days is also beyond 30 days).
EXAMPLE: Place selected data on special tiers and all other data on generic tier

<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM
"/opt/VRTSfspro/config/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0">
  <RULE Name="John_Smith_data" Flags="data">
    <SELECT>
      <USER>
        jsmith
      </USER>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS>
            jsdata
            </CLASS>
        </DESTINATION>
      </ON>
    </CREATE>
    <RELOCATE>
      <TO>
        <DESTINATION>
          <CLASS>
            jsdata
            </CLASS>
        </DESTINATION>
      </TO>
    </RELOCATE>
  </RULE>
  <RULE Name="Mindy_Jones_data" Flags="data">
    <SELECT>
      <USER>
        mjones
      </USER>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS>
            mjdata
            </CLASS>
        </DESTINATION>
      </ON>
    </CREATE>
    <RELOCATE>
      <TO>
        <DESTINATION>
          <CLASS>
            mjdata
            </CLASS>
        </DESTINATION>
      </TO>
    </RELOCATE>
  </RULE>
</PLACEMENT_POLICY>
This policy places:

- files owned by "jsmith" on tier "jsdata"
- files owned by "mjones" on tier "mjdata"
- all other files on tier "misc"

Note that the "all_other_data" rule must go last in the file. DST uses the first rule matching each file; if the "all_other_data" rule were first, all files would match it and the following rules would be ignored.
About the Author

Marianne Lent is a Symantec Advisory Engineer.

She has watched computer technology progress from vacuum tubes to transistors to integrated circuits, while working with keypunches, card sorters, mainframes, network stacks, microkernels, and open systems. Marianne spent 6 years in the Veritas File System Group, where she contributed to the specification and implementation of the XDSM (DMAP) standard and was a leader of the Veritas Cluster File System development team.

Marianne is currently a member of the Symantec Data Center Management Group’s product architecture team.

Marianne has a degree in journalism from the Department of Communication at Stanford University.

About the Editor

W. David Schwaderer is the Symantec Veritas Architect Network (VAN) Editor-In-Chief.

He has authored six commercial software programs and ten technical books, including the Symantec Yellow Books™ volume titled Secure Business Continuity: Strategies for Business Continuity Management and Disaster Recovery, A comprehensive approach to enhancing business continuity planning and testing.

His latest book is titled Data Lifecycles—Managing Data for Strategic Advantage published by John Wiley & Sons Ltd.


David has a Masters Degree in Applied Mathematics from the California Institute of Technology and an MBA from the University of Southern California. He lectures at Stanford on the subject of innovation.