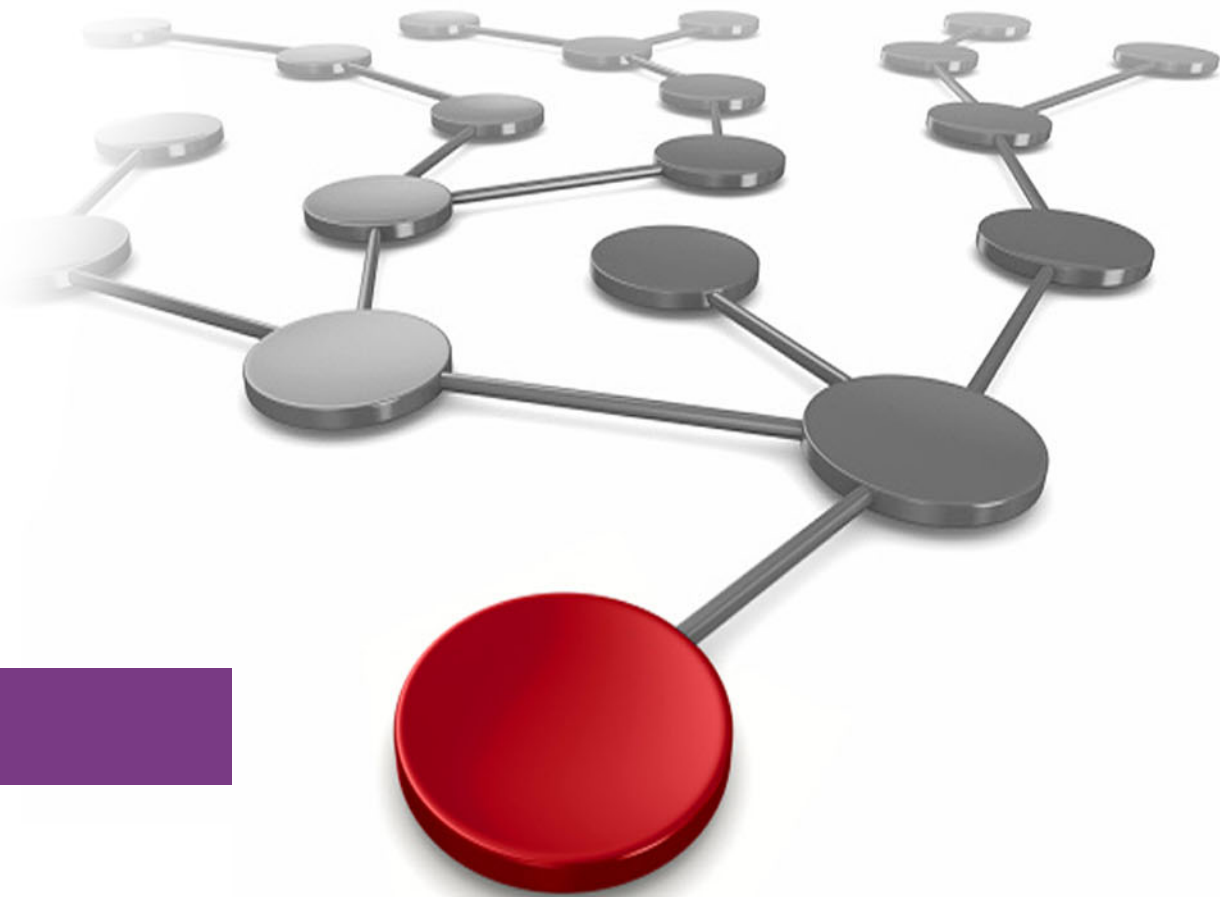


# IBM Storage Scale Information Lifecycle Management Policies

Nils Haustein



Storage





IBM Redbooks

**IBM Storage Scale ILM Policies**

October 2024

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (October 2024)**

This edition applies to Version 5, Release 2, Modification 0 of IBM Storage Scale.

This document was created or updated on October 28, 2024.

© Copyright International Business Machines Corporation 2024. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>Preface</b> .....	vii
Authors .....	vii
Now you can become a published author, too! .....	vii
Comments welcome .....	viii
Stay connected to IBM Redbooks .....	viii
<b>Chapter 1. Introduction</b> .....	1
1.1 The Basics .....	2
1.2 IBM Storage Scale policy engine .....	4
<b>Chapter 2. silverLM Policies and Rules</b> .....	7
2.1 Running a policy .....	8
2.1.1 Running an active policy .....	8
2.1.2 Running a scheduled policy .....	9
2.1.3 Options for mmapplypolicy .....	10
2.1.4 Controlling the number of files processed .....	12
2.1.5 String substitution in rules .....	13
2.1.6 Using existing file lists .....	14
2.2 File attributes .....	15
2.3 Placement rules .....	16
2.3.1 Applying placement policies .....	17
2.4 Migration rules .....	18
2.4.1 Automated migration .....	19
2.4.2 Migration Callback .....	20
2.4.3 Scheduled migration .....	21
2.4.4 External pools .....	21
2.4.5 Excluding files and directories .....	27
2.4.6 Pre-migration .....	30
2.5 Lists rules .....	32
2.5.1 Excluding files in List policies .....	33
2.5.2 Showing file attributes .....	34
2.6 Macros .....	35
<b>Chapter 3. Recall</b> .....	37
3.1 Recall with policy .....	38
<b>Chapter 4. Examples and use cases</b> .....	39
4.1 Placement policies .....	40
4.1.1 Placement by file name .....	40
4.1.2 Fileset placement .....	40
4.1.3 Prevent storing certain file types .....	41
4.2 Macro examples .....	41
4.2.1 HSM file states .....	41
4.2.2 Access age .....	42
4.2.3 File size conversion .....	42
4.2.4 AFM states .....	42

4.2.5 Weight macro . . . . .	42
4.3 Including rules in a policy file . . . . .	43
4.4 Threshold-based migration . . . . .	44
4.4.1 Three pool migration with callback . . . . .	45
4.4.2 Prioritized migration . . . . .	47
4.5 Manual or scheduled Migration . . . . .	48
4.5.1 Migrating multiple pools . . . . .	49
4.5.2 Grouping multiple pools . . . . .	50
4.5.3 Migration based on file size . . . . .	51
4.5.4 Migration based on modification age . . . . .	51
4.5.5 Migration based on fileset name . . . . .	52
4.5.6 Migration in AFM cache filesets . . . . .	52
4.6 Pre-migration and migration . . . . .	53
4.6.1 Pre-migration using a policy . . . . .	55
4.7 String substitution in rules . . . . .	55
4.8 Quota based migration for filesets . . . . .	57
4.8.1 Quota LIST policy . . . . .	57
4.8.2 Quota Migration policy . . . . .	58
4.8.3 Quota callback script. . . . .	59
4.8.4 Quota callback . . . . .	60
4.8.5 Further considerations . . . . .	61
4.9 Generating file lists . . . . .	62
4.9.1 Listing files by HSM state . . . . .	62
4.9.2 Listing tape IDs for files. . . . .	62
4.9.3 Listing files based on time stamps . . . . .	64
4.10 Processing file lists . . . . .	66
4.10.1 Extracting file names from file list . . . . .	68
<b>Related publications . . . . .</b>	<b>69</b>
IBM Redbooks . . . . .	69
Other publications . . . . .	69
Online resources . . . . .	69
Help from IBM . . . . .	70

# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <https://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

Redbooks (logo) ®

IBM®

Redbooks®

The following terms are trademarks of other companies:

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.



# Preface

This document provides comprehensive guidance for IBM® Storage 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 Storage Archive Enterprise Edition and IBM Storage Protect for Space Management. IBM Storage Scale is a clustered and scalable file system software from IBM. IBM Storage Scale includes a policy engine allowing to place, migrate and list files based on their attributes and characteristics.

## Authors

This paper was produced by a team of specialists from around the world working with the IBM Redbooks, Tucson Center.

**Nils Haustein** is a Senior Technical Staff Member in IBM Client Engineering Storage EMEA. He is responsible for designing and piloting file and object storage solutions. He co-authored the book “Storage Networks Explained”. As a leading IBM Master Inventor, he has created more than 200 patents for IBM and is a respected mentor for the technical community worldwide.

Thanks to the following people for their contributions to this project:

Marc A Chaplain  
**IBM Storage Infrastructure (retired)**

Dominic Müller-Wicke  
**IBM Storage Infrastructure**

Adam Christ,  
**Former IBM Storage Infrastructure**

Larry Coyne  
**IBM Redbooks®, Tucson Center**

## Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](https://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:

<https://www.linkedin.com/groups/2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/subscribe>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<https://www.redbooks.ibm.com/rss.html>



# Introduction

This Redpaper provides comprehensive guidance for IBM Storage 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 Storage Archive Enterprise Edition and IBM Storage Protect for Space Management. IBM Storage Scale is a clustered and scalable file system software from IBM. IBM Storage 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 Storage Scale ILM policies and focuses on the integration of IBM Storage Scale with IBM Storage Archive Enterprise Edition. The integration of IBM Storage Scale with IBM Storage Protect for Space Management is further described in [\[7\]](#). Readers of this document should have basic administrative knowledge regarding IBM Storage Scale concepts, architectures, and policies.

Before the concepts and syntax of placement, migration and list rules and policies are explained in Chapter 2, “silverILM Policies and Rules” on page 7, we give you a short introduction to IBM Storage Scale ILM and related file system and storage pool concepts (see 1.1, “The Basics” on page 2. At the end of this document in Chapter 4, “Examples and use cases” on page 39 you will find more examples and use cases for ILM policies.

**Note:** The examples shown in this document may not work “out-of-the-box” in your environment as they are intended to highlight the important concepts, terms, and operations. The author and IBM do not assume any liability regarding the examples given. For more information, see the IBM Documentation for IBM Storage Scale [\[1\]](#), [\[2\]](#).

## 1.1 The Basics

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

The IBM Storage 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 an analogy; it may not be implemented like this. Furthermore, the IBM Storage Scale policy engine manages the identified files, based on the rule provided action codes.

The policy engine executes rules that are programmed in a policy. A *rule* is an SQL-like statement that instructs the IBM Storage Scale policy engine what to do with a file in a specific storage pool if the file meets specific criteria. A rule consists of clauses and 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 subsections, the basic IBM Storage Scale file system and storage pool concepts are explained (section "IBM Storage Scale file systems and pools"), which are the basis for ILM implemented in IBM Storage Scale. 1.2, "IBM Storage Scale policy engine" on page 4 explains ILM policies in general and how the IBM Storage Scale policy engine works.

## IBM Storage Scale file systems and pools

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

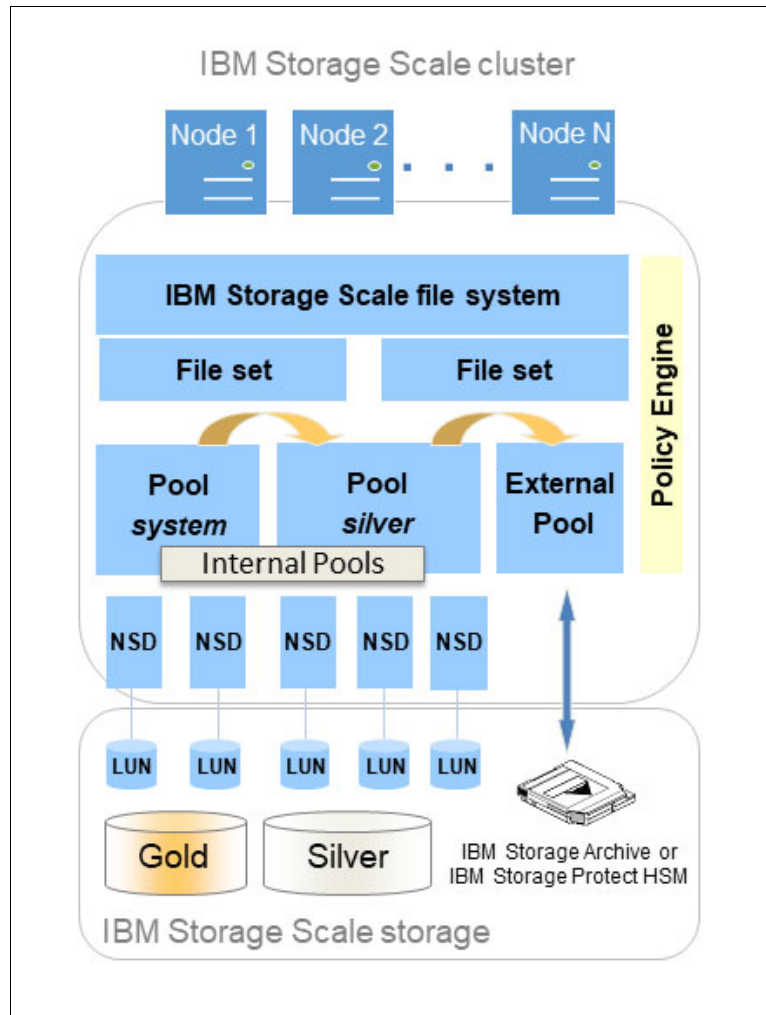


Figure 1-1 Basic IBM Storage Scale file system architecture

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

There are two kinds of IBM Storage 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 2.4.4, “External pools” on page 21). 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 Storage Protect for Space Management (SP HSM), or IBM Storage Archive Enterprise Edition. The interface script for the IBM Storage Protect HSM is the HSM client and for IBM Storage Archive EE it is the IBM Storage Archive client.

The term disk in the context of IBM Storage 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 one pool. All NSD in one internal pool are comprised of storage LUNs with similar characteristics. Referring to Figure 1-1 on page 3, 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.

**Note:** It is not best practice to configure 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, for example, 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 1-1 on page 3 are IBM Storage 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.

## 1.2 IBM Storage Scale policy engine

The IBM Storage Scale policy engine can identify files based on programmable conditions and manage their lifecycle. Lifecycle management includes the following operations:

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

In this section we focus on placement, migration, and listing.

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

Many policies - except for placement, threshold-based migration, and encryption policies - are executed by the `mmapplypolicy` command (see 2.1, “Running a policy” on page 8) which invokes the *policy engine*. This command requires at least a file system name and a policy filename as input. 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 for the defined scope (file system, directory, or fileset).
2. Rule evaluation and file selection: evaluates the rules provided in the policy file (top down) by matching file metadata with conditions programmed in rules and selects the matching files. Within a policy a given file is selected once by the first matching rule.
3. Execution of file operations for the selected files: 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 nodes, making the policy engine ultrafast and efficient.







## silverILM Policies and Rules

In this chapter policies, file attributes, placement, migration, list rules, and macros are described. It focuses on the basic syntax and semantics of rules and policies, detailed examples are given in Chapter 4, “Examples and use cases” on page 39.

This chapter starts with instructions for running the policy engine (see 2.1, “Running a policy” on page 8), followed by selected file attributes (see 2.2, “File attributes” on page 15) that can be evaluated by the policy engine. In the subsequent section rules and policies for placement (see 2.3, “Placement rules” on page 16), migration (see 2.4, “Migration rules” on page 18), lists (see 2.5, “Lists rules” on page 32, and macros (see 2.6, “Macros” on page 35) are explained.

Sample code and policies are stored in a GitHub repository [\[5\]](#).

## 2.1 Running a policy

A policy consists of a set of rules and the rules are executed by the policy engine. The IBM Storage Scale policy engine is represented by the command `mmappolicy` [3]. There are two types of policies:

1. *Automated policy*: are automatically executed. One may also call it active policy because the automated policy 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 automatically applied when a file is created in the file system. Migration rules in an automated policy are typically threshold-based and automatically executed when a file system pool reaches the rule-defined occupation threshold (see 2.4.1, “Automated migration” on page 19). When using migration rules in an automated policy a migration callback must be configured (see 2.4.2, “Migration Callback” on page 20).
2. *Scheduled policy*: are executed based on schedules. One may call it manual policy because unlike automated policies it must be started manually or by schedulers. A scheduled policy must not contain placement rules. Migration rules in a scheduled policy can be threshold-based (see 2.4.3, “Scheduled migration” on page 21). Migration rules in a scheduled policy do not require a callback to be configured.

An automated migration policy is automatically invoked when the storage occupation of a file system pool reaches a critical level (for example, 80% or 90%). When this occurs, then the threshold based migration rules of an automated migration policy must move file data to another pool in order to free up space. For this reason automated migration policies must be simple and efficient, especially the conditions for file selection. If the automated migration policy selects only certain files for migration, then the policy execution may not move sufficient capacity to another pool. This can cause a file system pool to run out of storage space.

A scheduled migration policy can have more sophisticated conditions for file selection. For example, a migration rule of a scheduled policy selects and moves files that were not accessed for 30 days.

An automated migration policy kicks in when a critical file system pool occupation level is reached. This may happen at times of high user workloads causing degradations for user workloads. Scheduled policies can be executed at times with fewer user workloads. Thus, the impact to user workloads can be minimized and is more predictive.

**Note:** Design scheduled migration policies to perform the daily job in accordance with the business needs. Design automated migration policies to be efficient if a pool reaches a critical occupation threshold. Consider automated migration policies as the last line of defense if scheduled policies cannot move enough data out of a pool.

This section explains how to execute automated (or active) and scheduled migration policies. Examples for both types of migration policies can be found in section 2.4.1, “Automated migration” on page 19 and 2.4.3, “Scheduled migration” on page 21.

### 2.1.1 Running an active policy

Each file system can have exactly one active policy that can be configured to run automated migration and other automated file management actions such as placement, encryption, etc. The automated policy for a file system is activated with the command shown in Example 2-1 on page 9.

*Example 2-1 Activating an automated or active policy for a file system*

---

```
# 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 Storage Scale observes the utilization of the file system pools and if the %high threshold is met, then it throws an event (lowDiskSpace). This event is caught by the callback (see 2.4.2, “Migration Callback” on page 20) 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 2.4.2, “Migration Callback” on page 20 and 2.1.3, “Options for mmapplypolicy” on page 10).

To show the active policy configured for a file system the following command shown in Example 2-2 can be used.

*Example 2-2 listing the active policy of a file system*

---

```
# 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.

## 2.1.2 Running a scheduled policy

With scheduled migrations the **mmapplypolicy** command is invoked by the scheduler or manually by an administrator. Example 2-3 shows the basic syntax of the **mmapplypolicy** command [3].

*Example 2-3 Executing the policy engine with a policy file*

---

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

---

The parameter **scope** defines the processing scope for the policy. It can be the file system name or path, or it can be a sub-directory within the file system. The scope directly influences how many inodes must be processed by the policy rules. It is best practice to define the scope in accordance with the needs. For example, if files from a fileset must be processed by the policy, then the scope should be the fileset path and not the entire file system path. This limits the number of inodes to be processed by the policy.

The parameter **policyfile** is the name of the file including the policy to be executed. Some important options are further explained in section 2.1.3, “Options for mmapplypolicy” on page 10.

**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 installed on the nodes providing the migration service, for example the nodes where the IBM Storage Archive EE software is installed. The interface script is defined in the external pool rule.

### 2.1.3 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 Storage Archive EE some **mmapplypolicy** options must be considered. See Table 2-1 on page 11.

Table 2-1 mmapplypolicy options

Option	Description	IBM Storage Archive EE Consideration
-m	Number of threads dispatched for file management operation (for example, migration) on every participating node.	Should be set to the number of tape drives / number of nodes, for example 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 (for example, 10,000), for large files choose a lower number (for example, 1,000).
-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 Storage Archive EE is installed and running. Must be specified. If not specified, then nodes specified in config parameter <b>defaultHelperNodes</b> 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 the number of files in file system to be scanned and selected. Default is <b>/tmp</b> . 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 <b>sharedTmpDir</b> . Default at Storage Scale version 5.0 is <b>.mmSharedTmpDir</b> 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.
-I	<p><b>test:</b> scan the file system and do not run the actual operation (for example, migration).</p> <p><b>defer:</b> create the file lists, but do not run the actual file operation (for example, migration).</p> <p><b>yes:</b> run the policy and file operations (default).</p>	<p>Use <b>-I test</b> to test the policy syntax.</p> <p>Use <b>-I defer</b> with LIST policies to generate the file lists.</p>
-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 you 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 on one node. 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 node names are specified with the parameter **-N**.

**Note:** With IBM Storage Archive EE it is important to control the number of files being dispatched for migration or recalls, see 2.1.4, “Controlling the number of files processed” on page 12 for more details.

Note of the **-s** parameter. The 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 Storage 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 (stored in file `mypolicy.txt`) that migrates to an external pool and for file system named *filesystem* the option **-I test** can be used as shown in Example 2-4.

*Example 2-4 Testing the syntax of a policy stored in file mypolicy.txt*

---

```
# 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 (Example 2-5) can be used.

*Example 2-5 Executing a policy stored in file mypolicy.txt for an external pool.*

---

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

---

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

## 2.1.4 Controlling the number of files processed

Internally IBM Storage 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 Storage Archive EE node and each thread processes a list of **B** files. The

numbers **m** and **B** are defined with parameters **-m** and **-B** with the **mmapplypolicy** command. In addition, the *SIZE* clause that can be provided with the EXTERNAL pool definition (see 2.4.4, “External pools” on page 21) 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 (**-B**) defines the maximum number of files in one file list. The *SIZE* clause defines the maximum total capacity of all files provided in one file list. The limit that is reached first is applied. For example, assume the bucket size (**-B**) is 10,000 and the *SIZE* clause is 10 GB. If the total capacity of a file list with 8,000 file names is 10 GB then the file list passed to one migration task contains 8,000 file names representing a total capacity of 10 GB. In this case the *SIZE* clause is the first limit that is applied.

Each thread 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 Storage 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:

$$\text{Files\_on\_queue} = (\text{Number of EE nodes}) \times (\text{parameter m}) \times (\text{parameter B})$$

This formula does not consider the *SIZE* limit. 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 shown in Table 2-2.

*Table 2-2 Maximum number of files on the queue recommendation*

Operation	Maximum number of files on queue
Migration	50.000
Recall	30.000

For example, let us assume the policy engine is started with the parameters: **-m 3 -B 1000** and the number of IBM Storage Archive EE Nodes participating in the migration is 2. The total number of files that appear on the queue is:  $2 \times 3 \times 1000 = 6000$ . If the policy engine selected more files for migration, then the next bunch of 1000 files is loaded on the queue if 1,000 files have finished processing.

Note, the limitations in Table 2-2 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 shown in Table 2-2.

## 2.1.5 String substitution in 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, in accordance with the setting of the **-M** parameter.

For example, assume you have two file systems, each should be migrated to a different IBM Storage Archive EE pool.

The IBM Storage Archive EE pool name is part of the EXTERNAL POOL rule definition. Instead of creating two different policy files, one can be created that looks like Example 2-6.

*Example 2-6 Define rules parameters for use by single policy*

---

```
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm' OPTS 'EEP00L'  
RULE 'MigToLTFS' MIGRATE FROM POOL 'system' TO POOL 'ltfs'  
WHERE ((LOWER(NAME) LIKE '%.mp3') AND (KB_ALLOCATED > 0))
```

---

Notice the string EEP00L 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 Storage Archive EE pool Pool1@Lib1 and data from fs2 is to be migrated to pool Pool2@Lib1, as shown in Example 2-7.

*Example 2-7 -M string substitution option with mmapplypolicy command*

---

```
# mmapplypolicy /fs1 -P mypolicy.txt -m 3 -n 1 -B 1000 -N eenode1,eenode2  
-single-instance -M "EEP00L=-p pool1@lib1"  
# mmapplypolicy /fs2 -P mypolicy.txt -m 3 -n 1 -B 1000 -N eenode1,eenode2  
-single-instance -M "EEP00L=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 2.1.5, “String substitution in rules” on page 13 for more details how to implement and use this.

## 2.1.6 Using existing file lists

The policy engine can use existing file lists instead of scanning the name space. For example, if you have the path and file names for a set of files and want to process a subset of files that match certain conditions, 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, as shown in Example 2-8.

*Example 2-8 Simple example of file list format with one path and file name per line*

---

```
/path/file1  
/path/file2  
...
```

---

To run the policy engine with an input file list (`existing_files.list`) the following command shown in Example 2-9 can be used.

*Example 2-9 Executing the policy engine with an existing file list*

---

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

---

The policy engine does not scan the file system (fsname) but use the existing file list passed with the parameter **-i existing\_files.list** as input and apply the policy rules against these files. The resulting file list is stored under the filename `./my.[.].list`.

Examples for producing file lists can be found in 4.9, “Generating file lists” on page 62.



If you have an existing file list created by the policy engine, then you must extract the path and file names. See 4.10.1, “Extracting file names from file list” on page 68 for more information about extracting path and file names from file lists produced by the policy engine.

## 2.2 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 clause *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 Storage Scale Documentation [6]. Table 2-3 gives an overview about the most used file attributes.

Table 2-3 Overview of most used file attributes

File attribute	Description
NAME	Name of the file (without 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.

File attribute	Description
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 <a href="#">File Attributes available to the policy engine [6]</a> .
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 <code>dmap.i.BMTPS</code> . Further system defined attributes can be listed with the command <code>mm1sattr -L -d filename</code>

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. 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 Storage Scale file system block size.

File attributes can also be listed by LIST rules. More details for this can be found in 2.5.2, “Showing file attributes” on page 34.

## 2.3 Placement rules

A placement rule instructs IBM Storage 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 4.1.2, “Fileset placement” on page 40). File system pools selected for placement should allow storing data (usage type `dataAndMetadata` or `dataOnly`), otherwise files matching the placement rule criteria cannot be created (see 4.1.3, “Prevent storing certain file types” on page 41). Data cannot be placed in external pools (see 2.4.4, “External pools” on page 21).

There can be multiple placement rules for a file system, all in one active policy. However, too many and complex placement rules in a policy for a given file system can slow down the system because the placement rules are 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 shown in Example 2-10.

*Example 2-10 Basic placement rule syntax*

---

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

---

This rule has the name *rule-name* and instructs placing (SET) file names matching to *match-pattern* in pool *pool-name*. Rule names are optional and make the console output of a policy engine run better readable.

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 that 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 1-1 on page 3 in “IBM Storage Scale file systems and pools”. Both pools are comprised of (some) NSD allowing for data usage. This policy consists 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. See Example 2-11.

*Example 2-11 Placing mp3 files in silver pool and all other files in 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 clause transforms the file name to lower case letters causing that 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 evaluated from top to bottom and the first hit is applied. 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. Example 2-12 shows that the file is placed on 'system' pool.

*Example 2-12 mmlsattr command 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 Chapter 4, “Examples and use cases” on page 39.

## 2.3.1 Applying placement policies

Placement rules must be applied to a file system as an automated policy using the `mmchpolicy` command (see section 2.1.1, “Running an active policy” on page 8). The `mmchpolicy` command allows applying one policy comprising multiple rules to a file system. To apply a policy for a file system it must be written to a file. See Example 2-13.

*Example 2-13 Apply a placement policy (filesysetm\_policy.txt) to 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 2.4.1, “Automated migration” on page 19 and Chapter 4, “Examples and use cases” on page 39 more comprehensive examples for placement policies are shown, also in combination with rules for migration.

## 2.4 Migration rules

A migration rule instructs IBM Storage 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 moved remain accessible in the global name space of the IBM Storage Scale file system. When migrating from an internal pool to another internal pool - such as from the 'system' pool to the 'silver' pool (Figure 1-1 on page 3) - the files remain under the control of IBM Storage 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 2.4.4, “External pools” on page 21).

With the migration to external pools, a stub-file is left in the IBM Storage 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 shown in Example 2-14.

*Example 2-14 Basic syntax for a migration rule*

---

```
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 can be lower in the storage hierarchy than the destination pool, for example, allowing migrating from tape back to disk. With the `THRESHOLD` clause 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 migrate rule is only executed if the occupancy percentage of the FROM pool is greater than or equal to the %high value.
- %low** The migrate rule execution 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 storage capacity 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 2.4.6, “Pre-migration” on page 30). Pre-migration is not relevant when migrating between internal pools because it requires more disk capacity in the pools because a pre-migrated files is available in both pools. Pre-migration is relevant when migrating from internal pool to external pool.

The *WEIGHT* clause allows to prioritize files for migration, for example, based on size (KB\_ALLOCATED) or access time (ACCESS\_TIME).

The FILESET clause allows further filtering of files subject for migration based on their storage location within the file system directory structure. The *WHERE* clause defines conditions for file selection.

As described in 1.2, “IBM Storage Scale policy engine” on page 4 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 2.4.1, “Automated migration” on page 19). It is invoked automatically when the threshold is reached. The threshold describes an occupation percentage of a file system pool.

Scheduled migration is run manually or by a schedule and is typically independent of thresholds (see 2.4.3, “Scheduled migration” on page 21). With scheduled migration, the selection criteria of files can be more granular based on file attributes. Typical criteria for file migration are:

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

More information about file attributes can be found in 2.2, “File attributes” on page 15.

A special form of migration is pre-migration (see 2.4.6, “Pre-migration” on page 30). 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 2.4.5, “Excluding files and directories” on page 27. This is particularly important to consider when migrating to external pools (see 2.4.4, “External pools” on page 21) because access to files on external pools might be slow.

## 2.4.1 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, then files should be migrated to another pool (internal or external). Each file system can have its own automated migration rules within an automated policy. There can only be one automated policy per file system and an automated policy can contain threshold-based migration rules as well as placement and encryption rules. Example 2-15 shows the syntax of a threshold-based migration rule.

*Example 2-15 Syntax of a threshold-based migration rule*

---

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

---

This rule instructs the IBM Storage Scale policy engine to migrate files from pool 'system' to pool 'silver' (see Figure 1-1 on page 3 in 1.2, “IBM Storage Scale policy engine” on page 4) when the pool 'system' is occupied 80% or more. The migration is done until the 'system' pool occupation drops below 70%. The *WEIGHT* clause 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 2.3, “Placement rules” on page 16) with a migration policy the policy file and the `mmchpolicy` command for file system `/filesystem` looks like Example 2-16.

*Example 2-16 Placement policy with a migration policy*

---

```
# 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 Example 2-16 is the `mmchpolicy` command used to applying the policy to the file system (see 2.1, “Running a policy” on page 8). 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 2.4.2, “Migration Callback” on page 20.

More examples for automated migration rules are given in 4.4, “Threshold-based migration” on page 44. The most basic advice for automated migration rules is to keep 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 Storage Scale file system to go offline.

Rules shown in Example 2-16 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 2.4.4, “External pools” on page 21).

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 4.8, “Quota based migration for filesets” on page 57 we discuss an approach to trigger the migration when quota limits for filesets have been exceeded.

Automated migration policies can also be configured using the IBM Storage Scale GUI. Navigate to **Files** → **Information Lifecycle Management**. Select a file system and **Add rule**. Now use the graphical elements to configure your automated policy and apply it. This will also configure a migration callback.

## 2.4.2 Migration Callback

To start an automated migration policy when a certain threshold is reached for a file system pool, a callback must be configured in the IBM Storage Scale cluster. A callback is triggered by defined events and invokes a defined script to be executed. A callback has a cluster-wide scope and is invoked for all file systems when one of the defined 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 command in Example 2-17 shows the configuration of a standard migration callback.

*Example 2-17 IBM Storage Scale command to configure a callback for migration*

---

```
# 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 Storage 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 2.1, “Running a policy” on page 8). The script `/usr/lpp/mmfs/bin/mmstartpolicy` invokes 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 4.4.1, “Three pool migration with callback” on page 45.

## 2.4.3 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 shown in Example 2-18.

*Example 2-18 Basic syntax of a scheduled migration rule*

---

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

---

This rule instructs the IBM Storage Scale policy engine to migrate files last accessed 30 days ago or longer from pool 'system' to pool 'silver' (see Figure 1-1 on page 3 in 1.2, “IBM Storage Scale policy engine” on page 4). This rule is only valid for migration between internal pools, such as pools 'system' and 'silver' according to Figure 1-1 on page 3. When migrating to external pools an additional rule is required (see 2.4.4, “External pools” on page 21).

To execute a scheduled policy the **mmapplypolicy** command is used (see 2.1, “Running a policy” on page 8). More examples for scheduled migration policies are given in 4.5, “Manual or scheduled Migration” on page 48.

## 2.4.4 External pools

As explained in section 1.2, “IBM Storage Scale policy engine” on page 4 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 Storage Archive Enterprise Edition and IBM Storage Protect for Space Management (SP HSM).

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 in Example 2-19.

*Example 2-19 EXTERNAL POOL rule used to define an external pool*

---

```
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 “program” given in the EXEC clause specifies the interface script used to manage files in the external pool. The OPTS parameter allows passing additional options to the external pool script (see also 4.10, “Processing file lists” on page 66). In the context of IBM Storage 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 named within the EXEC clause must be installed on all nodes that perform the migration to external pools. In addition, the policy engine (`mmapp1ypol1cy`) should be invoked on a node that has the interface script installed. Nodes, that do not have the interface script installed produce error messages.

Example 2-20 shows an automated migration policy from an internal pool 'silver' to an external pool 'ltfs' when the 'silver' pool is 90% occupied.

*Example 2-20 Automated migration policy from internal pool “silver” to external pools*

---

```
/* 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/eedm'
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 Storage Archive EE it is important to exclude at least these subdirectories (see 2.4.5, “Excluding files and directories” on page 27). The path name '%/.ltfsee/%' refers to the actual directory of the IBM Storage 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' that is managed by the interface script /opt/ibm/ltfsee/bin/eedm. The **eedm** command is invoked with the migrate option because the third rule is a MIGRATE rule. The **OPTS** parameter denotes the IBM Storage Archive EE tape pool and library names used as destination of the migration. Up to three pool @ library clauses can be given here (separated by comma) whereby the migration job creates three copies in three different tape pools (see “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 2.1.5, “String substitution in rules” on page 13). The **SIZE** clause defines the total size of all files passed in one file list to the **eedm 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 **eedm migrate** commands with different file lists are executed (see section 2.1.4, “Controlling the number of files processed” on page 12).

The third rule 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, Example 2-21, can be combined with the automated policy for migrating from 'system' to 'silver' pool (see 2.4.1, “Automated migration” on page 19) and the placement policy for mp3-files (see 2.3, “Placement rules” on page 16).

*Example 2-21 Policy combining rules for migrating from pool “system” to “silver” to “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%')

/* Migration rule for internal pools 'system' to silver */
RULE 'autoMigSystem' MIGRATE FROM POOL 'system' THRESHOLD(80,70)
WEIGHT(CURRENT_TIMESTAMP - ACCESS_TIME) TO POOL 'silver'

/* Migration rules from silver pool to external pool ltfs*/
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm'
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:

- ▶ Defines an exclude rule for directories in the file system that should not be migrated (see 2.4.5, “Excluding files and directories” on page 27).
- ▶ Threshold based migration for internal pools from pool 'system' to pool 'silver' when 'system' pool has reached 80%.
- ▶ Threshold based migration from internal pool 'silver' to external pool 'ltfs' 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 2.4.1, “Automated migration” on page 19 (threshold based)).

The following policy in Example 2-22 is a scheduled policy that migrates files last accessed 30 days ago or longer from pool 'silver' to pool ltfs:

*Example 2-22 Scheduled policy that migrates files last accessed 30 days ago or longer*

---

```
/* 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%')

/* Migration rules for migration from silver to ltfs */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm'
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 2.4.5, “Excluding files and directories” on page 27) and then migrates files accessed 30 days ago or longer from pool 'silver' to pool 'ltfs' that is managed by IBM Storage Archive EE. Files are migrated to the IBM Storage Archive EE pool Pool1@Lib1.

To run this policy, see 2.1, “Running a policy” on page 8. More examples for policies and use cases with external pools can be found in Chapter 4, “Examples and use cases” on page 39.

## Creating multiple copies on tape

IBM Storage 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 Example 2-23. The tape pools can consist of tapes located in up to 2 tape libraries. Alternatively, the tape pools can be in different logical libraries of the same physical library.

The following policy, Example 2-23, defines a rule to migrate all files older than 30 days to two tape pools. One tape pool (Pool1@Lib1) consists of tapes in library 1, and the second tape pool (Pool2@Lib2) consists of tapes in library 2.

*Example 2-23 Scheduled policy migrating files older than 30 days to two tape pools*

---

```
/* 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%')

/* Migration rules for migrating from silver to ltfs with two pools */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm' 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 creates two copies on a two different tape for each selected file: one on Pool1 @ Lib1 and one on Pool2 @ Lib2. To leverage a second library, the IBM Storage Archive EE must be configured for multi-library support and must have attached two distinct logical libraries. It is also possible to create up to three copies within one tape library. This requires up to three tape pools to be configured, all using tape within one logical tape library “lib”, for example pool1 @ lib1, pool2 @ lib1 and pool3 @ lib1.

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 Storage 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 limited. The following policy, shown in Example 2-24, migrates all files in directory /ibm/gpfs0/data1 from the associated tapes to the IBM Storage Scale file system. The file system pool ('system' in this case) matters. It must be the same pool from which the files were migrated.

*Example 2-24 Policy migrating back to internal disk from tape*

---

```
/* recall all files under /ibm/gpfs0/data1 directory */
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm'
RULE 'ee_recall' MIGRATE FROM POOL 'ltfs' TO POOL 'system'
WHERE LOWER(PATH_NAME) LIKE '/ibm/gpfs0/data1/%'
```

---

As shown in Example 2-24 the *OPTS* clause in the EXTERNAL POOL rule can be omitted. Likewise, the EXCLUDE rules must not be specified, because files from these directories should not have been migrated to tape in the first place. The use of FOR FILESET does not work for the recall policy.

This policy can be run using the `mmapp1policy` command (see 2.1, “Running a policy” on page 8).

## External pool considerations

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

Using IBM Storage Scale *Active File Management (AFM)* on a fileset where files are migrated to an external pool has some limitations depending on the AFM mode of the fileset [4]. Currently, IBM Storage Archive EE only support AFM Independent Writer mode. AFM is based on a home - cache model. In a simple AFM configuration, the home IBM Storage Scale cluster exports a fileset which is cached on the cache fileset of the cache IBM Storage 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 Storage 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 Storage 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" or "uncached" are not migrated, otherwise this may cause error messages. To prevent migration of "dirty" or "uncached" files, configure the migration policy to migrate files that have been modified a reasonable amount of time before they are migrated (see section 4.5.6, "Migration in AFM cache filesets" on page 52).

When invoking the policy engine for the migration to an external pool some specific parameters should be considered (see 2.1, "Running a policy" on page 8). This may also apply to the migration callback which also invokes the policy engine (see 2.4.2, "Migration Callback" on page 20).

Prevent migration of *certain sub-directories used for IBM Storage Scale internal processing* in a file system or fileset. IBM Storage Scale internal processing directories are placed 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 2.4.5, "Excluding files and directories" on page 27).

## 2.4.5 Excluding files and directories

For migration policies, especially when migrating to external pools, it is important to exclude certain files and directories 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 executed for. The most common files and directories to be excluded are:

- ▶ Directory `.SpaceMan` including configuration and temp files for HSM.
- ▶ Directory `.snapshots` including snapshots.
- ▶ Directory for the IBM Storage Archive EE metadata cache (`.l1tfsee`) if this has been set up 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`).
- ▶ When AFM is used, then exclude AFM meta directories that reside in AFM cache filesets (`.afm`, `.ptrash`, `.pconflicts`).

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

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

- ▶ EXCLUDE rule within a migration policy (see "Using EXCLUDE rules").
- ▶ EXCLUDE macro used within the *WHERE* clause of a migration rule (see "Using Exclude macros").

The fundamental difference between both methods is at which point of the policy execution files are excluded (see 1.2, "IBM Storage Scale policy engine" on page 4). Depending on the complexity of the exclude clause this can have performance impacts for the policy run. With EXCLUDE rules files are excluded before they are being evaluated by any migration policy. For this reason, EXCLUDE rules must come before migration rules in the policy file. Exclusion within the *WHERE* clause is evaluated with the migration rule. The latter one is therefore slower.

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

## Using EXCLUDE rules

IBM Storage Scale provides a rule type for excluding files and directories from being processed for migration (and deletion). The basic syntax of the exclude rule is shown in Example 2-25.

*Example 2-25 Basic syntax of EXCLUDE rule*

---

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

---

The DIRECTORY\_PLUS clause indicates that 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.

**Note:** The exclude rules must be placed before the migration rules in a migration 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. Policy Example 2-26 shows an EXCLUDE rule placed prior to the migration rule.

*Example 2-26 Policy shows an EXCLUDE rule placed prior to migration rule*

---

```
/* Exclude rule to exclude directories */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.l1tfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
PATH_NAME LIKE '%/.afm/%' OR
PATH_NAME LIKE '%/.ptrash/%' OR
PATH_NAME LIKE '%/.pconflicts/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* here comes the migration rules for silver to l1tfs*/
RULE EXTERNAL POOL 'l1tfs' EXEC '/opt/ibm/l1tfsee/bin/eeadm'
OPTS '-p pool1@lib1' SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'l1tfs' WHERE (further
conditions)
```

---

EXCLUDE rules should be embedded in automated (see 2.4.1, “Automated migration” on page 19) and schedule migration policies (see 2.4.3, “Scheduled migration” on page 21). EXCLUDE rules are important when migrating to external pools (see 2.4.4, “External pools” on page 21). EXCLUDE rules must be placed before the MIGRATE rules in a migration policy. Files selected by the EXCLUDE rule are not matched against the migration rule.

Excluding files using exclude rules is the recommended and most efficient way for excluding files from being migrated. EXCLUDE rules cannot be used in combination with LIST rules in a policy (see 2.5.1, “Excluding files in List policies” on page 33). To exclude files and directories with for LIST rules us exclude macros (see “Using Exclude macros”).

## Using Exclude macros

Example 2-27 shows another way for excluding files using an EXCLUDE macro within the *WHERE* clause of the migration (see 2.6, “Macros” on page 35).

*Example 2-27 EXCLUDE macro within the WHERE clause of the migration*

---

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

/* here comes the migration rule with exclude_list */
RULE EXTERNAL POOL 'lftfs' EXEC '/opt/ibm/lftsee/bin/eedm'
OPTS '-p pool1@lib1' SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'lftfs' WHERE NOT
(exclude_list)
```

---

This EXCLUDE macro "exclude\_list" is used as one condition in the *WHERE* clause of the MIGRATE policy. The macro "exclude\_list" is executed within the migration rule. This can add overhead to the MIGRATE rule execution because all files are matched against the EXCLUDE macro. The more efficient way for excluding files are 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 Chapter 4, “Examples and use cases” on page 39.

## Further excludes

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

The following examples of clauses should be inserted within the *WHERE* clause of the EXCLUDE rule or macro.

When backup with **mmbackup** is used, consider the exclude clauses in Example 2-28.

*Example 2-28 When backup with mmbackup consider these exclude clauses*

---

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

---

When Active File Management is used, consider the exclude clauses shown in Example 2-29 on page 30.

*Example 2-29 When AFM is used, consider these exclude clauses*

---

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

---

For older versions of IBM Storage Scale (< 5) the following excludes should be considered. See Example 2-30.

*Example 2-30 Consider using these exclude clauses with IBM Storage Scale older than 5*

---

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

---

## 2.4.6 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 does not free up storage capacity in the internal pool. Pre-migration is only possible when migrating from an internal pool to an external pool. For more information about external pools see 2.4.4, “External pools” on page 21.

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 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 expressing percentages: **THRESHOLD** (%high, %low, %premig)

- %high**      The rule is only executed 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 then files are migrated until the %low 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, not greater. The storage capacity 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 is between %low and %premig.

For example, the following rule pre-migrates all files to tape if the 'silver' pool occupancy is 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. See Example 2-31.

*Example 2-31 This rule pre-migrates all files to tape if the "silver" pool occupancy lies between 0 -30%*

---

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

```
/* DO 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 Storage Archive EE (see 2.4.4, "External pools" on page 21) and one defining the pre-migration. This policy can be run in a scheduled manner using the **mmapplypolicy** command (see 2.1, "Running a policy" on page 8). It must 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 pre-migration or migration . When setting %high to 0, then pre-migration is triggered all the time. The rule in Example 2-32 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' until whereby the storage capacity of files being pre-migrated is approximately 60% of the total 'silver' pool capacity (70 - 10). This policy could be used in conjunction with automated migration.

*Example 2-32 Automated policy with pre-migration rule*

---

```
/* 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/eedm'  
OPTS '-p pool1@lib1' SIZE 10485760
```

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

---

Example 2-32 first defines the exclude rule (see section 2.4.5, "Excluding files and directories" on page 27), then defines the external pool 'ltfs' managed by IBM Storage 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 files were 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 thresholds are derived from the file system and not from the fileset. Example 2-33 shows how the policy pre-migrates all files in fileset 'test' if the occupancy of the pool 'silver' is below 70%.

*Example 2-33 Schedule pre-migration rule for fileset "test"*

---

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

---

Running the policy in Example 2-33 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 4.6, “Pre-migration and migration” on page 53.

An alternate method for running pre-migration in a scheduled manner is explained in 4.6.1, “Pre-migration using a policy” on page 55).

## 2.5 Lists rules

The IBM Storage Scale policy engine can also be used to list files stored in an IBM Storage Scale file system and matching certain conditions. With such lists of files statistics can be derived, the effectiveness of migration policies can be checked, or selected files can be processed. The advantage to use the IBM Storage 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.

An EXTERNAL LIST policy has two rules, as shown in Example 2-34.

*Example 2-34 Example of an EXTERNAL LIST policy with two rules*

---

```
RULE 'rule-name' EXTERNAL LIST 'list-name' EXEC 'program'
OPTS options
RULE 'rule-name' LIST 'list-name' WHERE (conditions)
```

---

The first rule defines the external list name (list-name) and an interface script (program) that can be invoked to process the selected file list. For more details about interface scripts with EXTERNAL LIST rules refer to section 4.10, “Processing file lists” on page 66. 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 defines the conditions for file selection.

To execute an EXTERNAL LIST policy the `mmapplypolicy` command can be used. Example 2-35 demonstrates executing the EXTERNAL LIST policy and storing the selected file names in a list.

*Example 2-35 Executing an EXTERNAL LIST policy and storing the list of selected files*

---

```
# 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 string `'external-list-name'` is equivalent to the EXTERNAL LIST name (in Example 2-36 on page 33 this is `'list-name'`). Using the prior examples 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 conditions in 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. Example 2-36 shows a list policy result file.

*Example 2-36 Format of the list of selected files produced by the policy engine*

---

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

---

The three first numbers are IBM Storage Scale internal numbers (inodenum, inodegeneration, snapid). The file name is the 5<sup>th</sup> field in the result file. See 4.10.1, “Extracting file names from file list” on page 68 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. See Example 2-37.

*Example 2-37 EXTERNAL LIST policy that 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. The second rule selects all files that are migrated. To run this policy stored in file name policyfile use the following command in Example 2-38.

*Example 2-38 Executing the external list policy and storing the result in file name file.list.migrated*

---

```
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 4.9, “Generating file lists” on page 62.

EXTERNAL LIST policies can also be used to process files selected by the policy engine. This requires an interface script to be implemented and specified in the EXTERNAL LIST rule. This interface script is automatically invoked when the policy is executed.

More examples for processing file lists can be found in 4.10, “Processing file lists” on page 66.

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

## 2.5.1 Excluding files in List policies

It is also possible to exclude files within in a LIST rule using the LIST EXCLUDE rule. See Example 2-39.

*Example 2-39 Using LIST EXCLUDE rule to exclude files with LIST rules*

---

```
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 within the WHERE clause are not selected. The LIST EXCLUDE rule must come first within a LIST policy because file that are excluded by the LIST EXCLUDE rule are not matched in a LIST rule.

## 2.5.2 Showing file attributes

LIST rules can also show other file attributes using the SHOW clause, for example, to show the file size the following LIST rule can be used. See Example 2-40.

*Example 2-40 LIST rule showing the file size for file selected by the WHERE clause*

---

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

---

Example 2-41 shows the resulting output after executing the policy shows the file size in the 4<sup>th</sup> field.

*Example 2-41 Format of the list of selected files produced by the policy engine including file size*

---

```
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 shown in Example 2-42 can be used.

*Example 2-42 Rule to show the file size and the tape ID for migrated files*

---

```
/* 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 Example 2-43 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.

*Example 2-43 Output produced by the policy engine*

---

```
48900 1741777473 0 size=4096 tape=1
SLE033L6033490ef5-be30-4f23-b328-527b5e3109a4@0000013100730409 -- /mnt/filesystem/file1
```

---

Note, the string displayed for the tape ID has two numbers. The first number is the number of the copy on tape. The number 1 means that this is the first copy. The second number includes the tape ID and a long representation of the tape pool and library ID. These values are obtained from the extended attribute dmapi.IBMTPS.

## 2.6 Macros

Macros allow defining more complex clauses and conditions to make rules better readable. Excluding files for migration - as shown in 2.4.5, “Excluding files and directories” on page 27 - is one example for macros. The basic syntax for macros is shown in Example 2-44.

*Example 2-44 Basic syntax for macros including macro name and condition or expression*

---

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

---

The macro-name is an arbitrary name which is used as condition in a rule. The condition must follow the rule syntax. Example 2-45 is a macro example that defines the state of a migrated file.

*Example 2-45 Macro “is\_migrated” defines the migrated file state*

---

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

---

The macro names can now be used as conditions in rules. Example 2-46 shows how threshold-based policy can use macros.

*Example 2-46 Define migration rule using the macro “is\_migrated”*

---

```
/* define exclude_list macro */
define( exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.l1tfsee/%' 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 'l1tfs' EXEC '/opt/ibm/l1tfsee/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 'l1tfs' WHERE
NOT (exclude_list) AND NOT (is_migrated)
```

---

For more examples, please see 4.2, “Macro examples” on page 41.





## Recall

This chapter will discuss how recalls apply to ILM configurations with external pools (see section 2.4.4, “External pools” on page 21). A recall is a reverse migration process. While migration moves or copies files from an internal pool to an external pool, recall copies the file from the external pool (tape) to the internal pool (disk).

Recalls can be triggered in two ways:

- ▶ **Transparent:** upon file access a migrated file is automatically recalled from tape to the Storage 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 takes a list of files to be recalled as input and recalls the files from tape to disk in an optimized manner by sorting the file by their tape-ID and position on tape.

Tape optimized recalls are faster and resource efficient [8]. With tape optimized recalls each required tape is mounted once and files are read in accordance with their sequence on tape. With transparent recalls, required tapes are mounted multiple times. For each mount one or a few files are read from any location on the tape. If more than one file is read from one tape, than these files are not read in sequence, causing the tape drive to locate back and forth on the tape.

The implementation of tape optimized recalls is disruptive to the user. The user must contact the administrator and provide a list of file names to be recalled. The administrator must compose a list containing fully qualified path and file names and then issue the tape optimized recall request using the composed file list. Tape optimized recalls avoid recall storms triggered by many transparent recall requests at the same time. There are ways to automate tape optimized recalls [9].

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

## 3.1 Recall with policy

Recalls for a selected set of migrated files can be performed using MIGRATE policies. Example 3-1 shows the policy that recalls all migrated files residing in folder /mypath. The external pool rule executes the **eeadm** command but does not require any further options.

*Example 3-1 Policy that recalls all migrated files residing in folder /mypath*

---

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

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

/* define migration rule for recall */
RULE 'recall' MIGRATE FROM POOL 'ltfs' TO POOL 'system' WHERE
(recall_dir)
```

---

The first rule defines a macro including the path name of files to be recalled. The second rule defines the external pool without any options. And the third rule migrates from pool 'ltfs' to pool 'system' which is a reverse migration.

**Note:** The recall policy only recalls files that were migrated from pool 'system'. Files that were migrated from other pools to tape are not recalled. To recall those files, the TO POOL clause must match the pool from where the files were migrated.

To execute the recall policy the command in Example 3-2 can be used. The recall policy is stored in recall-policy.txt.

*Example 3-2 Executing the recall policy*

---

```
mmapplypolicy /mypath -P recall-policy.txt -N eenodes -m 2 -B 1000
-single-instance
```

---

With the **mmapplypolicy** command above it is important to consider parameters **-N**, **-m** and **-B**. The parameter **-N** defines the node names of the IBM Storage Archive nodes. The parameter **-m** defines the number of parallel threads per node. This parameter shall not be larger than the number of tape drives attached to one node. The parameter **-B** defines the number of files recalled by one thread. For small files the parameter **-B** can be larger than 1000. For large files, the parameter **-B** can be smaller than 1000.





## Examples and use cases

This chapter presents more examples for IBM Storage Scale ILM rules and policies. For additional background about the syntax and semantic refer to Chapter 2, “silverILM Policies and Rules” on page 7.

Sample scripts and policies are published in a repository on GitHub [\[5\]](#).

For IBM Storage Archive Enterprise Edition users example policies are available in `/opt/ibm/ltfsee/share/` of any server where IBM Storage Archive Enterprise Edition is installed.

## 4.1 Placement policies

In this section some more examples for placement policies are given. Remember, placement can only be done on internal pools (see 2.3, “Placement rules” on page 16). The basic syntax of a placement rule is shown in Example 4-1.

*Example 4-1 Basic syntax of placement rules*

---

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

---

The conditions in the *WHERE* clause select files based on file attributes to be placed in the pool referenced by 'pool-name'. The following attributes can be used in the conditions: *FILESET\_NAME*, *USER\_ID*, *GROUP\_ID*, and *NAME*.

Placement policies are applied to a file system using the `mmchpolicy` command as shown in Example 4-2.

*Example 4-2 Command for applying placement rules defined in 'policyfile'*

---

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

---

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

### 4.1.1 Placement by file name

The importance of files stored in an IBM Storage 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. In Example 4-3 the 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').

*Example 4-3 Policy places excel-files (.xls) in pool 'system' and other files in pool 'silver'*

---

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

---

Place the default placement rule at the end.

### 4.1.2 Fileset placement

Filesets are logical partitions in a file system that can be managed separately, for example with quota and snapshots. With placement rules dependent and independent 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 1-1 on page 3 “IBM Storage Scale file systems and pools”). Pool 'system' consists of Flash storage while pool 'silver' consists of NL-SAS disk. The file system has two filesets, one fileset for storing home folders for 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 Flash storage. Example 4-4 on page 41 shows a placement policy that stores all files of the fileset `homefileset` in pool 'silver' and all files stored in the fileset `dbfileset` in pool 'system'.

*Example 4-4 Placement policy aligning fileset to pools*

---

```
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.

### 4.1.3 Prevent storing certain file types

Placement policies can also be (mis-)used to prevent storing files matching certain criteria. This can be achieved by creating a pool with only one NSD of type descOnly (see Chapter 2, “silverILM Policies and Rules” on page 7). Let us name the pool 'dummy'. Example 4-5, shows how to place mp3-files on pool 'silver', mov-files on pool dummy and all other files on pool system.

*Example 4-5 Policy to store mp3-files on pool 'silver', mov-files on pool dummy*

---

```
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' that 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 a .mov file. The user can store any other file whereby .mp3 files will be stored in pool 'silver'. Notice the order of the rules, the default rule must be at the end.

## 4.2 Macro examples

As explained in 2.6, “Macros” on page 35, macros allow to make rules easier to read by pre-defining more complex conditions and expression.

### 4.2.1 HSM file states

Example 4-6 macro defines HSM states a file can have such as migrated, pre-migrated, and resident.

*Example 4-6 Macro defining HSM states*

---

```
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%') )
```

---

## 4.2.2 Access age

In Example 4-7, macros define the age of files in days based on access time and modification time.

*Example 4-7 Macros defining 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)) )
```

---

## 4.2.3 File size conversion

For larger files it might be valuable to evaluate the file size in megabytes, instead of kilobytes. Example 4-8 macro defines the file size in megabytes.

*Example 4-8 Macro to convert file size in MB*

---

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

---

## 4.2.4 AFM states

Example 4-9 macro defines different AFM states from an AFM cache. The following AFM file states exist:

- ▶ Uncached files are not yet fetched to cache or are evicted.
- ▶ Dirty files are changed on cache and not yet replicated to home.
- ▶ Local files are fetched to cache.

*Example 4-9 Macros defining different AFM file states in an AFM*

---

```
define( is_uncached, ( NOT REGEX(misc_attributes, '[u]') ) )

define( is_dirty, ( REGEX(misc_attributes, '[P]') AND
  REGEX(misc_attributes, '[w|v|x|y|j]') ) )

define( is_local, ( NOT RegEx(misc_attributes, '[DXaw]') AND
  RegEx(misc_attributes, '[z]') AND NOT RegEx(mode, '[bcps]') ) )
```

---

## 4.2.5 Weight macro

The *WEIGHT* clause allows to prioritize selected files for processing. Files are selected based on the condition in the *WHERE* clause. The order selected files are processed can be influenced with the *WEIGHT* clause. Files with the highest weight are processed first.

If the *WEIGHT* clause is not explicitly given in a policy, then the default weight is based on kilobyte allocated (*KB\_ALLOCATED*). This means that files with the largest allocated capacity are processed first

Example 4-10 on page 43 defines a macro assigning weight numbers to files based on file attributes.

*Example 4-10 Prioritize files based on file attributes*

---

```
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  
  )  
)
```

---

With this *macro weight expression on files* with a low *access\_age* get a low weight. Files with a small size (*mb\_allocated*) are weighted by their *access\_age*. Large pre-migrated files that were not accessed for longest time get a high weight. All other files are weighted by their size and access age.

The macro *weight\_expression* uses the macros *access\_age*, *mb\_allocated*, and *is\_premigrated*, that are explained prior in this section. The macros *weight\_expression* is used with the *WEIGHT* clause in a *MIGRATE* policy as shown in Example 4-11.

*Example 4-11 Macros weight\_expression with WEIGHT clause in a MIGRATE policy*

---

```
/* define external pool managed by LTFS */  
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm'  
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 (exclude_list) AND NOT (is_migrated)
```

---

When using this policy shown in Example 4-11 all of the macros must be embedded in the policy file, including the macro *weight\_expression*.

## 4.3 Including rules in a policy file

Common macros, exclude rules and other rules that are used in multiple policies can be written to a separate policy file and included into other policy files. The syntax for including policy files is shown in Example 4-12.

*Example 4-12 Basic syntax for including policy files*

---

```
include (include-file.txt)
```

---

Example 4-13 shows macros and exclude rules that are used in multiple policies.

*Example 4-13 Common macros are stored in a include files*

---

```
/* common excludes */
RULE 'exclude' EXCLUDE WHERE
  ( PATH_NAME LIKE '%/.SpaceMan/%' OR
    PATH_NAME LIKE '%/.ltfsee/%' OR
    PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
    PATH_NAME LIKE '%/.snapshots/%' )
)

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

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

---

Instead of repeating these macros and exclude rules in multiple policy files, they can be written to a separate file (for example, `include-file.txt`). The file containing the common macros can be included in other policies. Example 4-14 includes the common macros and exclude rules at the beginning and then defines rules for the migration to tape afterwards.

*Example 4-14 Include common macros defined in file `include-file.txt` in another policy file.*

---

```
/* include file */
include (include-file.txt)
/* will execute EXCLUDE rule from include file */

/* migration policy */
RULE 'extpool' EXTERNAL POOL 'lufs'
  EXEC '/opt/ibm/lufsee/bin/eedm'
  OPTS '-p pool1@lib1' SIZE 10485760

RULE 'ageMig' MIGRATE FROM POOL 'system'
  TO POOL 'lufs' WHERE (NOT is_migrated) and access_age > 10
```

---

Rules are executed in the sequence they appear in the policy file. This means the exclude rule defined in the `include-file.txt` is executed at the beginning of the policy.

**Note:** INCLUDE clauses work, but are not officially supported. Because of this lack of official support, problems associated with INCLUDE clause cannot be addressed with IBM Support.

## 4.4 Threshold-based migration

As explained in sections IBM Storage 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 as shown in Example 4-15 on page 45.

*Example 4-15 Command to apply active policy for a file system*

---

```
# 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 2.4.2, “Migration Callback” on page 20). 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.

#### 4.4.1 Three pool migration with callback

Imagine there are three pools, just like in Figure 1-1 on page 3 (section “IBM Storage Scale file systems and pools”). The automated policy should do the following:

- ▶ When the 'system' pool is occupied at 80% or above, files shall be migrated from the 'system' pool to 'silver' pool.
- ▶ When the 'silver' pool is occupied at 90% or above, files shall be migrated from the 'silver' pool to 'ltfs' pool. The 'ltfs' pool is managed by IBM Storage Archive EE and stores the files in tape pool "Pool1@Lib1".
- ▶ Additionally, all files ending with .jpg, .png, .mp3, .mp4, .mpeg, and .mov should be stored on the 'silver' pool.

The associated automated (or active) policy is shown in Example 4-16.

*Example 4-16 Automated policy with multiple migration and placement rules*

---

```
/* 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/eedm' 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'
```

---

Activate the automated policy the `mmchpolicy` command, using the command shown in Example 4-17.

*Example 4-17 Command to activate the automated policy*

---

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

---

In addition, a callback must be configured. The challenge here is that the parameters for the policy engine (`mmapplypolicy` command) are different when migrating to external pools compared to migration with internal pools (see 2.1, “Running a policy” on page 8). One solution is to adjust the callback script and let it decide upon the parameters for the policy engine based the pool name that triggered the event. The pool name can be passed to the callback script.

To adjust the callback script, use the `mmaddcallback` command as shown in Example 4-18.

*Example 4-18 Adjusting the callback script*

---

```
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 Storage Scale automatically invokes the callback script `/root/silo/callback/mystartpolicy` with the parameters `eventName`, `fsname`, and `storagePool`. The script `mystartpolicy` invokes the `mmapplypolicy` command with the file system name and the parameters derived from the storage pool name within the following script. The command `mmapplypolicy` will execute the policy, applied with `mmchpolicy`.

The script `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 bold **blue** in Example 4-19.

*Example 4-19 Three pool migration: 1. change to mmstartpolicy - assign value to \$pool*

---

```
eventType=$arg1
device=$arg2
deviceName=${device##+(/)dev+(/)} # Name stripped of /dev/ prefix.
#New assign 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 with 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 to facilitate the requirements for IBM Storage Archive EE. The new lines in the callback script following the main body of code comment are shown in bold **blue** in Example 4-20 on page 47.



*Example 4-20 Three pool migration: 2. change to mmstartpolicy - set appropriate options*

---

```
# 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 must specify all relevant IBM Storage 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 Storage Archive EE.

## 4.4.2 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 described in the previous example 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 just with many new files in *Directory3*.

An active migration policy must be implemented to free up space in the file system pools. The priorities for different directories can be honored with different weights provided in the *WEIGHT* clause. 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*.

This *macro directory\_weight* definition can now be used as *WEIGHT* clause in an active migration policy that migrates files from pool 'silver' to 'lifs' if the 'silver' pool is 90% full. See Example 4-21.

*Example 4-21 Migration policy using the macro directory\_weight as WEIGHT clause*

---

```
/* define exclude rule*/
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
```

```

PATH_NAME LIKE '%/.l1tfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* 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)
)

/* here comes the migration rules for silver to l1tfs*/
RULE EXTERNAL POOL 'l1tfs' EXEC '/opt/ibm/l1tfsee/bin/eedm'
OPTS '-p Pool1@Lib1'SIZE 10485760

RULE 'autoMigSilver' MIGRATE FROM POOL 'silver' THRESHOLD(90,70)
WEIGHT(directory_weight) TO POOL 'l1tfs'

/* 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. If the unexpected situation occurs that there are many new files in *Directory3*, then files from *Directory3* will also be migrated to gain space in the file system pool.

## 4.5 Manual or scheduled Migration

This section gives some more examples for scheduled policies (see 2.4.3, “Scheduled migration” on page 21 for more background). Scheduled policies are typically executed using the command `mmapplypolicy` (see 2.1, “Running a policy” on page 8). The parameters of the `mmapplypolicy` command depend on the type of pools. When migrating to an external pool consider the command shown in Example 4-22.

*Example 4-22 Example of mmapplypolicy command when migrating to external pool*

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

---

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

For simplicity the examples described in the following section are scheduled migration policies among internal pools. For external pool migration the external pool definition must be provided in addition (see 2.4.4, “External pools” on page 21).

## 4.5.1 Migrating multiple pools

As shown in Figure 1-1 on page 3 (see “IBM Storage Scale file systems and pools”) there can be multiple storage pools in a file system. Scheduled migrations from one pool to another must be orchestrated to achieve the goals for migration.

Let us 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 has larger storage capacity and files should be migrated to pool 'ltfs' after 60 days. The age of the files in both cases is based on access time. Two migration policies can be configured for this scenario

Example 4-23 shows the first migration policy for migrating files accessed 30 days ago or longer from pool 'system' to 'silver'.

*Example 4-23 Policy for migrating files accessed 30 days ago or longer from pool '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
```

---

Example 4-24 shows the second migration policy for migrating files accessed 60 days ago or longer from pool 'silver' to 'ltfs'.

*Example 4-24 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/eedm'
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 `mmapp1policy` command, whereby the parameters given to this command may differ (see 2.1, “Running a policy” on page 8).

It is not recommended running scheduled policies for different pools of the same file system concurrently because both policies will cause I/O workloads in 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.

It is possible to combine both policy into one policy file. In this case the first migration policy migrating from pool 'system' to pool 'silver' should come first.

The next subsection outlines examples for rules to control schedule migrations.

## 4.5.2 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 is shown in Example 4-25.

*Example 4-25 Basic syntax of GROUP pool rule*

---

```
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 in Example 4-26 can be used.

*Example 4-26 Rebalance rule for GROUP pools*

---

```
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. The GROUP pool rule and the MIGRATE rule for rebalancing must be used together in one policy as shown in Example 4-27.

When combining these two rules in one policy, 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 2.1, “Running a policy” on page 8).

**Note:** Rebalancing does not take place in real time, only when the GROUP policy is executed 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 Example 4-27.

*Example 4-27 GROUP pool policy with internal and external pools*

---

```
/* Exclude rule */
RULE 'exclude' EXCLUDE WHERE
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.1tfsee/%' 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/eedm'
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 Example 4-27 on page 50 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.

### 4.5.3 Migration based on file size

The following migration policy, see Example 4-28, migrates files larger than 10 MB from 'system' to 'silver' pool.

*Example 4-28 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)

```

---

### 4.5.4 Migration based on modification age

The migration policy in Example 4-29 migrates files modified longer than 30 days ago from 'system' to 'silver' pool.

*Example 4-29 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)

```

---

## 4.5.5 Migration based on fileset name

The following migration policy, see Example 4-30, migrates all files stored in fileset “*homefileset*”, larger than 10 MB, and accessed 30 days ago or longer from 'system' pool to 'silver'. Such a policy would typically be used as schedule policy and not as automated policy.

*Example 4-30 Migration policy for fileset based on age and size*

---

```
/* 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 in Example 4-31 (also see 4.1, “Placement policies” on page 40).

*Example 4-31 Example of placement rule to place files in fileset 'homefileset'*

---

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

---

## 4.5.6 Migration in AFM cache filesets

Files in AFM cache fileset can have certain states that do not allow migration to external pools. For this reason, files in dirty or uncached state must be excluded from migration. Dirty files are changed on cache, but not yet transferred to home. Uncached files are not fetched into cache.

Example 4-32 shows a policy that does not migrate dirty or uncached files. It uses the macros for AFM file states (see 4.2.4, “AFM states” on page 42 and the excludes for AFM directories (see 2.4.5, “Excluding files and directories” on page 27).

*Example 4-32 Policy does not migrate dirty or uncached files and uses macros for AFM file states*

---

```
/* 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
PATH_NAME LIKE '%/.afm/%' OR
PATH_NAME LIKE '%/.ptrash/%' OR
PATH_NAME LIKE '%/.pconflicts/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* define macros */
define( is_uncached, ( NOT REGEX(misc_attributes, '[u]') ) )
define( is_dirty, ( REGEX(misc_attributes, '[P]') AND
REGEX(misc_attributes, '[w|v|x|y|j]') ) )

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

RULE 'manMigSilver' MIGRATE FROM POOL 'silver' TO POOL 'ltfs'
WHERE NOT is_dirty AND NOT is_uncached
```

---

The policy in Example 4-32 on page 52 defines an EXCLUDE rule that includes certain AFM metadata paths. The second set of statements defines AFM states for dirty and uncached files. The last set of rules defines an external pool for Storage Archive EE and the migration rule that excludes dirty and uncached files.

## 4.6 Pre-migration and migration

As explained in 2.4.6, “Pre-migration” on page 30 the %high threshold triggers the execution of a threshold-based migration policy. The %low in combination with the %premig thresholds define how many files are pre-migrated. 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 an automated policies is not appropriate because it constantly triggers the policy execution. Therefore, it is recommended to combine scheduled and automated policies for pre-migration.

Let us 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 shown in Example 4-33 will pre-migrate files from pool 'silver' to an external pool managed by IBM Storage Archive EE if the 'silver' pool occupancy is between 50% and 80%.

*Example 4-33 Scheduled policy will pre-migrate files from 'silver' to external based on occupancy*

---

```

/* 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/eedm'
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, see Example 4-34, is invoked with the `mmapplypolicy` command which can be scheduled with the cron-daemon.

*Example 4-34 Example of mmapplypolicy command when migrating to external pool*

---

```
# 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. See Example 4-35.

*Example 4-35 Defining automated migration policy with triggers based on occupancy*

---

```
/* 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/eedm'
OPTS '-p pool1@lib1' SIZE 10485760

/* 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 in Example 4-35 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, shown in Example 4-36, 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).

*Example 4-36 Threshold-based premigration rule*

---

```
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 4.6.1, “Pre-migration using a policy” on page 55). This method relies on a customized script that wraps the `eeadm premigrate` command.

### 4.6.1 Pre-migration using a policy

Files can be pre-migrated using a policy. For this purpose, the command `eeadm migrate -premigrate` is used in the external pool rule.

Example 4-37 shows the policy that pre-migrates all files that are not pre-migrated and are larger than zero bytes. The external pool rule include the parameter `-premigrate` in the OPTS clause. The MIGRATE rule selects all files that are not yet pre-migrated and not empty.

*Example 4-37 Pre-migrate policy using the -premigrate parameter in the OPTS clause*

---

```
/* 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 external pool with -premigrate in the options */
RULE EXTERNAL POOL 'ltrfsPremig' EXEC '/opt/ibm/ltfsee/bin/eeadm'
OPTS '-p test@eelib1 --premigrate' SIZE 10485760

/* Migration rule that selects all files that are not pre-migrated and not empty*/
RULE 'MigToExt' MIGRATE FROM POOL 'system' TO POOL 'ltrfsPremig'
WHERE (NOT (is_empty) AND NOT (is_premigrated))
```

---

To execute the pre-migrate policy the command in Example 4-38 can be used. The pre-migrate policy is stored in file `premig-policy.txt`.

*Example 4-38 Executing the pre-migrate policy*

---

```
mmapplypolicy /mypath -P premig-policy.txt -N eenodes -m 2 -B 1000
--single-instance
```

---

For more information about the parameter of the `mmapplypolicy` command refer to section 2.1.3, “Options for `mmapplypolicy`” on page 10.

## 4.7 String substitution in rules

As explained in 2.1, “Running a policy” on page 8, 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 `"-p Poolname@libraryname"`.

The wrapper script, shown in Example 4-39, uses these parameters to invoke the policy engine with substitution variables.

*Example 4-39 Example of the mmapplypolicy command providing string substitution*

---

```
# 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 substitution variables to migrate all files from pool 'silver' to pool 'ltfs' is shown in Example 4-40.

*Example 4-40 LIST policy with substitution strings POOLVAR and FILESYSVAR*

---

```
/* 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%')  
  
/* External pool rule with POOLVAR being substituted */  
RULE EXTERNAL POOL 'ltfs' EXEC '/opt/ibm/ltfsee/bin/eedm'  
OPTS 'POOLVAR' SIZE 10485760  
  
/* Migration rule with FILESYSVAR being substituted */  
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 required because the file system scope is given with the `mmapplypolicy` command in Example 4-39.

## 4.8 Quota based migration for filesets

The trigger for an automated migration is based on the occupancy of a storage pool as discussed in 2.4.1, “Automated migration” on page 19. In this section we discuss an approach to trigger the migration for a fileset when a fileset quota limit (soft or hard) is 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 a file list on disk (see 4.8.1, “Quota LIST policy” on page 57).
2. A MIGRATE policy takes the list of files generated by the LIST policy step 1, evaluates these against the migration rule and migrates selected files (see 4.8.2, “Quota Migration policy” on page 58).
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 4.8.3, “Quota callback script” on page 59).
4. Finally, 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 4.8.4, “Quota callback” on page 60.

When migrating to an external pool the MIGRATE policy must be executed on a node which has the migration software (for example, Storage Archive Enterprise Edition) installed.

The event `softQuotaExceeded` is raised on one cluster node which is the file system manager for the file system at the point in time when the soft quota is exceeded. If the file system manager node is not identical with the IBM Storage Archive EE node then some commands must be propagated to the appropriate nodes within the callback script (see 4.8.5, “Further considerations” on page 61). Furthermore, the event `softQuotaExceeded` is trigger one time when the soft quota is exceeded.

An alternative approach is to invoke the migration directly from the EXTERNAL LIST policy described in step 1. This however requires adjusting the interface script for migration or to create wrapper-script. This approach is not discussed here.

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

### 4.8.1 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 list.

Example 4-41 on page 58 is a policy example showing 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. Note that the `FOR FILESET` clause expects a fileset name to be passed to this policy via the `mmapplypolicy` command using string substitution (see section 4.7, “String substitution in rules” on page 55).

*Example 4-41 LIST policy example identifying files if the soft quota limit is crossed*

---

```
/* 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 4.8.3, “Quota callback script” on page 59. This LIST policy must be stored in file referenced by the variable `$listpol` in the callback script. The callback script executes the list policy in Example 4-42 with the following command shown in Example 4-42.

*Example 4-42 Command to execute list policy*

---

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

---

This `mmapplypolicy` command produces an output file that is specified by the `-f $outfile` parameter. This output file represents a file list including the file names of files selected by the policy in Example 4-43. This file list needs to be adjusted by extracting the path and filenames of the files using the following command (Example 4-43) in the callback script.

*Example 4-43 Command for extracting path and file name from output file produced by policy*

---

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

---

For more information about extracting the path and file names from an policy engine produced output file see section 4.10.1, “Extracting file names from file list” on page 68.

Also note that we use parameter substitution: the parameter `FSETNAME` in the LIST rule in Example 4-43 is substituted with the actual fileset name given by the `mmapplypolicy` command with the parameter `-M FSETNAME=$fsetName`.

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

## 4.8.2 Quota Migration policy

The migration policy can be invoked with the list of files (see section 2.1.6, “Using existing file lists” on page 14) provided by the list policy (see 4.8.1, “Quota LIST policy” on page 57). When providing a list of files to the policy engine it will not scan the file system metadata but use the files in the file list as input for the selection.

The migration policy in Example 4-28 on page 51 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 migrates from pool 'system' to pool 'lufs'. See Example 4-44 on page 59.

*Example 4-44 MIGRATE policy example to migrate files to an external pool*

---

```
/* 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 '%/.l1tfsee/%' OR
PATH_NAME LIKE '%/.mmSharedTmpDir/%' OR
FILE_NAME like '%.mmbackupShadow%')

/* migration rules with external pool */
RULE EXTERNAL POOL 'l1tfs' EXEC '/opt/ibm/l1tfsee/bin/eedm'
OPTS '-p pool1@lib1' SIZE 10485760

RULE 'quotaMig' MIGRATE FROM POOL 'system' WEIGHT(access_age)
TO POOL 'l1tfs'
FOR FILESET ('FSETNAME') WHERE NOT (is_empty)
```

---

This MIGRATE policy is executed with following command (Example 4-45) in the callback script (see 4.8.3, “Quota callback script” on page 59) This assumes that the MIGRATE policy in Example 4-28 on page 51 is stored in a file referenced by the parameter **\$migpol**.

*Example 4-45 Example for executing MIGRATE policy with a list of files defined in \$outfile.list*

---

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

---

Note, that we are again using 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.

### 4.8.3 Quota callback script

The callback script is invoked by the callback (see 4.8.4, “Quota callback” on page 60) when the event “softQuotaExceeded” was triggered. The callback passes three parameters to the callback script (%eventName as \$1, %fsname as \$2, and %filesetName as \$3).

The callback script first executes the LIST policy that produces an output file including file names to migrated (see 4.8.1, “Quota LIST policy” on page 57). The LIST policy file must be stored in a location denoted by parameter **\$listpol**. Afterwards the callback script executes the MIGRATE policy that migrates the files provided in the output file of the LIST policy (see 4.8.2, “Quota Migration policy” on page 58). The MIGRATE policy file must be stored in a location denoted by parameter **\$migpol**. Example 4-46 on page 60 shows a pseudo code example for this script.

*Example 4-46 Pseudo code for callback script to manage the softQuotaExceeded event*

---

```
# 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 script
evName=$1
fsName=$2
fsetName=$3

# set the name of the output file to $outfile-fsname-fsetname
outfile=$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 the callback can be configured.

## 4.8.4 Quota callback

Once you have tested the policies in section 4.8.1, “Quota LIST policy” on page 57 and 4.8.2, “Quota Migration policy” on page 58 and the script in section 4.8.3, “Quota callback script” on page 59, 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 shown in Example 4-47.

*Example 4-47 Creating the callback for soft-quota migration*

---

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

---

The trigger and execution of the callback can be monitored in the GPFS log (/var/adm/ras/mmfs.log.latest) of node managing the file system.

## 4.8.5 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 is not triggered again. Therefore, it is important to make sure that the callback script works 100% and that it alerts the admin if not.

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.

## 4.9 Generating file lists

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

### 4.9.1 Listing files by HSM state

Example 4-48 shows how the LIST policy identifies files based on the HSM state and writes these lists of files to appropriate output files.

*Example 4-48 LIST policy identifying files based on the HSM state and writes them to 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 shown in Example 4-49.

*Example 4-49 Executing the policy in deferred mode and saving the file list of selected files*

---

```
# 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` that include the files in accordance with the file migration state.

### 4.9.2 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.



Example 4-50 shows how the list policy will list migrated and pre-migrated files, including the migration state and the tape ID.

*Example 4-50 List policy listing migrated and pre-migrated files*

---

```
/* define exclude list */
define( exclude_list,
(PATH_NAME LIKE '%/.SpaceMan/%' OR
PATH_NAME LIKE '%/.snapshots/%' OR
PATH_NAME LIKE '%/.l1tfsee/%' 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 shown in Example 4-51.

*Example 4-51 Executing the list policy identifying pre-migrated and migrated files*

---

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

---

This command, shown in Example 4-52, 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 2.5, “Lists rules” on page 32).

*Example 4-52 Format of the output produced by the LIST policy*

---

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

---

The three first fields are IBM Storage Scale internal numbers (inodenumber, inodegeneration, snapid). The 4<sup>th</sup> field is the file state given with the *SHOW* statement in the policy in Example 4-50. The 6<sup>th</sup> field is the tape ID and the 8<sup>th</sup> field is the file name.

As demonstrated in Example 4-50, with the *SHOW* clause it is possible show more information about a file. In Example 4-50 we show the extended attribute `'dmapi.IBMTPS'` for each file. There are more attributes that can be shown. The command `mm lsattr` allows to list the available attributes as shown in Example 4-53 on page 64.



*Example 4-56 Executing the rule with time stamp substitution*

---

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

---

In Example 4-57 the time stamp LAST is expressed in UNIX epoch format by the **date** command.

*Example 4-57 Format of the time stamp using the UNIX command: date*

---

```
# 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.

## 4.10 Processing file lists

As explained in 2.5, “Lists rules” on page 32 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. Example 4-58 shows an EXTERNAL LIST policy that identifies files accessed within the last day and invokes the interface script:

```
/root/process_youngest.sh.
```

*Example 4-58 EXTERNAL LIST policy processes files with external script 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. See Example 4-59.

*Example 4-59 Executing the External policy*

---

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

---

The interface script will receive 2 or more parameters as described in the following list.

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. Example 4-60 is an example of a list policy result file.

*Example 4-60 Format of the file list produced by the policy engine*

---

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

---

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

Example 4-61 on page 67 is pseudo code for the implementation of the interface script `/root/process_youngest.sh`. Note, based on the string describing the operation in the 1<sup>st</sup>

parameter the 2<sup>nd</sup> parameter might either be the name of the file system (TEST) or the name of the policy result file (LIST).

*Example 4-61 Pseudo code for the implementation of the process\_youngest.sh*

---

```
#!/bin/bash

# function to process the policy results
function process {
    # processes the policy result file given as fName
    return 0
}

# assign parameters received from mmapplypolicy
opCode="$1"; fName="$2"; opts="$3"

# based on op code perform action
rc=0
case $1 in
TEST )
    echo "TEST option received for di-rectory $fName."
    if [[ -d "$fName" ]]; then
        echo "TEST directory $fName exists."
    else
        echo "WARNING: TEST directory $fName does not exists."
        rc=1
    fi;;
LIST )
    echo "LIST option received, starting receiver task"
    if [[ -a "$fName" ]]; then
        # 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
    else
        echo "WARNING: LIST file name $fName does not exists."
        rc=1
    fi;;
* )
    echo "Unknow argument $1 received"
    rc=1;;
esac

exit $rc
```

---

This is just an example. The function “process()” needs to be implemented.

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

## 4.10.1 Extracting file names from file list

As explained in Example 4-61 on page 67 the file lists generated by the policy engine include one file per line in the format shown in Example 4-62.

*Example 4-62 Standard format of file lists produced by the policy engine*

---

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

---

The three first numbers are IBM Storage Scale internal numbers (inodenumber, inodegeneration, snapid). The 5<sup>th</sup> field contains the fully qualified path and file name. The following examples show the file list generated by the policy engine is provided in file `outfile`.

An easy way to extract the 5<sup>th</sup> field with the path and file name is using `awk` command. See Example 4-63.

*Example 4-63 Using the awk command to extract path and file names*

---

```
# cat outfile | awk '{print $5}'
```

---

However, this does not take blanks in the path and file name into account. To accommodate for blanks, it may be suitable to use the `cut` command shown in Example 4-64.

*Example 4-64 Using the cut command to extract path and file names*

---

```
# cat outfile | cut -d' ' -f 7-20
```

---

However, the prior command 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 command can be used. See Example 4-65.

*Example 4-65 Using sophisticated awk command to extract path and file names*

---

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

---

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Accelerating IBM watsonx.data with IBM Fusion HCI*, [REDP-5720](#)
- ▶ *Accelerating AI and Analytics with IBM watsonx.data and IBM Storage Scale*, [REDP-5743](#)
- ▶ *Implementation Guide for IBM Elastic Storage System 3500*, [SG24-8538](#)
- ▶ *IBM Spectrum Scale and IBM Elastic Storage System Network Guide*, [REDP-5484](#)
- ▶ *IBM Storage Scale: Encryption*, [REDP-5707](#)
- ▶ *IBM Spectrum Scale Immutability Introduction, Configuration Guidance, and Use Cases*, [REDP-5507](#)
- ▶ *IBM Storage Scale System Introduction Guide (ESS)*, [REDP-5729](#)

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

[ibm.com/redbooks](https://ibm.com/redbooks)

## Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Storage Scale 5.2 Product Documentation in Guide Format*
- ▶ *IBM TS4500 R9 Tape Library Guide*, [SG24-8235](#)

## Online resources

These websites are also relevant as further information sources:

- ▶ [1] [IBM Storage Scale Documentation Home](#)
- ▶ [2] [Information Lifecycle Management in IBM Storage Scale Documentation](#)
- ▶ [3] [Syntax of the mmapplypolicy command](#)
- ▶ [4] [Whitepaper: Configuring Storage Protect for Space Management \(TSM HSM\) or Storage Archive EE \(IBM Storage Archive EE\) with Storage Scale Active File Management](#)
- ▶ [5] [GitHub repository including sample scripts and policies](#)
- ▶ [6] [File Attributes available to the policy engine](#)

- ▶ [7] [IBM Storage Protect for Space Management with IBM Storage Scale - Configuration Guidance and best practices](#)
- ▶ [8] [Comparison of tape optimized recalls vs. transparent recalls](#)
- ▶ [9] [Best practices for tiering to tape](#)

## Help from IBM

IBM Support and downloads

[ibm.com/support](https://ibm.com/support)

IBM Global Services

[ibm.com/services](https://ibm.com/services)







REDP-5739-00

ISBN 0738461822

Printed in U.S.A.

Get connected

