IBM
Spectrum
Scale

# IBM® Spectrum Scale™ Information Life cycle Management Policies

An introduction to IBM Spectrum Scale ILM policies for file archiving and statistics with IBM Spectrum Archive Enterprise Edition

Version 10.2
(08/17/2020)

Nils Haustein | Senior Technical Staff Member | IBM EMEA Storage Competence Center

## Table of Contents

Thanks to my colleague and friend Achim Christ (CTO IBM Systems Lab Services, Germany) for his thorough review and the great ideas which made this document more exciting. Thanks to Yaron Daniel (Storage Architect, IBM Systems Lab Services, Israel) for the feedback and additions.

## Introduction

This document presents some of my experience regarding IBM® Spectrum Scale™ Information Lifecycle Management (ILM) policies applicable to internal and external storage pools. External pools can be provided with extra software from IBM such as IBM Spectrum Archive Enterprise Edition and IBM Spectrum Protect for Space Management. IBM Spectrum Scale is a clustered and scalable file system software from IBM. IBM Spectrum Scale includes a policy engine allowing to place, migrate and list files based on their attributes and characteristics.

This document guides you through the concepts of IBM Spectrum Scale ILM policies and focuses on the integration of IBM Spectrum Scale with IBM Spectrum Archive Enterprise Edition. The integration of IBM Spectrum Scale with IBM Spectrum Protect for Space Management is further described in [7]. Readers of this document should have basic administrative knowledge regarding IBM Spectrum Scale concepts, architectures and policies.

Before the concepts and syntax of placement, migration and list rules and policies are explained in section ILM Policies and Rules I will give you a short introduction to IBM Spectrum Scale ILM and related file system and storage pool concepts (see section The Basics). At the end of this document in section Examples and use cases you will find more examples and use cases for ILM policies.

Note: The examples shown in this document may not work in your environment. The author and IBM do not assume any liability regarding the examples given. It is not my intention to repeat the relevant section of the IBM Spectrum Scale knowledge center [1], [2], I will only highlight the most important concepts and terms. The reader should be familiar with the IBM Spectrum Scale knowledge center and other resources.

## The Basics

IBM Spectrum Scale includes a policy engine capable to identify files based on their attributes and automatically manage the identified files. Management of files includes placement, migration, listing, deletion, encryption, etc. The IBM Spectrum Scale policy engine operates cluster wide.

The IBM Spectrum Scale *policy engine* can identify files based on selection criteria - provided with policy rules - without browsing through the directories. You can envision a database where all file attributes (path and file names, size, dates, etc.) are stored and the policy engine runs queries against this database. This is just an analogy, it might not be implemented like this. Furthermore, the IBM Spectrum Scale policy engine manages the identified files, based on the rule provided action codes.

The policy engine can be programmed by the means of policy rules. A *rule* is an SQL-like statement that instructs the IBM Spectrum Scale policy engine what to do with a file in a specific storage pool if the file meets specific criteria. A rule can apply to any file being created or only to files being created within a specific directory.

A *policy* is a set of rules that describes the data lifecycle for a set of files based on the files' attributes. Policies can be used for different purposes such as for managing the lifecycle of files (ILM policies), listing files based on certain characteristics (list policies), deleting files (deletion policy), encrypting files (encryption policy) and so on. In this document we focus on policies and rules for placement, migration and listing.

In the following sub-section the basic IBM Spectrum Scale file system and storage pool concepts are explained (section IBM Spectrum Scale file systems and pools) which are the basis for ILM implemented in IBM Spectrum Scale. The section IBM Spectrum Scale Policy engine explains ILM policies in general and how the IBM Spectrum Scale policy engine works.

## IBM Spectrum Scale file systems and pools

Figure 1 shows the basic concept of an IBM Spectrum Scale file system and pools in the context of ILM.

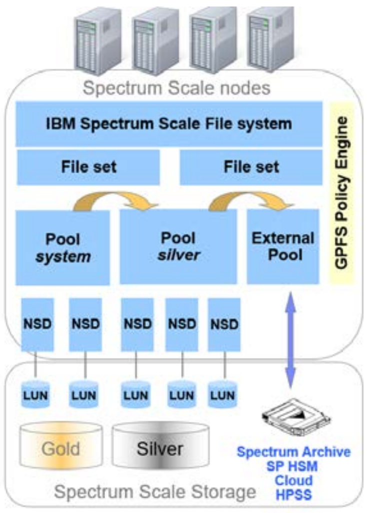

Figure 1: Basic IBM Spectrum Scale file system architecture

The IBM Spectrum Scale file system represents a global name space over all IBM Spectrum Scale nodes that are members of the IBM Spectrum Scale cluster. The IBM Spectrum Scale file system is comprised of storage pools. The example in figure 1 shows three storage pools named: ‘system’, ‘silver’ and external. Any IBM Spectrum Scale file system has at least one pool named ‘system’. The file system is comprised of data - such as files - and metadata in the storage pools. File system metadata comprises file characteristics and IBM Spectrum Scale internal metadata and can only be stored in the pool named ‘system’.

There are two kinds of IBM Spectrum Scale storage pools: internal and external pools. An *internal pool* is a collection of disks (hard disk or SSD) with similar properties that are managed together. There must be a minimum of one internal pool (the ‘system’ pool) and a maximum of eight internal pools per file system. Placement and migration can be done on internal pools.

An *external pool* is an external storage device that is represented and managed by an interface script (see section External pools). An external pool must not have disks, it can also be based on tape or other storage devices. Storing and retrieving data (files) in an external pool is managed by the interface script. Typical external pools are IBM Spectrum Protect for Space Management (SP HSM), IBM Spectrum Archive Enterprise Edition (LTFS EE) or HPSS. The interface script for the SP HSM is the HSM client, for IBM Spectrum Archive EE it is the IBM Spectrum Archive client and for HPSS it is the HPSS client (GHI).

The term disk in the context of IBM Spectrum Scale internal pools refers to so called *Network Shared Disk (NSD)*. An NSD maps 1:1 to a storage LUN or volume provided by a storage system and is dedicated to a pool. All NSD in one internal pool are comprised of storage LUNs with similar characteristics. Referring to figure 1, the NSDs of the ‘system’ pool are configured on LUNs from SSD drives and the NSDs of the ‘silver’ pool are configured on LUNs from Near-Line SAS drives. Therefore, each pool has its own I/O characteristic.

It is not according to best practices to have LUNs of different characteristics (such as SSD and NL-SAS) in one internal pool. External pools have no NSD.

NSD can be configured with different usage types, the most important are:

- *dataOnly*: only data (such as file data) is stored
- *metadataOnly*: only file system metadata is stored
- *dataAndMetadata*: data and metadata are stored
- *descOnly*: no data or metadata is stored, this is a descriptor disk only
- *localCache*: local read-only cache device

Depending on the usage type of the NSD the pool can store data, metadata, both or none. Metadata can only be stored in pool named ‘system’, i.e. the usage type of NSD in the ‘system’ pool must be metadataOnly or dataAndMetadata. The usage type of other pools (not named ‘system’) is typically dataOnly. Using NSD with usage type descOnly is only for file system descriptor providing quorum capability. An NSD with usage type localCache provides local read-only caching for the file system.

Also shown in figure 1 are IBM Spectrum Scale filesets. A fileset is a logical sub-tree of the file system that is managed as a logical partition within the file system. Snapshots and quota can be configured on a fileset level. In the context of ILM filesets can be used to control placement and migration. A fileset is identified by the fileset name and the associated path name within the file system. Placement and migration of files in a fileset can be controlled based on the fileset name or path names in conjunction with other file attributes such as file size, file type, access time, etc.

## IBM Spectrum Scale policy engine

The IBM Spectrum Scale policy engine can identify files based on programmable criteria and conditions and manage their lifecycle. Management of files includes:

- Placement of files when they are created
- Migration of files during the lifecycle
- Listing files
- Encrypting files
- Deleting files

In this document we focus on policies for placement, migration and listing.

The programming is done with ILM policies. *ILM policies* include rules for placement, migration and listing files.

A *migration rule* describes the migration of all or a subset of files from one file system pool to another file 'system' pool. The selection of files for migration is based on file attributes. Both file system pools must belong to the same file system and allow for data usage. The file system pool can be an internal pool or an external pool (see section Migration). A migration rule is run by the policy engine (`mmapplypolicy` command or callback).

A *placement rule* describes in which storage pool files are stored upon creation. Matching files to pool during creation is based on file attributes. The file system pool must exist and belong to the file system. Placement is only possible on internal storage pools, not on external pools. Placement rules are configured as active policies for a file system using the `mmchpolicy` command. Placement policies cannot be executed using the `mmapplypolicy` command (see section Placement).

Depending on the way an ILM policy is run we can distinguish two types of ILM policies:

1. *Automated policy*: I call it automated because this policy is run automatically. One may also call it active policy because it is always active for the file system. There can be a maximum of one automated policy for a file system that must include all relevant rules, such as placement rules and migration rules. Placement rules of an automated policy are evaluated when a file is created in the file system. Migration rules in an automated policy are typically threshold-based and automatically invoked when a file system pool reaches the rule-defined occupation threshold (see section Automated migration). When using migration rules in an automated policy a migration callback must be configured (see section Migration Callback).

2. *Scheduled policy*: I call it schedule policy because this policy is run manually or by means of scheduling. One may call it manual policy because unlike automated policies it must be started manually or by schedulers. A scheduled policy does not typically include placement rules. Migration rules in a scheduled policy can be, but do not have to be threshold-based (see section Scheduled migration). Migration rules in a scheduled policy do not require a callback to be configured.

A *list policy* includes rules to identify files based on file attributes such as name, size, state, etc. The output of a list policy - the list of files that have been identified based on the list rules - can be used as input for other processing (see section Lists).

Many policies - except for placement and encryption policies - are run by the `mmapplypolicy` command (see section Running a policy) which invokes the *policy engine*. This command requires at least a file system name and a policy filename as input to apply the policy in the policy file to the file system. For automated ILM policies the policy engine is started by the callback which is invoked by the policy applied with the `mmchpolicy` command. For scheduled policies the policy engine is started by a schedule or administrator using the mmapplypolicy command. The policy engine runs in the following phases:

1. Directory scan: reads file metadata from file inodes
2. Rule evaluation: evaluates the rules provided in the policy file (top down) by matching file metadata with rules and selects the identified files
3. Execution of file operations: for migration rules performs the migration according to the migration rule. For list rules it lists the files. For deletion rules it deletes the file

The work in all phases can be distributed on all or a subset of IBM Spectrum Scale nodes, making the policy engine ultrafast and efficient.

## ILM Policies and Rules

In this section placement, migration, recall and list rules and policies are described. I will focus on the basic syntax and semantics of rules and policies, detailed examples are given in section Examples and use cases. This section starts with important file attributes that can be evaluated by the policy engine.

Since a policy is comprised of one or more rules the transition from a rule to a policy is transparent. So please forgive me if I use the word policy for a rule and vice versa.

Sample code and policies are stored in a GitHub repository [5].

### File attributes

The policy engine can inquire and match many attributes of files against values. The matching of file attribute is done by matching the attribute name against a value. This is typically done in a WHERE clause of a rule. The following sample matches the name of the file against the pattern of *.mp3:

```
WHERE lower(NAME) like ‘%.mp3%’
```

The function `lower(NAME)` converts the file name into all lowercase letters. The character ‘%’ represents a wildcard.

A complete list of file attributes can be found in the IBM Spectrum Scale Knowledge Center [6]. The table below gives an overview about the most commonly used file attributes:

| File attribute | Description |
|---|---|
| NAME | Name of the file (excluding path name) |
| PATH_NAME | Path and file name |
| FILE_SIZE | Size of the file in Bytes. Note, after migrating a file to an external pool, the file size remains the same. |
| KB_ALLOCATED | Number of kilobytes of disk space allocated for the file data. If the file is migrated to an external pool then KB_ALLOCATED is 0 (if the stub-size is 0) |
| ACCESS_TIME | Date and time that the file was last accessed (POSIX atime) |
| MODIFICATION_TIME | Date and time that the file was last modified (POSIX mtime). This time changes whenever the file data is modified. |
| CHANGE_TIME | Date and time that the file was last changed (POSIX ctime). This time changes whenever the file data or attributes of the file are changed. |
| EXPIRATION_TIME | Date and time that the file will expire. Only available with immutable filesets |
| FILE_HEAT | Access temperature of a file based on the frequency of file access |
| POOL_NAME | Name of the pool where the file is currently stored. Note, for files migrated to an external pool the pool name will be the name of the source pool from which the file has been migrated. |
| USER_ID | User ID of the owning user |
| GROUP_ID | Group ID of the owning user |
| MISC_ATTRIBUTES | A collection of file attributes represented as string value. For example, file migration states (resident, pre-migrated and migrated) are encoded in the MISC_ATTRIBUTES sting. Further examples are encryption, compression and replication setting. For a complete list of attributes see [6] |
| XATTR | A collection of system defined file attributes. For example, the tape ID for pre-migrated and migrated files is encoded in an attribute named `dmapi.IBMTPS`. Further system defined attributes can be listed with the command: `mmlsattr -L -d` *`filename`* |

One word about FILE_SIZE and KB_ALLOCATED: FILE_SIZE expresses the size of a file in bytes, KB_ALLOCATED expresses the kilobytes allocated for the file data. FILE_SIZE and KB_ALLOCATED may not be identical because a file may allocated more data blocks than its actual size. The fundamental difference between FILE_SIZE and KB_ALLOCATED is that after a file has been migrated the FILE_SIZE does not change, but KB_ALLOCATED is 0 (assuming the stub-size is 0). If the stub-size is larger than 0 then KB_ALLOCATED is greater than 0 and identical to the stub-size. Note, the stub-size must be a multiple of the IBM Spectrum Scale file system block size.

File attributes can also be listed by LIST rules. More details for this can be found in section Showing file attributes.

## Placement

A placement rule instructs IBM Spectrum Scale to store files matching certain criteria in a certain file system pool. The most common criteria for placement are typically the name of the path and files. The name of the path can be the name of the fileset (see section Fileset placement). File system pools selected for placement should allow storing data (usage dataAndMetadata or dataOnly), otherwise files matching the placement rule criteria cannot be created (see section Prevent storing certain file types). Data cannot be placed in external pools (see section External pools).

There can be multiple placement rules for a file system, all in one policy. However, too many and complex placement rules in a policy for a given file system can slow down the system because a placement policy is evaluated every time a file is created in the file system. The most basic advice is: “Make it simple”.

The basic syntax of a placement rule is:

```
RULE 'rule-name' SET POOL 'pool-name'
WHERE (NAME) LIKE '%match-pattern%'
```

This rule has the name *rule-name* and instructs placing file names matching to *match-pattern* in pool *pool-name*. Rule names are optional, but provide significant benefits when it comes to debugging and problem determination.

A placement policy for a file system includes one or more placement rules. The rules are evaluated top down. The last rule should always be a default placement rule which places all files not matching prior criteria. The following example assumes that the file system has two pools: ‘system’ and ‘silver’, as shown in figure 1 (see IBM Spectrum Scale file systems and pools). Both pools are comprised of (some) NSD allowing for data usage. This policy is comprised of two rules and will place files ending with “.mp3” on pool ‘silver’ (rule name is mp3-rule), while all other files are placed on ‘system’ pool

```
RULE 'mp3-rule' SET POOL 'silver'
WHERE LOWER (NAME) LIKE '%.mp3%'

RULE 'default' SET POOL 'system'
```

The first rule will place all file names ending with .mp3 on the ‘silver’ pool. The LOWER expression causes the file name to be transformed in lower letters assuring that all .mp3 and .MP3 files are captured. The second rule places all other files in the ‘system’ pool. The order of the rules is important because they are evaluate from top to bottom. Therefore, the rule named ‘default’ must be at the end of the placement rules, otherwise all files would be placed on ‘system’ pool.

The command `mmlsattr` can be used to check the various characteristics of a file including the storage pool where the file is located in. The following example shows that the file is placed on ‘system’ pool:

```
# mmlsattr -L /filesystem/file1
file name:            /filesystem/file1
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    system
fileset name:         root
snapshot name:
creation time:        Wed Sep  3 13:19:39 2014
Windows attributes:   ARCHIVE
```

For more use cases and examples of placement rules refer to section Examples and use cases.

### Applying placement policies

Placement rules must be applied to a file system as an automated policy using the *mmchpolicy* command. The mmchpolicy command allows applying one policy comprising multiple rules to a file system. In order to apply a policy for a file system it must be written to a file.

The following example shows how to apply a placement policy stored in file *filesystem_policy.txt* to the file system *fsname*:

```
# cat filesystem_policy.txt

RULE 'mp3-rule' SET POOL 'silver' WHERE LOWER (NAME) LIKE '%.mp3'

RULE 'default' SET POOL 'system'

# mmchpolicy fsname -P filesystem_policy.txt
```

The mmchpolicy command also supports an option "-I test" allowing to test the syntax of the policy before applying it to the file system. Placement rules cannot be applied with the mmapplypolicy command. In sections Automated migration and Examples and use cases more comprehensive examples for placement policies are shown, also in combination with rules for migration.

## Migration

A migration rule instructs IBM Spectrum Scale to identify files according to certain criteria and move these files from one storage pool (internal) to another (internal or external). The files being migrated remain accessible in the global name space of the IBM Spectrum Scale file system, so the migration is transparent to the user. When migrating from an internal pool to another internal pool - such as from the 'system' pool to the 'silver' pool (figure 1) - the files remain under the control of IBM Spectrum

Scale. When migrating from an internal pool to an external pool - such as from 'silver' pool to external pool - the files migrated to external pool are under control of the interface script configured with the external pool (see section External pools). Typically, a stub-file is left in the IBM Spectrum Scale files system (on the internal pool) that references the file stored in the external pool. This stub-file is represented by an inode with no data blocks. When the user accesses the stub-file the file data will automatically be recalled.

The basic syntax for a migration rule is:

```
RULE 'rule-name' MIGRATE FROM POOL 's-pool'
THRESHOLD(%high,%low,%premig) WEIGHT(expression) TO POOL 'd-pool'
FOR FILESET ('fileset-name') WHERE (conditions)
```

The verb MIGRATE defines this as a migration rule. The migration is done from a source pool (s-pool) to a destination pool (d-pool), both must exist for this file system. Note, the source pool be lower in the storage hierarchy than the destination pool, e.g. allowing migrating from tape back to disk. With the THRESHOLD expression the migration can be triggered based on occupancy and is typically used with automated migration or pre-migration. The THRESHOLD parameter allows for three values:

**%high:** The rule is only applied if the occupancy percentage of the FROM pool is greater than or equal to the %high value.

**%low:** The rule stops if the occupancy percentage of the FROM pool is reduced to less than or equal to the %low value.

**%premig:** Optional, defines an occupancy percentage of a storage pool that is below the lower limit (%low). Files are pre-migrated if the storage pool occupancy is between the %low and the %premig limit. The value of %premig must be between 0 and the %low value. The amount of files being pre-migrated is equivalent to (%low - %premig), however the occupancy of the pool does not change because pre-migration copies the files to the external pool (for more information about pre-migration refer to section Pre-migration).

The WEIGHT expression files can be prioritized for migration, e.g. based on size (KB_ALLOCATED) or access time (ACCESS_TIME). The FILESET expression allows further filtering of files subject for migration based on their storage location within the file system directory structure. The WHERE expression describes further conditions.

As described in section IBM Spectrum Scale policy engine there are essentially two kind of migration policies: automated and scheduled migrations. Automated migration is typically based on thresholds configured in a MIGRATE rule (see Automated migration). It is invoked automatically when the threshold is reached. The threshold denotes an occupation percentage of a file system pool. Scheduled migration is run manually or by a schedule and is typically independent of thresholds (see section Scheduled migration). With scheduled migration ILM requirements independent of the storage pool occupation can be implemented, such as based on file age or other criteria. Typical criteria for file migration are:

- File size
- Files and directory names, including fileset name
- Files types based on file names
- File ownership, etc.

A special form of migration is pre-migration (see section Pre-migration). With a pre-migration policy files are copied to an external pool. Thus, the file is dual resident, on the internal and on the external pool. Pre-migration only works in when migrating from an internal to an external pool.

When migrating files it might be desirable to exclude certain files based on path and / or file names or other attributes (see section Excluding files and directories. This is particularly important to consider when migrating to external pools (see External pools) because access to files on external pools might be slow.

## Automated migration

*Automated migration policies* typically include one or more threshold-based rules, indicating that when a file system pool reaches a certain occupation percentage, files should be migrated to another pool (internal or external). Each file system can have its own automated migration rules within an active policy. There can only be one active policy per file system and an active policy can contain automated migration rules as well as placement and encryption rules. The syntax of a threshold-based migration rule is:

```
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'silver'
```

This rule instructs the IBM Spectrum Scale policy engine to migrate files from pool 'system' to pool 'silver' (see figure 1 in section IBM Spectrum Scale file systems and pools) when the pool 'system' is occupied 80 % or more. The migration is done until the 'system' pool occupation drops below 70 %. The %premig value is not used in this rule because pre-migration is not required. The WEIGHT statement causes files to be ordered for migration based on access time, oldest files are migrated first.

An automated migration policy is activated using the `mmchpolicy` command. There can only be one policy per file system, so the policy file being activated must include migration and placement rules (and others when required). Extending the example for the placement policy (see section Placement rules) with a migration policy the policy file and the mmchpolicy command for file system /filesystem looks like this:

```
# cat filesystem_policy.txt

/* here comes the migration rule */
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'silver'

/* here is the placement rules */
RULE 'mp3-rule' SET POOL 'silver'
WHERE LOWER (NAME) LIKE '%.mp3%'


RULE 'default' SET POOL 'system'

#
```

```
# mmchpolicy filesystem -P filesystem_policy.txt
```

The last line in the example above is the `mmchpolicy` command used to applying the policy to the file system (see section Running a policy). In addition to applying the policy to the file system a callback must be configured. A callback is invoked upon pre-defined events and calls a script. For more details see section Migration Callback.

More examples for automated migration rules are given in section Threshold based migration. The most basic advice for automated migration rules is again: Make it simple! When the occupancy threshold of a pool is reached it is important to move files to another pool as quickly and efficiently as possible. Sophisticated rules may not achieve this, resulting in files not being moved away and the source pool becoming over-occupied. Over-occupied pools can cause the IBM Spectrum Scale file system to go offline.

Rules shown above are valid for automated migration between internal pools, such as pools 'system' and 'silver' according to figure 1. When migrating to external pools an additional rule is required (see section External pools).

Note, the automated migration can only be triggered by the occupation percentage of a storage pool and not by the quota occupation of filesets. In section Fileset Quota based migration we discuss an approach to trigger the migration when quota limits for filesets have been exceeded.

## Migration Callback

To start an automated migration policy when a certain threshold is reached for a file system pool, callback must be configured in the IBM Spectrum Scale cluster. A callback is triggered upon defined events and invokes a defined script to be executed. The callback also allows sending additional parameters to this script. A callback has a cluster-wide scope and is invoked for all file systems when one of the events is triggered.

To configure a callback for migration, the events *lowDiskSpace* and *noDiskSpace* must be specified. There is a standard script which starts the policies configured for the file systems, called `/usr/lpp/mmfs/bin/mmstartpolicy`. The following command shows the configuration of a standard migration callback:

```
# mmaddcallback MIGRATION --command
/usr/lpp/mmfs/bin/mmstartpolicy
--event lowDiskSpace,noDiskSpace
--parms "%eventName %fsName -m 3 -N eenode1,eenode2 -B 1000
-n 1 -s <temp-dir> --single-instance”
```

This callback is named MIGRATION and is triggered by the events "lowDiskSpace" and "noDiskSpace". When one of these events occurs then IBM Spectrum Scale automatically invokes the callback script /usr/lpp/mmfs/bin/mmstartpolicy with the parameters given with the -parms string (for an explanation of these parameters see section Running a policy). The script /usr/lpp/mmfs/bin/mmstartpolicy will invoke the mmapplypolicy command with the file system name and the mmapplypolicy parameters given with the --parms option. The command mmapplypolicy will run the policy applied with mmchpolicy.

When migrating from an internal pool to another internal pool and from there to an external pool within one file system special care must be taken for the callback. Because the parameters for invoking the policy engine within the callback script might be different when migrating to an external pool. For more details of a migration callback in a file system with two internal pools and one external pool is given in section Three pool migration.

## Scheduled migration

*Scheduled migration policies* can be - but do not have to be - threshold based. The basic concept of scheduled policies is to migrate files to another pool before the automated policy is invoked. More specifically, the purpose of a scheduled policy is to migrate enough files so that an automated policy never kicks in. The basic syntax of a scheduled migration rule is:

```
RULE 'manMigSystem' MIGRATE FROM POOL 'system' TO POOL 'silver'
WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 30)
```

This rule instructs the IBM Spectrum Scale policy engine to migrate files last accessed 30 days ago or longer from pool 'system' to pool 'silver' (see figure 1 in section IBM Spectrum Scale file systems and pools). This rule is only valid for migration between internal pools, such as pools 'system' and 'silver' according to figure 1. When migrating to external pools an additional rule is required (see section External pools).

To execute a scheduled policy the `mmapplypolicy` command is used (see section Running a policy). More examples for scheduled migration policies are given in section Manual or scheduled Migration.

## External pools

As explained in section IBM Spectrum Scale file systems and pools an *external pool* is an external storage device that is represented by an interface script. An external pool does not have to be disks based, it can also be based on tape or other storage devices. Storing and retrieving data (files) in an external pool is managed by the interface script. Typical examples for external pools are IBM Spectrum Archive Enterprise Edition (LTFS EE), IBM Spectrum Protect for Space Management (SP HSM) and HPSS (GPFS-HPSS-Interface)

In contrast to internal pools an external pool must be defined with an additional rule (EXTERNAL POOL rule). This additional rule essentially defines the name of the external pool and the interface script to be invoked, as shown below:

```
RULE EXTERNAL POOL 'ext-pool' EXEC 'program' OPTS option

RULE 'rule-name' MIGRATE FROM POOL 'int-pool' TO POOL 'ext-pool'
WHERE (conditions)
```

The first rule defines the external pool with the name "ext-pool". The OPTS parameter allows passing additional options to the external pool script (see also Processing file lists). In the context of IBM Spectrum Archive EE, the OPTS parameter denotes the migration destination tape pool and the library.

The second rule defines to migrate from an internal pool named “int-pool” to the external pool “ext-pool” under certain conditions.

**Note,** the interface script must be installed on all nodes that perform the migration to external pools. In addition, the policy engine (mmapplypolicy) must be invoked on a node that has the interface script installed.

The following example shows an automated migration policy from an internal pool ‘silver’ to an external pool ‘ltfs’ when the ‘silver’ pool is 90 % occupied:

```
/* Exclude rule to exclude certain directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define ltfs as external pool with Tapepool as destination */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

/* migration rule */
RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'ltfs'
```

The first EXCLUDE rule defines path names to be excluded. When migrating to an external pool managed by IBM Spectrum Archive EE it is important to exclude at least these subdirectories (see section Excluding files and directories). The path name ‘%/.ltfsee/%’ refers to the actual directory of the IBM Spectrum Archive EE metadata directory, if this has been setup in the space managed file system.

The second EXTERNAL POOL rule defines an external pool named ‘ltfs’ which is managed by the interface script `/opt/ibm/ltfsee/bin/eeadm`. The `eeadm` command is invoked with the migrate option because the third rule is a MIGRATE rule. The OPTS parameter denotes the IBM Spectrum Archive EE tape pool and library names used as destination of the migration. Up to 3 pool @ library statements can be given here (separated by comma) whereby the migration job creates 3 copies in 3 different tape pools (see section Creating multiple copies on tape). Alternatively, the mmapplypolicy command can be used to pass the appropriate tape pool name to the policy before it is interpreted by the policy engine (see section Passing strings to rules). The SIZE clause defines the total size of all files passed in one file list to the `eeadm migrate` command. In this case the total size is 10 GB. If more than 10 GB of file capacity has been selected by the policy then multiple `eeadm migrate` commands with different file lists are executed.

The third rule is a migration is a migration rule that defines to migrate from ‘silver’ pool to the external pool ‘ltfs’ if the ‘silver’ pool occupation is above 90 %. Files to be migrated are ordered by their access time, whereby oldest files are on top of the list.

This policy example can be combined with the automated policy for migrating from 'system' to 'silver' pool (see section Automated migration) and the placement policy for mp3-files (see section Placement):

```
/* Exclude rule to exclude certain directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* here comes the migration rule for 'system' to silver */
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'silver'

/* here comes the migration rules for silver to ltfs*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'ltfs'

/* here comes the placement rules */
RULE 'mp3-rule' SET POOL 'silver'
WHERE LOWER (NAME) LIKE '%.mp3%'

RULE 'default' SET POOL 'system'
```

This policy defines four things:

- Defines an exclude rule for directories in the file system that should not be migrated (see section Excluding files and directories)
- Threshold based migration from pool 'system' to pool 'silver' when 'system' pool has reached 80 %
- Threshold based migration from pool 'silver' to pool when 'silver' pool has reached 90 %, The pool "ltfs" is represented by tape pool Pool1 in Library Lib1
- Placement policy for mp3-files on 'silver' pool, all other files on 'system' pool

This policy must be activated using the `mmchpolicy` command (see section Automated (threshold-based) migration).

The following policy example for a scheduled policy will migrate files last accessed 30 days ago or longer from pool 'silver' to pool ltfs:

```
/* Exclude rule to exclude certain directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define ltfs pool and migration rule */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

RULE 'manMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'ltfs'
WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 30)
```

This policy first defines an exclude rule (see Excluding files and directories) and then migrates files accessed 30 days ago or longer from pool ‘silver’ to pool ‘ltfs’ that is managed by IBM Spectrum Archive EE. Files are migrated to the IBM Spectrum Archive EE pool Pool1@Lib1.

To run this policy, see section Running a policy. More examples for policies and use cases with external pools can be found in section Examples and use cases.

### Creating multiple copies on tape

IBM Spectrum Archive EE allows to create up to 3 copies of any file on up to 3 different tapes during one migration job. This is accomplished by specifying multiple tape pools in the external pool definition, as shown in this example below. The tape pools can consists of tapes located in up to 2 tape libraries.

The following policy defines a rule to migration all files older than 30 days to two tape pools. One tape pool (Pool1@Lib1) is comprised of tapes in Library1 and the second tape pool (Pool2@Lib2) is comprised of tapes in Library2.

```
/* Exclude rule to exclude certain directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define ltfs pool and migration rule */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm' OPTS
'-p pool1@lib1,pool2@lib2'

RULE 'manMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'ltfs'
WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 30)
```

The migration job will create two copies on a least two different tape for each selected file: one on Pool1@Lib1 and one on Pool2@Lib2. In order to leverage a second library, the IBM Spectrum Archive EE must be configured for multi-library support and must have attached two distinct libraries. But even with only one tape library up to three copies on three different tape pools can be created.

Note, when a file with multiple copies on tape is recalled then the tape according to the first pool in the list is used. If this tape is damaged IBM Spectrum Archive EE will automatically recall the file from an additional copy tape, when possible.

## Migrating from external pool to internal pool

Files migrated to LTFS tape can also be migrated back (recalled) to internal disk using the policy engine. The file attributes used for file selection however are very limited. The following policy migrates all files in directory */ibm/gpfs0/data1* from the associated tapes to the IBM Spectrum Scale file system. Please note that the file system pool ('system' in this case) matters. It must be the same pool from which the files have been migrated in the first place.

```
/* recall all files under /ibm/gpfs0/data1 directory */

RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'

RULE 'ee_recall' MIGRATE FROM POOL 'ltfs' TO POOL 'system'
WHERE LOWER(PATH_NAME) LIKE '/ibm/gpfs0/data1/%'
```

As shown in the example above the OPTS statement in the EXTERNAL POOL rule can be omitted. Likewise, the path for .Spaceman and .snapshots do not have to be excluded because files from these directories should not have been migrated to tape in the first place. Note, the use of FOR FILESET does not work for the recall policy.

This policy can be run using the mmapplypolicy-command (see section Running a policy).

## External pool considerations

Migrated files that are in an IBM Spectrum Scale file system or fileset *snapshot* will be recalled automatically when the migrated file is deleted in IBM Spectrum Scale. When deleting multiple migrated files this may cause massive recall. Therefore, it is recommended to not use IBM Spectrum Scale file system or fileset snapshots in a space managed file system.

Using IBM Spectrum Scale *Active File Management* (AFM) on a fileset where files are migrated to an external pool has some limitations depending on the role of the fileset [4]. IBM Spectrum Archive EE only support AFM Independent Writer mode. AFM is based on a home - cache model. In a simple AFM configuration, the home IBM Spectrum Scale cluster exports a fileset which is cached on the cache fileset of the cache IBM Spectrum Scale cluster. The following considerations apply:

- When migrating files on home the pre-fetch operation to cache may cause massive recalls on home when migrated files are to be pre-fetched. The recommendation is to use tape optimized recalls prior to the pre-fetching operation. The IBM Spectrum Scale policy engine on home can be used to identify files for recall and pre-fetch.
- When migrating files on home the replication of modified files from cache to home may cause massive recalls. The recommendation is to migrate files on home when they are no longer changed on cache.
- When migrating files on cache using IBM Spectrum Archive EE then you must enable AFM file state checking. To do this run the command: `ltfsee_config -m UPDATE_FS_INFO`.
- When migrating files on cache set the option AFMSKIPUNCACHEDFILES in the HSM client option file (`/opt/tivoli/tsm/client/ba/bin/dsm.sys`)
- When migrating files on cache ensure that files that are in status “dirty” are not migrated, otherwise this may cause error messages. To prevent migration of dirty files, configure the

migration policy to migrate files that have been modified a reasonable amount of time before they are migrated.

When invoking the policy engine for the migration to an external pool some specific parameters should be considered (see section Running a policy). This may also apply to the migration callback which also invokes the policy engine (see section Migration Callback).

Prevent migration of *certain sub-directories used for IBM Spectrum Scale internal processing* in a file system or fileset. IBM Spectrum Scale internal processing directories are place in the file system or fileset root and start with a dot ("."). To prevent migration of such directories, EXCLUDE rules or macros can be used (see section Excluding files and directories).

## Excluding files and directories

For migration policies, especially when migrating to external pools, it is important to exclude certain files and directories in order to prevent these from being migrated to an external pool. Files and directories to be excluded are relative to the file system directory the migration policy is applied for. The most common files and directories to be excludes are:

- Directory `.SpaceMan` including configuration and temp files for HSM
- Directory `.snapshots` including snapshots
- Directory for the IBM Spectrum Archive EE metadata cache (`.ltfsee`) if this has been setup in the space managed file system
- Directory for the temporary files of the policy engine `.mmSharedTmpDir`
- When the `mmbackup` command is used then exclude the shadow data base files (`.mmbackupShadow`)

Depending on the functions configured for the IBM Spectrum Scale file system there may be more files and directories to be considered, see section Further excludes.

There are two ways to exclude file from being migrated using the IBM Spectrum Scale policy engine:

- Using EXCLUDE rules within a migration policy (section Using EXCLUDE rules)
- Using the WHERE clause of a migration rule (in combination with macros, see section Using the WHERE clause)

The fundamental difference between both methods is at which point of the policy run (see section IBM Spectrum Scale policy engine) files are excluded. Depending on the complexity of the exclude statements this can have a performance impact on the policy run. With EXCLUDE rules files are excluded before they are being evaluated by any migration policy while exclusion within the WHERE clause are evaluated by the migration policy. The latter one is therefore slower.

**Note:** EXCLUDE rules are not supported with LIST policies.

## Using EXCLUDE rules

IBM Spectrum Scale provides a special rule type for excluding files and directories from being processed for migration (and deletion). The basic syntax of the exclude rule is:

```
RULE ['RuleName'] EXCLUDE [DIRECTORIES_PLUS]
[FOR FILESET ('FilesetName'[,'FilesetName']...)]
[WHERE SqlExpression]
```

The DIRECTORY_PLUS statement indicates that the not only files but also directories matching the rule should be considered. An exclude rule can also operate on a fileset level only considering files and directories in the named filesets. The WHERE clause allows to specify files attributes of files to be excluded.

The exclude rules must be placed before the migration rules in a policy. The exclude rules are applied to all migration rules within a policy. This may require separating migration rules in different policies if they cannot share the same exclude rule. The following exclude rule excludes all directories that should not be migrated:

```
/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')
```

In this rule the path .Spaceman, .snapshots, .ltfsee and .mmSharedTmpDir as well as all files matching the name pattern of `.mmbackupShadow` are excluded from migration.

The automated policy according to figure 1, for migrating from 'system' to 'silver' and from 'silver' to 'ltfs' including the exclude rule looks like this:

```
/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* here comes the migration rule for system to silver */
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'silver'

/* here comes the migration rules for silver to ltfs*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'ltfs'
```

```
/* here comes the placement rules */
RULE 'mp3-rule' SET POOL 'silver' WHERE LOWER (NAME) LIKE %.mp3%'
RULE 'default' SET POOL 'system'
```

A scheduled policy for migrating from pool ‘silver’ to pool ‘ltfs’ for files that are older than 60 days with an exclude list looks like this:

```
/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* migrate to ltfs */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

RULE 'manMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'ltfs'
WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 60)
```

Excluding files using exclude rules is the recommended and most efficient way for excluding files from being migrated. The exclude rule is not applied to LIST rules, LIST rules provide an exclude statement (see Lists).

**Note:** EXCLUDE rules are not supported with LIST policies. For LIST policies exclude macros must be used.

## Using the WHERE clause

Another way for excluding files from migration is within the migration rule WHERE clause. In this context exclude lists can be defined by macros and appended to the WHERE clause of the rules (see section Macros). To define an exclude list for these directories the following macro can be used:

```
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))
```

This macro must be embedded in the policy as shown in the next example. The automated policy according to figure 1, for migrating from ‘system’ to ‘silver’ and from ‘silver’ to ‘ltfs’ including the exclude list looks like this:

```
/* define exclude list as macro */
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* here comes the migration rule for system to silver */
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'silver'

/* here comes the migration rules for silver to ltfs*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'ltfs' WHERE
NOT (exclude_list)

/* here comes the placement rules */
RULE 'mp3-rule' SET POOL 'silver' WHERE LOWER (NAME) LIKE %.mp3%'
RULE 'default' SET POOL 'system'
```

The exclude list is defined first and does not have to be applied for the migration rule from pool ‘system’ to pool ‘silver’ because these are internal pools (1. Rule after the macro). It is been used in the migration rule to the external pool (3. Rule after the macro).

A scheduled policy for migrating from ‘silver’ to ‘ltfs’ for files that are older than 60 days with an exclude list looks like this:

```
/* define exclude list as macro */
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* migrate to ltfs */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p pool1@lib1’
SIZE 10485760

RULE 'manMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'ltfs'
WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 60) AND
NOT (exclude_list)
```

The macro named ‘exclude_list’ is executed within the migration rule which can add significant overhead because it had identified all files before and excludes files within the migration rule. A better way for excluding files might be using EXCLUDE rules (see Using EXCLUDE rules) because files are excluded before they are processed by the migration rule.

For more examples of rules and policies with and without exclude list see section Examples and use cases.

### Further excludes

Beside the common file and directories to be excluded (see section Using EXCLUDE rules) there may be more files and directories to be excluded in accordance to the functions configured for the IBM Spectrum Scale file system. Find some recommendations below.

The clauses below should be inserted within the WHERE clause of the EXCLUDE rule or macro:

When mmbackup is used consider the following excludes

```
/* mmbackup related excludes */
   PATH_NAME LIKE '%.mmbackupCfg/%' OR
   NAME LIKE '.mmbackupShadow%' OR
   NAME LIKE '.mmbackup%')
```

When Active File Management is used consider the following excludes:

```
/*AFM related excludes */
  PATH_NAME LIKE '%.afm/%' OR
  PATH_NAME LIKE '%.ptrash/%' OR
  PATH_NAME LIKE '%.pconflicts/%'
```

For older versions of IBM Spectrum Scale (< version 5) the following excludes should be considered:

```
/* artefacts from older versions */
  PATH_NAME LIKE '%.ctdb/%' OR
  NAME LIKE 'user.quota' OR
  NAME LIKE 'group.quota' OR
  NAME LIKE 'fileset.quota'
```

## Pre-migration

With pre-migration files are copied from an internal pool to an external pool, unlike migration where files are moved. A pre-migrated file is dual resident, in the internal pool and in the external pool. Pre-migration is only possible when migrating from an internal pool to an external pool. After pre-migration has finished the file is available in the internal pool and in the external pool. For more information about external pools see section External pools.

The main purpose of pre-migration is to copy files to the external pool while the internal pool has not reached the migration threshold. If the pool reaches the migration threshold the migration process is fast because it does not have to transfer files again to the external pool. Instead it just creates the stubs. With pre-migration the efficiency of ILM policies can be optimized.

Keep in mind, pre-migration is not a backup because if the file in the internal pool is deleted there is no reference to the file on the external pool. Thus, it is not trivial to recover a deleted file, which contradicts the idea of backup focusing on recovery of files.

Pre-migration can be defined in a MIGRATE rule using the THRESHOLD parameter. The THRESHOLD parameter allows for three values: THRESHOLD(%high, %low, %premig)

> **%high:** The rule is only applied if the occupancy percentage of the source pool is greater than or equal to the %high value.
>
> **%low:** If the occupancy percentage of the source pool is greater than the %low value files are migrated until the %value is met.
>
> **%premig:** Defines an occupancy percentage of a storage pool that is below the lower limit (%low). Files are pre-migrated if the storage pool occupancy is between the %low and the %premig limit. The value of %premig must be between 0 and the %low value. The number of files being pre-migrated is equivalent to (%low - %premig) of the total pool capacity, however the occupancy of the pool does not change because pre-migration copies the files to the external pool.

The pre-migration rule only kicks in if the storage pool occupancy is above the %high percentage. If this is the case and the storage pool occupancy is between %low and %premig then files are pre-migrated. Otherwise if the storage pool occupancy is above %low then files are migrated until the %low value is met. Afterwards files are pre-migrated if the storage pool occupancy lies between %low and %premig. Because the storage pool percentage does not change with pre-migration the pre-migration processes a capacity which is equivalent to (%low - %premig) of the total pool capacity.

For example, the following rule pre-migrates all files to tape if the ‘silver’ pool occupancy lies between 0 - 30 %. If the storage pool occupancy is above 30 % then files are migrated until the storage pool occupancy drops below 30 % before files are pre-migrated:

```
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p pool1@lib1’
SIZE 10485760

/* DON NOT USE THIS AS AUTOMATED MIGRATION POLICY */

RULE 'premig1' MIGRATE FROM POOL 'silver' THRESHOLD (0,30,0) TO
POOL 'ltfs'
```

Because pre-migration only works from internal to external pools two rules are required, one defining the external pool managed by IBM Spectrum Archive EE (see section External pools) and one defining the pre-migration. This policy can be run in a scheduled manner using the mmapplypolicy command (see section Running a policy). It shall not be run as an automated policy because the %high value of 0% would trigger constant policy engine runs.

Pre-migration in conjunction with automated policies should be well thought through because %high triggers the migration and it might not be appropriate to set %high to 0. The rule in the following example kicks in if the ‘silver’ pool occupancy is above 80 %. It first migrates files to pool ‘ltfs’ until the ‘silver’ pool occupancy drops below 70 %. Afterwards it pre-migrates files from pool ‘silver’ to pool ‘ltfs’ whereby the volume of files being pre-migrated is approx. 60% of the total ‘silver’ pool capacity (70 - 10). This policy could be used in conjunction with automated migration:

```
/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define ltfs pool and migration rule */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p pool1@lib1’
SIZE 10485760

RULE 'premig2' MIGRATE FROM POOL 'silver' THRESHOLD (80,70,10)
TO POOL 'ltfs'
```

The example above first defines the exclude list (see section Excluding files and directories), then defines the pool ‘ltfs’ managed by IBM Spectrum Archive EE and finally defines the pre-migration rule. On the first run, this policy migrates 10% of the data (80 - 70) and pre-migrates a volume of 60%. On the next run the migration of 10% is quick because it already pre-migrated.

One use case for pre-migration might be to pre-migrate everything within a certain fileset. In this case the values for %high and %premig should be 0 and the value for %low should be reasonably high. The policy below pre-migrates all files in fileset ‘test’ if the occupancy of the pool ‘silver’ is below 70%:

```
RULE 'premig2' MIGRATE FROM POOL 'silver' THRESHOLD (0,70,0)
TO POOL 'ltfs' FOR FILESET ('test')
```

Running the policy above as automated policy is not appropriate because it would trigger constantly. Therefore, it might be considerable use scheduled policies to control pre-migration and automated policies for migration only. For more details and examples see section Pre-migration and migration.

An alternate method for running pre-migration in a scheduled manner is explained in section Using customized external script).

## Running a policy

The IBM Spectrum Scale policy engine is represented by the command: `mmapplypolicy` [3]. As explained in section Migration, there are two kind of policies: one for Automated migration and one for Scheduled migration. The way the mmapplypolicy command is invoked differs between the two kind of policies.

## Running an active policy

Each file system can have exactly one active policy that can be configured to run automated migration and other file management actions such as placement, encryption, etc. The automated policy for a file system is activated with the command:

```
# mmchpolicy filesystem policyfile -I test | yes
```

This command takes the name of the file system (parameter 'filesystem') and the name of the file containing the policy rules of the automated policy (parameter '*policyfile*'). The options `-I test` allows to check the syntax prior to applying this policy.

Once the automated policy is applied to the file system it is active in the background. If the automated policy contains threshold based MIGRATE rules then IBM Spectrum Scale observes the utilization of the file system pools and if the %high threshold is met it throws an event (lowDiskSpace). This event is caught by the callback (see section Migration Callback) and the callback executes the MIGRATE rule of the active policy using the command mmapplypolicy.

The callback configuration also allows to pass certain options to the mmapplypolicy command (see sections Migration Callback and Options for mmapplypolicy).

To show the active policy configured for a file system the following command can be used:

```
# mmlspolicy filesystem -L
```

When an active policy must be changed, then the associated policy file can be changed, and these changes are activated using the `mmchpolicy` command.

## Running a scheduled policy

With scheduled migrations the `mmapplypolicy` command is invoked by the scheduler or manually by an administrator. The basic syntax of the `mmapplypolicy` command is [3]:

```
# mmapplypolicy filesystem -P policyfile [options]
```

The *filesystem* is the name of the file system or sub-directory which needs to be scanned. The *policyfile* name is the name of the file including the policy to run. Some important options are further explained in section Options for mmapplypolicy.

**Note,** when performing the migration to external pools the policy engine (`mmapplypolicy`) must be invoked on a node that has the interface script installed. The interface script is defined in the external pool rule.

## Options for mmapplypolicy

There are several options for the `mmapplypolicy` command [3], most options have default values. When migrating from an internal pool to another internal pool the default options are typically enough. However, when migrating from an internal pool to an external pool managed by IBM Spectrum Archive EE some `mmapplypolicy` options must be considered:

| Option | Description | IBM Spectrum Archive EE Consideration |
|---|---|---|
| -m | Number of threads dispatched for file management operation (e.g. migration) on every participating node | should be set to number of tape drives / number of nodes. E.g. with 2 nodes and 6 drives -m should be set to 3 at maximum. |
| -B | Bucket size defines the number of files each file management thread obtains | Depends on the size of the file and the total number of files to be migrated. For small files choose a higher number (e.g. 10.000), for large files choose a lower number (e.g. 1000). |
| -n | Number of directory scan threads on every node participating in the policy scan | choose a low number like 1 |
| -N | Node names executing the policy scans | Node names of the nodes where IBM Spectrum Archive EE is installed and running. Must be specified.<br>If not specified then nodes specified in config parameter `defaultHelperNodes` are used. If this config parameter is not specified the nodes with manager role may be used. |
| -s <dir> | Directory to store local temporary files | Depends on number of files in file system to be scanned and selected. Default is `/tmp`. For large file systems ensure that this directory is large enough. |
| -g <dir> | Directory to store global temporary files | Depends on number of files in file system to be scanned and selected. Default is defined by config parameter `sharedTmpDir`. Default at Spectrum Scale version 5.0 is .mmSharedTmpDir in the root of the file system |
| --single-instance | Only one instance of mmapplypolicy can run for this file system | Should be used to prevent conflicting migration jobs |

| Option | Description | IBM Spectrum Archive EE Consideration |
|---|---|---|
| -I | `test`: scan the file system and do not run the actual operation (e.g. migration)<br><br>`defer`: create the file lists, but do not run the actual file operation (e.g. migration)<br><br>`yes`: run the policy and file operations (default) | Use -I test to test the policy syntax.<br><br>Use -I defer with LIST policies to generate the file lists. |
| -L | Debug level for mmapplypolicy | Set to 4 - 6 for testing and debugging. It will show the selected files. Use in conjunction with -I test. |
| -M | Allows to substitute strings in the policy file | Special use cases |

The *parameters -m and -B* let you control the number of files being migrated in parallel. The number of files being migrated by one migrate thread is specified with the -B parameter. The number of threads per node is specified with the -m parameter. The number of files being scheduled for migration at a given point in time is calculated by:

Number_of_Files_migrated = N x m x B

With IBM Spectrum Archive EE it is important to control the number of files being dispatched for migration or recalls, see section Controlling the number of files processed for more details.

Also take a note of the *-s parameter*. As more files the file system has as more temporary space is required, this can be multiple tens of Gigabytes. Make sure that -s specifies a directory that has enough storage capacity. Perhaps use a subdirectory within the IBM Spectrum Scale file system as temporary work directory, however, do not forget to exclude it from migration. Note, when setting -s, this will also set the -g parameter (global work directory).

The *parameter -single-instance* assures that at one point of time only one mmapplypolicy process is running. For automated migration using active policies, this parameter is recommended to be used. For schedule migrations it depends on the likelihood that migrations jobs overlap.

The *parameter -I test* should always be used before running the policies. In test mode not only the syntax of your policy is checked, but you also get an idea how many files are being migrated. This parameter combined with the -L parameter also shows the names of the files being selected for migration.

With the *parameter -L <n>* mmapplypolicy can be run in debug mode. When using parameter `-L 5` this will show the names of the files being selected for migration. It makes a lot of sense to use this option in test mode (-I test) to check whether the policy selects the appropriate files.

To test a scheduled policy migrating to an external pool and stored in file name *mypolicy.txt* for file system named *filesystem* the option `-I test` can be used:

```
# mmapplypolicy filesystem -P mypolicy.txt -N eenode1,eenode2 -I
test
```

To run this policy with the recommended options for an external pool, the following command can be used:

```
# mmapplypolicy filesystem -P mypolicy.txt -m 3 -n 1 -B 1000 -N
eenode1,eenode2 -s /tempspace --single-instance
```

With IBM Spectrum Archive EE, it is important to control the number of files being processed at the same time, refer to the next sub-section for more guidance.

## Controlling the number of files processed

Internally IBM Spectrum Archive EE uses a queue for files being scheduled for migration or recall. This queue is used to track the names of the files being processed

When the policy engine has selected files for migration (or recall) it starts m threads on each participating IBM Spectrum Archive EE node and each thread processes a list of B files. The numbers m and B are the values given with the mmapplypolicy parameter: -m and -B. In addition, the SIZE clause that can be provided with the EXTERNAL pool definition (see section External pools) influences the number of files in the file list provided to each migration thread. The number of files in each file list is either limited by the bucket size (-B) or by the SIZE clause in the EXTERNAL pool definition. The bucket size defines the maximum number of files in one file list. The SIZE clause defines the maximum total size of all files provided in one file list. The first limit is applied. E.g. assume the bucket size is 10.000 and the SIZE clause is 10 GB. If the total capacity of a file list with 8.000 is 10 GB then the file list passed to one migration task contains 8.000 file names.

Each thread essentially starts the `eeadm migrate` command with the list of file names received from the policy engine and limited by either -B or the SIZE clause. IBM Spectrum Archive EE puts these file names on the queue and then schedules the copy processes from disk to tape. These copy processes are internally optimized based on utilization of the tape resources and the destination pools. The maximum number of files on the queue for a given migration process can be calculated with the following formula:

Files_on_queue = (Number of EE nodes) x (parameter m) x (parameter B)

If one file list has been processed the associated thread is ended and another thread is started, until all files selected by the policy engine have been processed.

The general recommendation of the maximum number of files on the queue is the following:

| **Operation** | **Max. number of files o queue** |
|---|---|
| Migration | 50.000 |
| Recall | 20.000 |

For example, let's assume the policy engine is started with the parameters: -m 3 -B 1000 and the number of IBM Spectrum Archive EE Nodes participating in the migration is 2. The total number of files that appear on the queue is: 2 x 3 x 1000 = 6000. If the policy engine selected more files for migration then the next bunch of 1000 files is loaded on the queue if 1000 files have finished processing.

Note, the above limitations regarding the maximum number of files on queue apply for manual migrations using the command: `eeadm migrate filelist`. The number of files in one file list or the total number of files in all file lists for concurrent `eeadm migrate` processes should be lower than the limits above.

## Strings substitutions to policy rules

The `mmapplypolicy` parameter `-M` allows to pass string substitutions to the policy rules before it is interpreted by the policy engine. This allows for a single policy file to be run with different rule parameters. For example, assume you have two file systems, each should be migrated to a different IBM Spectrum Archive EE pool. The IBM Spectrum Archive EE pool name is part of the EXTERNAL POOL rule definition. So instead of creating two different policy files, one can be created that looks like this:

```
RULE EXTERNAL POOL 'ltfs’ EXEC '/opt/ibm/ltfsee/bin/eeadm' OPTS
'EEPOOL'

RULE 'MigToLTFS' MIGRATE FROM POOL 'system' TO POOL 'ltfs' WHERE
((LOWER(NAME) LIKE '%.mp3') AND (KB_ALLOCATED > 0))
```

Notice the string 'EEPOOL' in the first rule, this can be substituted by the `mmapplypolicy` command using the -M options. The `mmapplypolicy` command for the two file systems (fs1 and fs2), where the data from fs1 is to be migrated to IBM Spectrum Archive EE pool 'Pool1@Lib1' and data from fs2 is to be migrated to pool 'Pool2@Lib1', looks like this:

```
# mmapplypolicy /fs1 -P mypolicy.txt -m 3 -n 1 -B 1000 -N
eenode1,eenode2 -single-instance -M “EEPOOL=-p pool1@lib1”

# mmapplypolicy /fs2 -P mypolicy.txt -m 3 -n 1 -B 1000 -N
eenode1,eenode2 -single-instance -M “EEPOOL=Pool2@Lib1”
```

This is just a simple example. If you think about generic policies for different file systems you may find more uses cases for string substitutions. See section String substitution in rules for more details how to implement and use this.

## Using existing file lists

The policy engine can use existing file lists instead of scanning the name space again. For example, if you have the path and file names for a list of files and want to process a subset of files that match certain rules, you can pass this file list to the policy engine and the policy engine will match the rules against these files only. This may be much faster than scanning an entire file system or fileset again.

There are different formats for the file list provides as input to the policy engine [3]. The most trivial format is with one path and file name per line, for example:

```
/path/file1
/path/file2
…
```

To run the policy engine with an input list (`existing_files.list`) the following command can be used:

```
# mmapplypolicy fsname -P policyfile -i existing_files.list
  -I defer -f ./my
```

The policy engine will not scan the file system (*fsname*) but use the existing file list passed with the parameter `-i existing_files.list` as input and determine the files that are migrated. The resulting file list is stored under the filename `./my.[..].list`.

For example, if you want select file names provided in the file list `existing_files.list` that are migrated, the following LIST policy can be used:

```
/* define exclude list */
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* macro definition */
define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )

/* list migrated files */
RULE EXTERNAL LIST 'migrated' EXEC ''
RULE 'm_files' LIST 'migrated' WHERE (is_migrated) AND NOT
(exclude_list)
```

When using this policy as policyfile with the mmapplypolicy command shown above then the resulting file list including the path and file names of migrated files will be named: `./my.migrated.list`.

If you have an existing file list created by the policy engine then you have to extract the path and file names. See section Extracting file names from file list for more information about extracting path and file names from file lists produced by the policy engine.

## Recall

Recalls can be triggered in two way:

- Transparent: upon file access a migrated file is automatically recalled from LTFS tape to the Spectrum Scale file system. If many migrated files are accessed at the same time then each file will be recalled individually with no optimization. Thus, it can take very long time.
- Tape optimized: using the command `eeadm recall`. This command requires a list of files to be recalled and recalls the files in an optimized manner by sorting the file by their tape-ID and position on tape. This method is much faster and less resource intensive.

Tape optimized recall can also be invoked with a policy. This is explained in section Recall policy

## Recall with policy

Recalls for a selected set of migrated files can be performed using MIGRATE policies. This is based on a customized interface script that wraps the `eeadm recall` command. As discussed in section External pools, the external pool rule defines an interface script that is invoked by the policy engine to process the file lists identified by MIGRATE rules. Instead of migrating files this customized interface script parses the parameters obtained from the mmapplypolicy command and processes this list of files using the `eeadm recall` command.  We use a standard migrate rule with a special external pool script.

The example below shows the policy that recalls all migrated files residing in folder `/mypath`. The external pool rule executes the customized interface script (`/path/ltfsee_recall.sh`) and passes the parameter `eelib1` which is the tape library name used for recall. The MIGRATE rule selects all files that are migrated:

```
/* define macros */
define(recall_dir, (PATH_NAME LIKE '%/mypath/%'))

/* define external pool */
RULE 'extpool' EXTERNAL POOL 'ltfs' EXEC
'/root/silo/recall/ltfsee_recall.sh' OPTS 'eelib1'

/* define migration rule */
RULE 'recall' MIGRATE FROM POOL 'ltfs' TO POOL 'system' WHERE
(XATTR('dmapi.IBMPMig') IS NULL) AND NOT
(XATTR('dmapi.IBMTPS') IS NULL) AND (recall_dir)
```

The customized interface script (`ltfsee_recall.sh`) essentially wraps the `eeadm recall` command. It parses the parameters obtained from the policy engine and invokes `eeadm recall` when the operation is migrate. In the example below the parameter `$polFile` is the list of files selected by the policy engine and the parameter `$option` is the library name given with the external pool rule (OPT clause) above.

```
# pseudo code for interface script that wraps eeadm recall

#global variables for this script
# set the default option in case $3 is not give, allows to
specify the library name
DEFOPTS=""
# define eeadm directory
EEADMDIR=/opt/ibm/ltfsee/bin
```

```
## Parse Arguments & execute
#$1 is the policy operation (list, migrate, etc)
op=$1
#$2 is the policy file name
polFile=$2
#$3 is the option given in the EXTERNAL LIST rule with OPTS '..'
should be the pool
shift 2
option=$*

## evaluate the operation passed by mmapplypolicy and act upon it
case $op in
  # there will always be a TEST call with $2 being the file
system path
  TEST )
  echo "INFO: TEST option received for $polFile"
  if [[ ! -z "$polFile" ]] then
    if [[ ! -d "$polFile" ]] then
      echo "WARNING: TEST directory $polFile does not exists."
    fi
  fi
  ;;
  RECALL | MIGRATE | LIST )
  # this is the actual migrate call where we call eeadm
premigrate
  echo "INFO: $op option received with file name $polFile and
options $option"
  #set option to default if not set
  if [[ -z $option ]] then
    if [[ -z $DEFOPTS ]] then
      echo "ERROR: Library name not specified in the external
pool rule."
       exit 1
    else
       option=$DEFOPTS
    fi
  fi

  echo "INFO: Start processing files $polFile with eeadm recall
from library $option"
  $LTFSEEDIR/eeadm recall -l $option $polFile
   rc=$?
   if (( rc != 0 )) then
     echo "WARNING: eeadm recall ended with return code $rc"
   fi
   ;;
  REDO )
  echo "INFO: REDO option received with file name $polFile and
options $option"
  ;;
  * )
  echo "WARNING: UNKNOWN operation ($op) received with file name
$polFile and options $option"
  ;;
esac

echo "$(date +"%Y-%b-%d %H:%M:%S") eeadm_recall ended"
exit 0
```

Note, the TEST operation must be implemented, because the policy engine expects a return code of zero for this in order to proceed. In this example we just check if the path name is valid. For more information about customized interface scripts refer to section Processing file lists.

Sample scripts and policies are stored on an IBM internal git-repository [5] in subdirectory /recall.

## Lists

The IBM Spectrum Scale policy engine can also be used to list files stored in an IBM Spectrum Scale file system and matching certain criteria. With such lists of files statistics can be derived, the effectiveness of migration policies can be checked, or files can be further processed. The advantage to use the IBM Spectrum Scale policy engine to identify files based on certain criteria is that the policy engine is much quicker than traversing the file system and evaluating each single file based on its attributes.

To run an EXTERNAL LIST policy at least two rules are required, like external pools (see section External pools). One rule defines an external list, the other defines the list condition, as shown below:

```
RULE EXTERNAL LIST 'list-name' EXEC 'program' OPTS option

RULE 'rule-name' LIST 'list-name' WHERE (conditions)
```

The first rule defines the external list name (list-name) and an interface script (program) to be executed for this list. The interface script can also be omitted by entering two single quotes (''). The OPTS string is optional and allows to pass more parameters to the interface script. The second rule describes the conditions files must match to be identified by this policy.

It is also possible to exclude files and / or directories within in a LIST rule using the EXCLUDE statement:

```
RULE EXTERNAL LIST 'list-name' EXEC 'program' OPTS option

RULE 'rule-name' LIST 'list-name' EXCLUDE WHERE (conditions)
```

With this set of rules files matching the condition are not listed and all other files are.

To run an EXTERNAL LIST policy the mmapplypolicy command can be used. Since the LIST policy does not perform migrations some options can be omitted.

```
# mmapplypolicy /filesystem -P policyfile -f ./fileprefix -I
defer
```

The -f parameter defines a file name prefix of the file list name. This file name prefix is appended by the string ‘list.external-list-name’. The external list name is given in the EXTERNAL LIST rule (in the example above this is ‘list-name’). Using the examples above the resulting file name would be

`./fileprefix.list.list-name`. The parameter "`-I defer`" instructs the policy engine to defer the execution phase, thus the policy engine will stop after the evaluation phase when files have been selected based on the criteria of the LIST rule. The result file of a list policy includes the file names matching the LIST rules. There is one file per line with some additional information. Find below an example of a list policy result file:

```
48900 1741777473 0   -- /filesystem/file1
```

The three first numbers are IBM Spectrum Scale internal numbers (inodenumber, inodegeneration, snapid). The file name is the 5th field in the result file. . See section Extracting file names from file list for more information about extracting path and file names from file lists produced by the policy engine.

For example, the following LIST policy lists all files that are migrated:

```
RULE EXTERNAL LIST 'migrated' EXEC ''

RULE 'm_files' LIST 'migrated' WHERE (MISC_ATTRIBUTES LIKE '%V%')
```

The first rule defines an external list named 'migrated'. The interface script is omitted because this policy should just produce an output file. To run this policy stored in file name `policyfile` use the command:

```
mmapplypolicy /filesystem -P policyfile -f ./file -I defer
```

The resulting file including all filenames for files that are migrated is named `file.list.migrated`. More examples of file lists can be found in section Generating file lists.

EXTERNAL LIST policies can also be used to process files identified by the policy engine. This requires the interface script to be implemented and specified in the EXTERNAL LIST rule. This interface script is automatically invoked when the policy is run. The interface script will receive 2 or more parameters as described below:

1. Parameter: string describing the operation. For an EXTERNAL LIST policy, the following strings are passed to the script: list and test
2. Parameter: name of the policy result file.
3. Parameter: the options given with  the first rule (optional) behind the OPTS clause

Based on this the interface script for a list policy can initiate actions with the files identified by the list policy. The result file of a LIST policy includes the file names of the files matching the rules. There is one file per line with some additional information. Find below an example of a list policy result file:

```
48900 1741777473 0   -- /mnt/filesystem/file1
```

The three first number are IBM Spectrum Scale internal numbers (inodenumber, inodegeneration, snapid). The file name is the 5th field in the result file.

One advantage processing selected files with a custom interface script is that the interface script is invoked by the policy engine automatically. The policy engine can invoke multiple instances of the interface script on multiple cluster nodes, all running in parallel. The policy engine provides each instance of the interface script a list of files to be processed. The node names, number of instances and number of files per list can be controlled with policy engine options (see section Options for mmapplypolicy).

More examples for processing file lists can be found in section Processing file lists.

Sample scripts and policies are stored on an IBM internal git-repository [5] in subdirectory /list.

## Showing file attributes

LIST rules can also show other file attributes using the SHOW clause. For example, to show the file size the following LILST rule can be used:

```
RULE 'm_files' LIST 'migrated' SHOW(varchar(FILE_SIZE) WHERE ..
```

The resulting output after executing the policy shows the file size in the 4th field:

```
48900 1741777473 0  4096 -- /mnt/filesystem/file1
```

Multiple file attributes can also be shown. For example, to show the file size and the tape ID for migrated files, the following rule can be used:

```
/* define is_resident */
define( is_resident,(MISC_ATTRIBUTES NOT LIKE '%M%') )

/* EXTERNAL LIST rule */
RULE EXTERNAL LIST 'mylist' EXEC ''

/* Selection rule that shows file size and tape for non-resident
files */
RULE 'size' LIST 'mylist'
  SHOW('size=' || varchar(FILE_SIZE) ||
       ' tape=' || xattr('dmapi.IBMTPS') )
FOR FILESET('test')
WHERE NOT(is_resident)
```

In this example the file size is prefixed by “size=” and the tape ID is prefixed by “tape=”. Here is an example of the output produced by the policy engine:

```
48900 1741777473 0  size=4096 tape=1 SLE033L6@33490ef5-be30-4f23-
b328-527b5e3109a4@0000013100730409 -- /mnt/filesystem/file1
```

Please note, the string displayed for the tape ID is long and represents the value of the `dmapi.IBMTPS` attribute.  See section Extracting file names from file list for more information about extracting path and file names from file lists produced by the policy engine.

## Macros

Macros allow defining more complex clauses and conditions to make rules better readable. Excluding files for migration - as shown in section Excluding files and directories - is one example for macros. The basic syntax for macros is the following

```
define( macro-name, (conditions & expressions) )
```

The macro-name is an arbitrary name which is used in a RULE later. The condition must follow the rule syntax. The following macro example defines the state of a migrated file:

```
define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )
```

The macro names can now be used as conditions in rules. For example, the following threshold-based policy uses the macro above and an exclude list:

```
/* define exclude_list macro */
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* define macro is_migrated */
define( is_migrated, (MISC_ATTRIBUTES LIKE '%V%') )

/* define external pool managed by LTFS */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p pool1@lib1’
SIZE 10485760

/* define migration rule using macros above */
RULE 'MigToTape' MIGRATE FROM POOL 'silver' THRESHOLD(90,70) TO
POOL 'ltfs' WHERE NOT (exlude_list) AND NOT (is_migrated)
```

For more example please see section Macro examples.

## Examples and use cases

In this section examples for IBM Spectrum Scale ILM rules and policies will be given. For additional background about the syntax and semantic refer to section ILM Policies and Rules.

Sample scripts and policies are published in a repository on GitHub [5].
For IBM Spectrum Archive Enterprise Edition users example policies are available in /opt/ibm/ltfsee/share/ of any server where IBM Spectrum Archive Enterprise Edition is installed.

## Placement policies

In this section some more examples for placement policies are given. Remember, placement can only be done on internal pools (see section Placement). The basic syntax of a placement rule is:

```
RULE 'rule-name' SET POOL 'pool-name' WHERE (condition)
```

Note, when file attributes are referenced in initial placement rules (WHERE clause), only the following attributes are valid: FILESET_NAME, GROUP_ID, NAME, and USER_ID.

Placement policies are applied to a file system using the mmchpolicy command:

```
# mmchpolicy filesystem -P policyfile
```

When applying a placement policy to a file system all relevant automated migration policies must be included.

### Placement by file name

The importance of files stored an IBM Spectrum Scale file system may vary. More important files may have to be stored on a pool comprised of faster and better protected disk, while all other files are stored on standard disk. The following policy places all excel-files (ending with .xls) stored in pool ‘system’ (rule name is important-rule) and all other files in pool ‘silver’ (rule name is default):

```
RULE 'important-rule' SET POOL 'system'
WHERE LOWER(NAME) LIKE '%.xls'

RULE 'default' SET POOL 'silver'
```

Remember to place the default placement rule at the end.

### Fileset placement

Filesets are logical partitions in a file system that can be managed separately, for example with quota and snapshots. With placement rules filesets can be aligned to file ‘system’ pools.

For example, there are 2 internal pools in the file system named ‘system’ and ‘silver’ (see figure 1 in section IBM Spectrum Scale file systems and pools). Pool ‘system’ is comprised of Flash storage while pool ‘silver’ is comprised of NL-SAS disk. The file system has two filesets, one fileset for storing home folders for NFS users (fileset name homefileset) and one for storing a database (dbfileset). The home folders should be stored on normal disk while the database should be stored on SSD. The following placement policy will store all data of the fileset “homefileset” in pool ‘silver’ and all data stored in the fileset “dbfileset” in pool ‘system’:

```
RULE 'homes' SET POOL 'silver' FOR FILESET ('homefileset')

RULE 'db' SET POOL 'system' FOR FILESET ('dbfileset')

RULE 'default' SET POOL 'silver'
```

Notice the default placement rule at the end. This is required to allow files to be stored in the file system outside of the filesets. If all files except of the files in the fileset ‘dbfileset’ shall be stored in pool ‘silver’ the placement policy above can be simplified:

```
RULE 'db' SET POOL 'system' FOR FILESET ('dbfileset')

RULE 'default' SET POOL 'silver'
```

### Prevent storing certain file types

Placement policies can also be (mis-)used to prevent creating files matching certain criteria. This can be achieved by creating a pool with only one NSD of type descOnly (see section IBM Spectrum Scale file systems and pools). Let’s name the pool ‘dummy’, the following placement policy shows how to place mp3-files on pool ‘silver’, mov-files on pool dummy and all other files on pool system:

```
RULE ‘do-not-place-mov’ SET POOL 'dummy'
WHERE LOWER (NAME) LIKE‘%.mov%’

RULE ‘mp3’ SET POOL 'silver'
WHERE LOWER (NAME) LIKE‘%.mp3%’

RULE 'default' SET POOL 'system'
```

Files ending with .mov are placed in pool dummy which does not contain any data NSD. Thus, these files cannot be stored in the file system. The user will receive an error when trying to store such file. If the user renames to .mp3 then the file will be stored in pool ‘silver’. Notice the order of the rules, the default rule must be at the end.

## Macro examples

As explained in section Macros, macros allow to make rules easier to read by pre-defining more complex conditions and expression.

The following macro defines HSM states a file can have such as migrated, pre-migrated and resident:

```
define( is_premigrated,(MISC_ATTRIBUTES LIKE '%M%' AND
MISC_ATTRIBUTES NOT LIKE '%V%') )

define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )
define( is_resident,(MISC_ATTRIBUTES NOT LIKE '%M%') )
```

The following macro defines the age of files in days based on access time and modification time:

```
/* macro to define access age */
define( access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME))
)

/* macro to define modification age */
define( mod_age,(DAYS(CURRENT_TIMESTAMP) -
DAYS(MODIFICATION_TIME)) )
```

For larger files it might be valuable to evaluate the file size in Megabytes, instead of Kilobytes. The following macro defines the file size in Megabyte

```
/* macro to define file size in MB */
define(  mb_allocated, (INTEGER(FILE_SIZE / 1024000)) )
```

With the above macros the WEIGHT expression can be made more sophisticated as the following macro shows:

```
define(

    weight_expression,

    (CASE

        /*=== The file is very young, the ranking is very low
===*/
       WHEN access_age <= 1  THEN 0

        /*=== The file is very small, the ranking is low ===*/
       WHEN mb_allocated < 1 THEN access_age

        /*=== The file is premigrated and large and old enough,

              the ranking is very high ===*/
       WHEN is_premigrated   THEN mb_allocated * access_age * 10

        /*=== The file is resident and large and old enough,

              the ranking is standard ===*/

        ELSE  mb_allocated * access_age

    END)
)
```

Please notice that this macro named `weight_expression` uses the macros `access_age`, `mb_allocated` and `is_premigrated`, that are explained above.

A policy using the above macros could look like the following:

```
/* define external pool managed by LTFS */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1'
SIZE 10485760

/* define migration rule using macros above */
RULE 'MigToTape' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(weight_expression) TO POOL 'ltfs' WHERE NOT (exlude_list)
AND NOT (is_migrated)
```

When using this policy all above macros must be embedded in the policy file.

## Threshold based migration

As explained in sections IBM Spectrum Scale Policy engine and Automated migration, threshold based rules are typically used with automated migrations. Automated migration policies are activated for a file system using the mmchpolicy command:

```
# mmchpolicy filesystem -P policyfile
```

When applying a threshold policy to a file system all relevant placement policies must be included in the policy file.

In addition, automated migration requires the configuration of a callback (see section Migration Callback). The callback is invoked when the file system pool reaches the high occupancy percentage configured with the threshold-based rule and starts a callback script. The callback script then invokes the policy engine.

### Three pool migration with callback

Imagine there are three pools, just like in figure 1 (section IBM Spectrum Scale file systems and pools). The automated policy should do the following:

- The system pool shall be migrated to 'silver' pool when occupied 80%.
- The 'silver' pool shall be migrated to 'ltfs' at 90%. LTFS stores the data on a tape pool named "Pool1@Lib1".
- In addition, all files ending with jpg, png, mp3, mp4, mpeg and mov should be stored on the 'silver' pool.

The associated automated migration policy looks like this:

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* macro to define access age */
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))

/* here comes the migration rule for system to silver */
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(access_age) TO POOL 'silver‘

/* here comes the migration rules for silver to ltfs*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm' OPTS
‘-p pool1@lib1’

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(access_age) TO POOL 'ltfs'

/* here comes the placement rules for silver */
RULE ‘media-rule’ SET POOL 'silver' WHERE
  LOWER (NAME) LIKE '%.mp3%' OR LOWER (NAME) LIKE '%.mp4%' OR
  LOWER (NAME) LIKE '%.jpg%' OR LOWER (NAME) LIKE '%.png%' OR
  LOWER (NAME) LIKE '%.mpeg%' OR LOWER (NAME) LIKE '%.mov%'
RULE 'default' SET POOL 'system'
```

To activate the automated policy the mmchpolicy command is used:

```
# mmchpolicy filesystem -P policyfile
```

In addition, a callback must be configured. The challenge here is that we need to control the migration on internal and external pools and the parameters for the policy engine are different (see section Running a policy). One solution is to adjust the callback script and let it decide upon the parameters for the policy engine based the pool name. The pool name can be passed to the callback script.

To configure the callback, use the following command:

```
mmaddcallback MIGRATION --command
/root/silo/callback/mystartpolicy
--event lowDiskSpace,noDiskSpace
--parms "%eventName %fsName %storagePool"
```

This callback is named MIGRATION and is triggered by the events “lowDiskSpace” and “noDiskSpace”. When one of these events occurs then IBM Spectrum Scale automatically invokes the callback script /root/silo/callback/mystartpolicy with the parameters eventName, fsname and storagePool. The script

/root/silo/callback/mystartpolicy invokes the mmapplypolicy command with the file system name and the mmapplypolicy parameters derived from the storage pool name within the script (see below). The command mmapplypolicy will run the above policy applied with mmchpolicy.

The script /root/silo/callback/mystartpolicy is an exact copy of the standard script /usr/lpp/mmfs/bin/mmstartpolicy with the following additional changes:

The first change is in the area where the parameters are parsed and assigned. The parameter %storagePool is arg3 which is assigned to the variable pool. In addition, the rest of the options must shift 3 instead of 2 as shown in blue below:

```
eventType=$arg1
device=$arg2
deviceName=${device##+(/)dev+(/)}  # Name stripped of /dev/
prefix.

#New get arg3 to be the pool
pool=$arg3
#Changed: shift 3 instead of 2
#shift 2
shift 3

options=$@
```

The next change is at the beginning of the main section. Here the variable "options" - which contains the mmapplypolicy options - is set in accordance to the pool variable. In this example when the pool is 'system', no special options are set because the 'system' pool is migrated to the 'silver' pool which is also an internal pool. If the pool name is 'silver' then special mmapplypolicy parameters are assigned to the options variable in order to facilitate the requirements for IBM Spectrum Archive EE. The new lines in the callback script following the main body of code comment is shown below in blue:

```
# main body of code

print -- "$(date): $mmcmd: $device generated event $eventType on
node $ourNodeName"

#NEW depending on the pool set the right options
print -- "$(date): Pool name triggering the policy is $pool"
if [[ "$pool" != "system" ]]
then
  print -- "Info: setting mmapplypolicy options=$options -N node1
-m 3 -n 1 --single-instance"
  options=$options" -N <eeNodes> -s <tempDir> -m 3 -n 1 --single-
instance"
else
  print -- "Info: setting mmapplypolicy options=$options"
  options=$options
fi
```

Note, the options being set in this example (-N <eeNodes> -s <tempDir>-m 3 -n 1 --single-instance) need to be adjusted according to the actual environment. Especially the parameter -N needs to specify  all relevant IBM Spectrum Archive EE nodes.

With this adjusted callback script, the migration from pool ‘system’ to pool ‘silver’ will be invoked with different parameters than the migration from pool ‘silver’ to tape managed by IBM Spectrum Archive EE.

## Prioritized migration

Imagine the file system subject for space management has different sub-directories or filesets. Each sub-directory has different priorities for migrations, for example:

- Directory1: files are migrated after 30 days
- Directory2: files are migrated after 60 days
- Directory3: files should never be migrated

One of the fundamental guidelines is that the active migration policy should be simple and functional. Implementing an active migration policy according to the priorities above can cause file system to run out of space. Just imagine that suddenly many large files are stored in Directory3. The priority for Directory3 demands no migration. Consequently, the file system might fill up with files in Directory3.

An active migration policy must be implemented to free up space in the file system pools in any case. The priorities for different directories can be honored with different WEIGHT. For example, the weight for Directory1 can be highest and the weight for Directory3 can lowest, resulting in files from Directory1 being selected first for migration and files from Directory3. The associated WEIGHT statement looks like this:

```
/* weight expression preferring certain paths*/
/* can be used in threshold-based migration to prefer certain
dirs */
define(
    directory_weight,
    (CASE
        /*=== dir1 has high priority ===*/
           WHEN PATH_NAME like '%/Directory1/%'  THEN 4
        /*=== dir2 has medium priority ===*/
           WHEN PATH_NAME like '%/Directory2/%'  THEN 3
        /*=== dir3 has low priority ===*/
           WHEN PATH_NAME like '%/Directory3/%'  THEN 2
    END)
)
```

This WEIGHT definition can now be used in an active migration policy that migrates files from pool ‘silver’ to ‘ltfs’ if the ‘silver’ pool is 90% full.

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* here comes the migration rules for silver to ltfs*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p Pool1@Lib1’
SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(directory_weight) TO POOL 'ltfs'

/* here come the placement rules … */
```

With this policy it can be assured that files from Directory1 are migrated first, then Directory2 and Directory3 is latest.

## Manual or scheduled Migration

This section gives some more examples for scheduled policies (see section Scheduled migration) for more background). Scheduled policies are typically executed using the command mmapplypolicy (see section Running a policy). The parameters of the mmapplypolicy command depend on the type of pools. When migrating to an external pool consider the following command:

```
# mmapplypolicy filesystem -P mypolicy.txt -m 3 -n 1 -B 1000 -N
eenode1,eenode2 --single-instance
```

Note, the above command assumes there are two IBM Spectrum Archive EE nodes named eenode1 and eenode2. Make sure to name the IBM Spectrum Archive EE node properly.

For simplicity the samples below describe scheduled migration policies among internal pools. For external pool migration the external pool definition must be provided in addition (see section External pools).

### Migrating multiple pools

As shown in figure 1 (see section IBM Spectrum Scale file systems and pools) there can be multiple storage pools in a file system. Scheduled migrations from one pool to another need to be orchestrated to achieve the goal for migrations.

Let’s assume the ‘system’ pool is configured with smaller capacities and files stored in this pool should be migrated after 30 days to the ‘silver’ pool. The ‘silver’ pool is larger and files and should be migrated to LTFS after 60 days. The age of the files in both cases is based on access time. There are two scheduled policies to be configured for this scenario.

The following example shows the rules for migrating files accessed 30 days ago or longer from ‘system’ to ‘silver’.

```
/* rule for migrating files older 30 days from system to silver*/
RULE 'manMigSystem' MIGRATE FROM POOL 'system' TO POOL 'silver'
WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 30)
```

The next example shows the rules from migrating files accessed 60 days ago or longer ‘silver’ to ‘ltfs’:

```
/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define access_age */
define( access_age,
(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)) )

/* migrate to ltfs */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p pool1@lib1’
SIZE 10485760

RULE 'manMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'ltfs'
WHERE (access_age > 60)
```

This policy includes an EXCLUDE rule and an additional macro for evaluating the access time.

Both scheduled policies are invoked with the separate `mmapplypolicy` command, whereby the parameters given to this command may differ (see section Running a policy).

I do not recommend running scheduled policies for different pools within one file system at the same time because both policies will cause I/O workloads on the internal pools. Typically, the policy for migrating from ‘system’ to ‘silver’ runs first and the policy for migrating from ‘silver’ to ‘ltfs’ would runs afterwards.

In the next subsection there are more examples for rules to control schedule migrations.

## Grouping multiple pools

With GROUP pools it is possible to combine multiple pools under one virtual pool and balance the data among these pools. Imagine there is a ‘system’ pool comprised of flash storage and a ‘silver’ pool comprised of NL-SAS disk. These two pools can be represented in a GROUP pool and files can be balanced across these pools based on file heat.

The GROUP pool definition looks like this:

```
RULE 'PoolGroup' GROUP POOL 'TIERS' IS
'system' LIMIT(80)
THEN 'silver'
```

This rule defines a GROUP pool named ‘TIERS’ that consists of ‘system’ pool and ‘silver’ pool. The ‘system’ pools occupation limit is 80%. There can also be more than two pools in a GROUP. The order of the pool names in the GROUP is important, the files are balanced based on this order, starting with the first pool.

To balance the files across these pools the following rule can be used:

```
RULE 'Rebalance' MIGRATE FROM POOL 'TIERS' TO POOL 'TIERS'
WEIGHT(FILE_HEAT)
```

This MIGRATE rule essentially defines to rebalance the files between the pools in the group ‘TIERS’ based on FILE_HEAT.

When combining these two policies then the hottest files will be placed in ‘system’ pool until this pool reaches 80%. The rest of the files is stored in ‘silver’ pool. Files that become hot in the ‘silver’ pool may be migrated upward to ‘system’ pool, while colder files are migrated downward from ‘system’ pool to ‘silver’ pool.

A GROUP policy should be run as scheduled policy because it is not easily possible to define thresholds for a group of pools (see section Running a policy).

Note: Rebalancing does not take place in real time, only when the GROUP policy is run using the `mmapplypolicy` command. This is not like an easy-tier solution, rather a solution to automatically rebalance files between tiers. The flexibility to select portions of files for migration is limited because the balancing takes place across all pools defined in the group.

It is also possible to include external pools in the group. The requires an external pool definition as shown in the example below.

```
/* Exclude rule */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define access_age */
define( access_age,
(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)) )

/* define ltfs pool*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
```

```
OPTS ‘-p pool1@lib1’
SIZE 10485760

/* define pool group with three pools */
RULE 'PoolGroup' GROUP POOL 'TIERS' IS
'system' LIMIT(80)
THEN 'silver' LIMIT (80)
THEN ‘ltfs’

/* define migration rule for rebalancing */
RULE 'Rebalance' MIGRATE FROM POOL 'TIERS' TO POOL 'TIERS'
WEIGHT(access_age)
```

In the example above the occupation for pools ‘system’ and ‘silver’ is limited to 80%. The balancing WEIGHT is based on the access age because FILE_HEAT may not be appropriate for files stored on tape.

### Migration based on file size

The following migration policy migrates files larger than 10 MB from ‘system’ to ‘silver’ pool:

```
/* define mb_allocated */
define(  mb_allocated, (INTEGER(KB_ALLOCATED / 1024)) )

/* migration based on file size */
RULE 'sizeMig' MIGRATE FROM POOL 'system' TO POOL 'silver'
WHERE (mb_allocated > 10)
```

### Migration based on modification age

The following migration policy migrates files modified longer than 30 days ago from ‘system’ to ‘silver’ pool:

```
/* define macro for modification time */
define( mod_age,
(DAYS(CURRENT_TIMESTAMP) - DAYS(MODIFICATION_TIME)) )

/* migration based on modification age*/
RULE 'modAge' MIGRATE FROM POOL 'system' TO POOL 'silver'
WHERE (mod_age > 30)
```

### Migration based on fileset name

The following migration policy migrates all files stored in fileset “homefileset” and larger than 10 MB and accessed 30 days ago or longer from ‘system’ pool to ‘ltfs’. Such policy would typically be used as schedule policy and not as automated policy:

```
/* migration for fileset */
```

```
RULE 'homeMigLtfs' MIGRATE FROM POOL 'system' TO POOL 'silver'
FOR FILESET ('homefileset') WHERE (access_age > 30) AND
(mb_allocated > 10)
```

To make this policy effectively working ensure that files stored in fileset “homefileset” are stored in the ‘silver’ pool. This can be accommodated with a placement policy, such as the following (also see section Placement policies):

```
RULE 'homes' SET POOL 'silver' FOR FILESET ('homefileset')
```

## Pre-migration and migration

As explained in section Pre-migration the %high threshold decides whether files are to be selected and the %low in combination with the %premig thresholds define how many file capacity are pre-migrated. The %high threshold makes it difficult to control pre-migration because it triggers the policy engine to run. If the occupancy of the pool has not met the %high value pre-migration will not be done. Setting %high to a low value in conjunction with automated policies is not appropriate because it may trigger constant policy runs. Therefore, it is recommended to combine scheduled and automated policies for pre-migration.

Let’s assume all data from the ‘silver’ pool should be pre-migrated on a regular basis if the ‘silver’ pool occupancy is between 50% and 80 %. When the occupancy of the ‘silver’ pool is above 80 % files should be migrated, until the ‘silver’ pool occupancy drops to 50 %. This migration will be very quick because most files are pre-migrated already. The following examples show a scheduled and automated migration policy.

The scheduled policy will pre-migrate files from pool ‘silver’ to an external pool managed by IBM Spectrum Archive EE if the ‘silver’ pool occupancy is between 50 and 80 %:

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define macro is_migrated */
define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )

/* define macro access_age */
define( access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME))
)

/* Define LTFS as external pool */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1' SIZE 10485760

/* it premigrates if pool occ < 30% */
/* DO NOT USE THIS AS AUTOMATED MIGRATION POLICY */
RULE 'premig3' MIGRATE FROM POOL 'silver' THRESHOLD (0,80,50)
WEIGHT (access_age) TO POOL 'ltfs' WHERE NOT (is_migrated)
```

Setting the %high value to 0 within the THRESHOLD statement ensures that the policy will run whatsoever. It will however only start processing files if the pool occupancy is between 50 - 80 %.

This scheduled policy is invoked with the mmapplypolicy command which can be scheduled with the cron-daemon:

```
# mmapplypolicy filesystem -P mypolicy.txt -m 3 -n 1 -B 1000 -N
eenode1,eenode2 --single-instance
```

The automated migration policy for the ‘silver’ pool will be triggered if the occupancy gets above 80% and it will migrate files until it reaches an occupancy of 50 %. This policy defines a %high value of 80% and no %premig value because pre-migration is done by the scheduled policy:

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define macro is_migrated */
define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )

/* define macro access_age */
define( access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME))
)

/* Define LTFS as external pool */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p pool1@lib1' SIZE 10485760

/* it migrate at 80% */

RULE 'premig4' MIGRATE FROM POOL 'silver' THRESHOLD (80,50)
WEIGHT (access_age) TO POOL 'ltfs' WHERE NOT (is_migrated)
```

This automated policy will run quickly because the scheduled pre-migration policy shown above makes sure that 30 % of the files are already pre-migrated. The migration does not have to transfer selected files, it will just create the stub-files.

Both policies use the statement “`WEIGHT(access_age)`” which ensures that files are selected for migration and pre-migration based on their usage.

Pre-migration can also be done with automated migration. The following rule will first migrate files if the occupancy of ‘silver’ pool is at 80% until it drops below 50% and then pre-migrate files equal to 20% of the total pool capacity (%low - %premig):

```
RULE 'premig5' MIGRATE FROM POOL 'silver' THRESHOLD (80,50,30)
WEIGHT (access_age) TO POOL 'ltfs' WHERE NOT (is_migrated)
```

The first run of this policy will first migrate and then pre-migrate files according to the threshold. The second run of this policy will already benefit from the pre-migrated files, if these have not changed.

The disadvantage of this automated pre-migration policy is that it is only triggered if the 'silver' pool occupation is above 80%.

An alternate approach for pre-migration using migrate rules without threshold is explained in section Using customized external script). This method relies on a customized script that wraps the `eeadm premigrate` command.

## Using customized external script

Another method to run pre-migration instead of migration is based on a customized interface script that wraps the `eeadm premigrate` command. As discussed in section External pools, the external pool rule defines an interface script that is invoked by the policy engine to process the file lists identified by MIGRATE rules. Instead of migrating files this customized interface script parses the parameters obtained from the policy engine and processes this list of files using the `eeadm premigrate` command.  This allows us to use a standard MIGRATE rule with a special external pool script.

The example below shows the policy that pre-migrates all files that are not pre-migrated and are larger than zero bytes. The external pool rule executes the customized interface script (`/path/ltfsee_premig.sh`). The MIGRATE rule selects all files that are not yet pre-migrated and not empty:

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define macros */
define( is_premigrated,
(MISC_ATTRIBUTES LIKE '%M%' AND MISC_ATTRIBUTES NOT LIKE '%V%') )
define( is_empty,(KB_ALLOCATED=0) )

/* Define LTFS as external pool with customized interface script
*/
RULE EXTERNAL POOL 'ltfsPremig'
EXEC '/path/ltfsee_premig.sh'
OPTS '-p test@eelib1' SIZE 10485760

/* here comes the migration rule that selects all files that are
not pre-migrated and not empty*/
```

```
RULE 'MigToExt' MIGRATE FROM POOL 'system' TO POOL 'ltfsPremig'
WHERE (NOT (is_empty) AND NOT (is_premigrated))
```

The customized interface script essentially wraps the `eeadm premigrate` command. It parses the parameters obtained from the policy engine and invokes `eeadm premigrate` when the operation is migrate. In the example below the parameter `$polFile` is the list of files selected by the policy engine and the parameter `$option` is the target pool for migration given with the external pool rule (OPT clause) above.

```
# pseudo code for interface script that wraps eeadm premigrate

#global variables for this script
# set the default option in case $3 is not given
DEFOPTS=""
# define eeadm directory
LTFSEEDIR=/opt/ibm/ltfsee/bin

### Parse Arguments & execute
#$1 is the policy operation (list, migrate, etc)
op=$1
#$2 is the policy file name
polFile=$2
#$3 is the option given in the EXTERNAL LIST rule with OPTS '..'
should be the pool
shift 2
option=$*

## evaluate the operation passed by mmapplypolicy and act upon it
case $op in
  # there will always be a TEST call with $2 being the file
system path
  TEST )
  echo "INFO: TEST option received for $polFile"
  if [[ ! -z "$polFile" ]] then
    if [[ ! -d "$polFile" ]] then
      echo "WARNING: TEST directory $polFile does not exists."
    fi
  fi
  ;;
  MIGRATE )
  # this is the actual migrate call where we call eeadm
premigrate
  echo "INFO: MIGRATE option received with file name $polFile and
options $option"
  #set option to default if not set
  if [[ -z $option ]] then
    if [[ -z $DEFOPTS ]] then
      echo "ERROR: Pool name not specified in the external pool
rule."
      exit 1
    else
      option=$DEFOPTS
    fi
  fi

  echo "INFO: Start processing files $polFile with eeadm
premigrate to pool $option"
  $LTFSEEDIR/eeadm premigrate -s $polFile $option
  rc=$?
  if (( rc != 0 )) then
    echo "WARNING: eeadm premigrated ended with return code $rc"
  fi
  ;;
  * )
  echo "WARNING: UNKNOWN operation ($op) received with file name
$polFile and options $option"
  ;;
esac

echo "$(date +"%Y-%b-%d %H:%M:%S") eeadm_premig ended"
echo
exit 0
```

Note, the TEST operation must be implemented, because the policy engine expects a return code of zero for this in order to proceed. In this example we just check if the path name is valid. For more information about customized interface scripts refer to section Processing file lists.

Sample scripts and policies are stored on the GitHub repository [5] in subdirectory /premigrate.

## String substitution in rules

As explained in section Running a policy, it is possible to pass strings from the `mmapplypolicy` command to the rule before this is being evaluated. The mmapplypolicy command parameter: `-M “STRING=VALUE”` is used for this. STRING is a string in a rule which is substituted by the value. This mechanism allows to create generic policies which can be applied to any file system using a custom wrapper script.

The wrapper script takes the following parameters as input.

- File system name
- Name of the policy file
- Name of the tape pool

These parameters can be given from the command line or within a schedule invoking the wrapper script. The wrapper script named ‘myscript’ may be invoked like this:

```
# myscript fsname policyfile tapepool
```

The parameter *fsname* is the name of the file system or directory to run the policy.
The parameter *policyfile* is the name of the generic policy file.
The parameter *tapepool* is the name of the tape pool given with the syntax Poolname@Libraryname.

The wrapper script uses these parameters to invoke the policy engine with substitution variables:

```
# mmapplypolicy $fsname -P $policyfile -m 3 -n 1 -B 1000 -N
eenode1,eenode2 -single-instance -M “POOLVAR=$tapepool” -M
“FILESYSVAR=$fsname”%””
```

The parameters *$fsname, $policyfile* and *$tapepool* are input to the wrapper script and these are now used to run a generic policy for a given file system. The strings ‘POOLVAR’ and ‘FILESYSVAR’ are passed into the policy.

A simple generic policy that uses these substation variables to migrate all files from pool ‘silver’ pool ‘ltfs’ looks like this:

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* Define LTFS as external pool */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS 'POOLVAR' SIZE 10485760

/* here comes the migration rule whereby the FILESYSTEM is given
by the runpolicy script*/
RULE 'MigToTape' MIGRATE FROM POOL 'silver' TO POOL 'ltfs' WHERE
(PATH_NAME LIKE 'FILESYSVAR') AND (KB_ALLOCATED > 0)
```

The string ‘`POOLVAR`’ is being substituted with the name of the tape pool and the string ‘`FILESYSVAR`’ is substituted with the file system directory name subject for this policy. Note, the condition statement (PATH_NAME LIKE 'FILESYSVAR') is used to demonstrate using string substitution, it might not be necessary from a rule perspective.

## Quota based migration for filesets

The trigger for an automated migration is based on the occupancy of a storage pool as discussed in section Automated migration. In this section we discuss an approach to trigger the migration for a fileset when a fileset quota limit (soft or hard) has been met.

For this approach we use the following techniques:

1. An EXTERNAL LIST policy that allows to identify files according to the quota limits using the clause `THRESHOLD (resourceclass)`. The resource class can be `FILESET_QUOTAS`

for hard quota limits and FILESET_QUOTA_SOFT for soft quota limits. Files are identified if the occupation in the fileset is above the quota limit. The LIST policy identifies a set of files that brings the occupation in the fileset down to a low threshold. The LIST policy stores the list of files in the file system (see Quota LIST policy)

2. A MIGRATE policy takes the list of files generated by the LIST policy, evaluates these against the migration rule and migrates selected files (see Quota Migration policy)
3. A callback script that is invoked when the event "softQuotaExceeded" is raised for a fileset. This callback script (program) runs the list policy of step 1 to identify files according to the quota limits and invokes the migration policy of step 2 with the files being identified. The callback script is the core of this approach (see Quota callback script).
4. Last but not least the callback is defined that receives the "softQuotaExceeded" and runs the callback script passing on certain parameters including the file system and fileset name that triggered this even Quota callback).

In order to run the migration to an external pool the MIGRATE policy needs to be started on a node which has the migration software (e.g. Spectrum Archive Enterprise Edition) installed. The event "softQuotaExceeded" is raised only on the file system manager. If the file system manager node is not identical with the IBM Spectrum Archive EE node then some commands must be propagated to the appropriate nodes within the callback script (see Further considerations).

An alternative approach is to invoke the migration from the EXTERNAL LIST policy. This however requires adjusting the interface script for migration or to create wrapper-script. This approach is not discussed in here.

Sample scripts and policies are stored on in the GitHub repository [5] in subdirectory /quota-migration.

## Quota LIST policy

The list policy identifies files that are above a high threshold relative to the quota limit and store these in a file.

In the policy example below fileset soft quota (THRESHOLD 'FILESET_QUOTA_SOFT') in the first rule is checked against the THRESHOLD(90,80) in the rule. If the high soft quota limit of 90 % is met, then the second rule selects files until the low soft quota limit of 80% is met. Please note that the "FOR FILESET" clause expects a fileset to be passed to this policy via the mmapplypolicy command:

```
/* define macros */
define(is_empty, (KB_ALLOCATED=0))
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))

/* check softquota against threshold and select files */
RULE EXTERNAL LIST 'softquota' THRESHOLD 'FILESET_QUOTA_SOFT'

RULE 'fsetquota' LIST 'softquota' THRESHOLD(90,80) WEIGHT(access_age)
FOR FILESET ('FSETNAME') WHERE NOT (is_empty)
```

This LIST policy is run by the callback script explained in section Quota callback script. This LIST policy must be stored in file referenced by the variable $listpol in the callback script. The callback script runs this policy with the following command:

```
# mmapplypolicy $fsName -P $listpol -f $outfile -M FSETNAME=$fsetName
--single-instance -I defer
```

This `mmapplypolicy` command produces an output file that is denoted by the `-f $outfile` parameter. In the callback script example above it is named: `qFiles-fsname-fsetname.list.softquota`, whereby the *fsname* is the name of the filesystem and *fsetname* is the name of the fileset for which the event was triggered. This file list needs to be adjusted by extracting the path and filenames of the files using the following command in the callback script above:

```
# cut -d' ' -f7-20 $outfile.list.softquota > $outfile.list
```

Also note that we use parameter substitution: the parameter FSETNAME in the LIST rule is replaced by the actual fileset name given by the `mmapplypolicy` command with the parameter `-M FSETNAME=$fsetName`

Next step is to invoke the MIGRATE policy with the file that has been generated by the LIST policy.

## Quota Migration policy

The migration policy can be invoked with the pre-selected list of files (`$outfile.list`) provided by the list policy (see section Quota LIST policy). In this case it will not scan the file system metadata but use the files in the file list as input for candidates to be chosen for migration.

The migration policy defines some macros and has the well-known EXCLUDE rule first. This is important since we cannot efficiently exclude files with the LIST policy. In the EXCLUDE rule we should exclude all files that should never be migrated. Subsequently we have the migration rule that essentially migrates from pool ‘system’ to pool ‘ltfs’ if the file has data allocated. This automatically excludes migrated files for migration assuming the stub size is 0.

```
/* define macros */
define(is_empty,(KB_ALLOCATED=0))
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))

/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* migration policy */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS ‘-p pool1@lib1’
SIZE 10485760

RULE 'quotaMig' MIGRATE FROM POOL 'system' WEIGHT(access_age)
TO POOL 'ltfs' FOR FILESET ('FSETNAME')
WHERE NOT (is_empty)
```

This MIGRATE policy is run by the following command in the callback script below (see section Quota callback script) and assumes that the LIST policy is stored in file referenced by the parameter `$migpol`.

```
# mmapplypolicy $fsName -P $migpol -N $eenodes --single-instance -M
FSETNAME=$fsetName -i $outfile.list
```

To run the MIGRATE policy the node names or node class of the IBM Spectrum Scale cluster nodes running the IBM Spectrum Archive EE software must be specified (parameter `-N $eenodes`)

Note, that we are again use parameter substitution for the parameter FSETNAME.

Once we have tested the LIST and MIGRATE policies we add them to the callback script and test it.

## Quota callback script

The callback script is invoked by the callback (see section Quota callback) when the event "softQuotaExceeded" has triggered. The callback passes three parameters to the callback script: %eventName as $1, %fsname as $2 and %filesetName as $3).

The callback script first runs the LIST policy that produces an output file including file names to migrated (see section Quota LIST policy). The LIST policy file must be stored in a location denoted by parameter `$listpol` in the script. Afterwards the callback script runs the MIGRATE policy that migrates the files provided in the output file of the LIST policy (see Quota Migration policy). The MIGRATE policy file must be stored in a location denoted by parameter `$migpol` in the script. Find below a pseudo code example for this script:

```
# callback script to manage the softQuotaExceeded event

# define the  static variables
workDir=/gpfs/.work
outfile=$workDir"/qFiles"
listpol=$workDir"/list.pol"
migpol=$workDir"/mig.pol"
logF=$workDir"/callback-quota.log"
eenodes=”eenode1,eenode2”

# assign parameters given to the scipt
evName=$1
fsName=$2
fsetName=$3

# check parameters
...

# set the name of the output file to $outfiles-fsname-fsetname
outfile=$outfile"-"$fsName"-"$fsetName

# run the list policy and store the result in $outfile
mmapplypolicy $fsName -P $listpol -f $outfile -M FSETNAME=$fsetName
--single-instance -I defer  >> $logF 2>&1

# the result file is named $outfile.list.softquota
if [[ -a $outfile.list.softquota ]];
```

```
  # adjusting the output file by extracting the path and filenames
  cut -d' ' -f7-20 $outfile.list.softquota > $outfile.list
  mmapplypolicy $fsName -P $migpol -N $eenodes --single-instance
  -M FSETNAME=$fsetName -i $outfile.list >> $logF 2>&1
fi

# examine return codes and send event notification when required
exit 0
```

The callback script and the policy files must be installed on all cluster nodes with manager role. This is because the event is only triggered on the file system manager that runs on a node with manager role. Once the callback script works we configure the callback.

### Quota callback

Once you have tested this script and the policies, you can create a callback. The callback is triggered (invoked) by the event "softQuotaExceeded" and executes the script "`/gpfs/.work/callback-quota.sh`" with the parameters: %eventName as $1, %fsname as $2 and %filesetName as $3. To create the callback run the following command:

```
# mmaddcallback SOFTQUOTA-MIGRATION --command /gpfs/.work/callback-
quota.sh --event softQuotaExceeded --parms "%eventName %fsName
%filesetName"
```

### Further considerations

The event "softQuotaExceeded" is only triggered once per fileset when soft quota limit is exceeded. It expects the space consumption to decrease under the quota limits. If this is the case and after a while the quota limit is reached again then this event is triggered again. Otherwise, if the space consumption does not decrease then the event might not be triggered again. Therefore, it is important to make sure that the callback script works 100 % and that it alerts the admin if not.

In order to re-trigger the event the soft quota limits can be increased, or files can be move out and back in again. Some delay between moving files out of the files and in should be planned (5 - 10 min).

You must enable quota on filesets and set quota limits. Using the GUI for this makes it easier.

Consider placing the temporary file generated by mmapplypolicy in a directory with enough space. Use the parameter -s with the mmapplypolicy command for this.

## Generating file lists

In this section some examples for lists are given. To generate these lists the `mmapplypolicy` command is used in conjunction with the policy file.

## Listing files by HSM state

The following LIST policy identifies files based on the HSM state and writes these lists of files to appropriate output files:

```
/* listing files by state */

/* define exclude list */
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* macro definition */
define( is_premigrated,(MISC_ATTRIBUTES LIKE '%M%' AND
(MISC_ATTRIBUTES NOT LIKE '%V%') )
define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )
define( is_resident,(MISC_ATTRIBUTES NOT LIKE '%M%') )


/* list migrated files */
RULE EXTERNAL LIST 'migrated' EXEC ''
RULE 'm_files' LIST 'migrated' WHERE (is_migrated) AND NOT
(exclude_list)

/* list pre-migrated files */
RULE EXTERNAL LIST 'premigrated' EXEC ''
RULE 'p_files' LIST 'premigrated' WHERE (is_premigrated) AND NOT
(exclude_list)

/* list resident files */
RULE EXTERNAL LIST 'resident' EXEC ''
RULE 'r_files' LIST 'resident' WHERE (is_resident) AND NOT
(exclude_list)
```

To run this policy, use the command:

```
# mmapplypolicy filesystem -P policyfile -f ./file -I defer
```

This will create three output files named ./file.list.migrated, ./file.list.premigrated and ./file.list.resident.

## Listing tape IDs for files

The following example shows how to list the tape ID for all pre-migrated and migrated files. We leverage the SHOW clause to show an extended attribute '`dmapi.IBMTPS`' for each file that includes the tape ID and tape pool name.

The following list policy will list migrated and pre-migrated files, including the migration state and the tape ID:

```
/* define exclude list */
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* define file states */
define( is_premigrated,(MISC_ATTRIBUTES LIKE '%M%' AND
MISC_ATTRIBUTES NOT LIKE '%V%') )
define( is_migrated,(MISC_ATTRIBUTES LIKE '%V%') )

/* define external lists */
RULE EXTERNAL LIST 'migrated' EXEC ''
RULE EXTERNAL LIST 'premigrated' EXEC ''

/* show migrated files and tape */
RULE 'MIGRATED' LIST 'migrated' SHOW('migrated ' ||
xattr('dmapi.IBMTPS')) WHERE (is_migrated) AND NOT (exclude_list)

/* show premigrated files and tape */
RULE 'PREMIGRATED' LIST 'premigrated' SHOW('premigrated ' ||
xattr('dmapi.IBMTPS')) WHERE (is_premigrated) AND NOT
(exclude_list)
```

There is no need to include resident files, because these have no association to tape IDs. To run this policy, use the command:

```
# mmapplypolicy filesystem -P policyfile -f ./map -I defer
```

This command will create two output files named `./map.list.migrated` and `./map.list.premigrated`. The output file includes one line for each file with the following fields (also see section Lists):

```
5888 789909723 0  migrated 1 SLE022L6 -- /filesystem/file2
```

The three first fields are IBM Spectrum Scale internal numbers (inodenumber, inodegeneration, snapid). The 4th field is the file state given with the SHOW statement in the policy above. The 6th field is the tape ID and the 8th field is the file name.

As demonstrated above, with the SHOW clause it is possible show more information about a file. In the example above we show the extended attribute '`dmapi.IBMTPS`' for each file. There are more attributes that can be shown. The command mmlsattr allows to list the available attributes:

```
# mmlsattr -L -d /filesystem/file2

file name:            /filesystem/file2
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
```

```
flags:
storage pool name:    system
fileset name:         root
snapshot name:
creation time:        Wed Sep  3 13:33:31 2014
Windows attributes:   ARCHIVE OFFLINE
dmapi.IBMUID:         "17262253869394011850-5759258757141801152-
789909723-5888-0"
dmapi.IBMSGEN#:       "1"
dmapi.IBMTPS:         "1 SLE022L6"
dmapi.IBMProv:        "ltfs????"
dmapi.IBMObj:
"??????????d??c?r?c??*[?,??l???w?????????????????????????????????
???????????????????????? ??????????????/????????????????????"
gpfs.dmapi.region:
0x0000000000000000000000000000000007000000000000000
```

## Listing files based on time stamps

The time stamps of files can be used as a selection criterion, for example if you want to find out all files that have been created, modified or changed after a certain time stamp. The time stamps (ACCESS_TIME, MODIFICATION_TIME and CHANGE_TIME) are extended attributes for each file (see section File attributes)

The following rule lists all files that have been created, modified or changed after the time stamp of 2019-03-20 14:00:00:

```
RULE 'timestamp' LIST 'files' WHERE
( MODIFICATION_TIME >= TIMESTAMP("2019-03-20 14:00:00") OR
  CHANGE_TIME >= TIMESTAMP("2019-03-20 14:00:00") )
```

The TIMESTAMP format is either a string with the following syntax: “YYYY-MM-DD hh:mm:ss” or it can be UNIX epoch time.

Because the time stamp must be embedded in the LIST rule it is not practical to change the time stamp manually for iterative policy runs. It is better to use substitution variables. The substitution variable LAST is used in the policy shown below:

```
RULE 'timestamp' LIST 'files' WHERE
( MODIFICATION_TIME >= TIMESTAMP(LAST) OR
  CHANGE_TIME >= TIMESTAMP(LAST) )
```

The time stamp LAST is given with the mmapplypolicy command as substitution variable:

```
# mmapplypolicy filesystem -P policyfile
-M LAST==`date -d "2019-03-20 14:00:00" +%s` -I defer
```

In this example the time stamp LAST is expressed is in UNIX epoch format by the date function:

```
# date -d "2019-03-20 14:00:00" +%s
```

Of course, the policy engine can be programmed to store the resulting file list in a file.

## Processing file lists

As explained in section Lists EXTERNAL LIST policies can also be used to process files identified by the policy engine. This requires the interface script to be implemented and specified in the EXTERNAL LIST rule. The following example shows an EXTERNAL LIST policy that identifies files accessed within the last day and invokes the interface script /root/process_youngest.sh:

```
define(  exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.ltfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%'))

/* define access_age */
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)) )

/* define external list with interface script */
RULE EXTERNAL LIST 'mylist' EXEC '/root/process_youngest.sh' OPTS
test

/* define rule to identify youngest files */

RULE 'youngfiles' LIST 'mylist' WHERE (access_age < 1) AND NOT
(exclude_list)
```

This interface script is automatically invoked when this policy is executed by the `mmapplypolicy` command:

```
# mmapplypolicy filesystem -P policyfile
```

The interface script will receive 2 or more parameters as described below:

1. Parameter: string describing the operation. For an EXTERNAL LIST policy, the following strings are passed to the script: list and test
2. Parameter: name of the policy result file.
3. Parameter: the options given with  the first rule (optional) behind the OPTS clause

Based on these parameters the interface script for a list policy can initiate actions with the files identified by the list policy. The result file of a list policy includes the file names of the files matching the rules. There is one file per line with some additional information. Find below an example of a list policy result file:

```
48900 1741777473 0   -- /mnt/filesystem/file1
```

The three first numbers are IBM Spectrum Scale internal numbers (inodenumber, inodegeneration, snapid). The file name is the 5th field in the result file.

Find below some pseudo code for the implementation of the interface script /root/process_youngest.sh. Please note, based on the string describing the operation in the 1st parameter the 2nd parameter might either be the name of the file system (TEST) or the name of the policy result file (LIST):

```
#!/bin/ksh

# function to process the policy results
function process
{
  # this function now processes the policy results $fName
}

###### MAIN #######

# assign parameters
opCode="$1"
fName="$2"
opts="$3"

# check parameters
...

# based on op code perform action
rc=0
case $1 in
  TEST )
  #$1 is the policy operation (list, migrate, etc)
  #$2 is the file system name
  #$3 is the option given with EXEC '/root/process_youngest.sh'
OPTS test in the policy
  echo "TEST option received for directory $fName."
  if [[ ! -z "fName" ]] then
    if [[ -d "fName" ]] then
        echo "TEST directory $fName exists."
    else
        echo "WARNING: TEST directory $fName does not exists."
        rc=1
      fi
  fi;;

  LIST )
  #$1 is the policy operation (list, migrate, etc)
  #$2 is the policy file name
  #$3 is the option given with EXTERNAL LIST rule followed
      by the  OPTS clause
  echo "LIST option received, starting receiver task"
  if [[ ! -z "fName" ]] then
    if [[ -a "fName" ]] then
```

```
        echo "LIST file name $fName exists."
      else
        echo "WARNING: LIST file name $fName does not exists."
        exit 1
       fi
  fi

  # if the option is test, then just copy the
    policy result file, otherwise process each file
  if [[ "$opts" = "test" ]] then
    cp $fName /root/mypolicyresult.txt
  else
    echo "Processing file $filename"
    # calling generic function to process each file name
    process $fName
    rc=$?
  fi;;

  REDO )
  # sometimes you may get a REDO action code, not sure what to do
here, so I ignore it, alternatively you may call the process
function
  echo "REDO option received, exiting"
  rc=1;;

  * )
  echo "Unknow argument $1 received"
  rc=1;;
esac

exit $rc
```

This is just an example, it will not work this way. The function "`process()`" needs to be implemented.

Sample scripts and policies are stored on the GitHub repository [5] and is named receiver.

## Extracting file names from file list

As explained above the file lists generated by the policy engine include one file per line in the following format:

```
48900 1741777473 0   -- /mnt/filesystem/file1
```

The three first numbers are IBM Spectrum Scale internal numbers (inodenumber, inodegeneration, snapid). The 5th field contains the fully qualified path and file name. In the examples below the file list generated by the policy engine is provided in file outfile.

An easy way to extract the 5th field with the path and file name is using `awk` command:

```
# cat outfile  | awk ‘{print $5}’
```

However, this does not take into consideration blanks in the path and file name. To accommodate for blanks, it may be suitable to use the `cut` command:

```
# cat outfile | cut -d’ ‘ -f 7-20
```

However, the command above accepts a certain number of blanks in the path and file names. In addition, if there are double blanks these may combined into one and makes the path and file name invalid. To account for this, another more comprehensive hack can be used:

```
# cat outfile | awk -F '[ ]' '{ for(i=7; i<=NF; i++) printf
"%s",$i (i==NF?ORS:OFS) }'
```

# Appendix

## References

[1] IBM Spectrum Scale Knowledge Center Home
https://www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html

[2] Information Lifecycle Management in IBM Spectrum Scale Knowledge Center
https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.5/com.ibm.spectrum.scale.v5r05.doc/bl1adv_storage_mgt.htm

[3] Syntax of the mmapplypolicy command
https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.5/com.ibm.spectrum.scale.v5r05.doc/bl1adm_mmapplypolicy.htm

[4] Whitepaper: Configuring Spectrum Protect for Space Management (TSM HSM) or Spectrum Archive EE (IBM Spectrum Archive EE) with Spectrum Scale Active File Management:
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Storage%20Manager/page/Configuring%20IBM%20Spectrum%20Scale%20Active%20File%20Management

[5] GitHub repository including sample scripts and policies
https://github.com/nhaustein/spectrum-scale-policy-scripts

[6] File Attributes available to the policy engine:
https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.5/com.ibm.spectrum.scale.v5r05.doc/bl1adv_usngfileattrbts.htm

[7] IBM Spectrum Protect for Space Management with IBM Spectrum Scale - Configuration Guidance and best practices
https://www.ibm.com/support/pages/node/6116842

## Disclaimer