



# **Storage Management Technical Specification, Part 5 Filesystems**

**Version 1.7.0, Revision 5**

*Abstract: This Working Draft defines an interface between WBEM-capable clients and servers for the secure, extensible, and interoperable management of networked storage.*

Publication of this Working Draft for review and comment has been approved by the SNIA Storage Management Initiative Technical Steering Group (SMI-TSG). This draft represents a "best effort" attempt by the SMI-TSG to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a "work in progress." Suggestion for revision should be directed to <http://www.snia.org/feedback/>.

***Working Draft***

***9 October, 2015***

# REVISION HISTORY

## Revision 1

### Date

8 Sept 2014

### SCRs Incorporated and other changes

None

### Comments

Editorial notes and DRAFT material are displayed.

## Revision 2

### Date

18 December 2014

### SCRs Incorporated and other changes

File Export (SMIS-170-Draft-SCR00004)

- Added mandatory indications for FileShare

File System (SMIS-170-Draft-SCR00004)

- Added mandatory indications for LocalFileSystem

### Comments

Editorial notes and DRAFT material are displayed.

USAGE text was revised to address code.

(now included in the front matter for all SNIA specifications)

## Revision 3

### Date

20 May 2015

### SCRs Incorporated and other changes

All recipes and associated text were removed.

TSG-SMIS-SCR00315.001

- Promoted the maturity level from DRAFT to EXPERIMENTAL: Updated profiles to remove SNIA\_ classes and use DMTF CIM\_ classes

- Revised CIM\_AssociatedPrivilege; Added CIM\_UserContact, CIM\_Identity, CIM\_AccountManagementService, CIM\_AssignedIdentity

File Export Manipulation Profile (TSG-SMIS-SCR00317)

- Added missing figure: FileShares and Simple Identity Management in Section 5.1.3.3.
- Promoted all draft material to Experimental.

### References

- Added DMTF DSP1054 v1.2.2, Indications Profile (and changed version to 1.2.2 throughout book)

Annex A SMI-S Information Model

- Deleted "Most SMI-S Profiles are primarily based on the DMTF Final MOFs" per 5/22/15 TSG meeting consensus.

Several profiles

- Changed LocalAccessAvailable LocalAccessAvailableToFS, to respond to a DMTF change.

#### Comments

- Editorial notes and DRAFT material were hidden.

### Revision 4

#### Date

9 September 2015

#### SCRs Incorporated and other changes

Multiple profiles

- Instances of *subprofile* were changed to *profile*. (TSG meeting voice vote)
- Profile versions and related text were updated. (TSG meeting voice vote)
- *CIM/XML* was changed to *CIM-XML* (Response to ballot comments)
- Removed instances of *Experimental* within profiles already labeled as *Experimental* to avoid confusion and redundancy. (Editorial change)

File Export Manipulation Profile

- Promoted to Stable (TSG-SMIS-SCR00319)
- Changed requirement to Mandatory and description for ProtocolVersions property in Tables 33-36.
- Changed requirement to Mandatory for FileSharingProtocol property in Table 34.
- Added material associated with the DMTF Simple Identity Management Profile (DS1034 rev 1.1.0 as it pertains to ACL manipulation on file shares. (TSG-SMIS-SCR00317)

File Server Manipulation Profile

- Promoted to Stable (TSG-SMIS-SCR00319)

Filesystem Profile

- Material related to ElementCapabilities (naming) incl 8.1.2.3: Promoted to Stable (TSG-SMIS-SCR00319)

Filesystem Manipulation Profile

- Material related to ElementCapabilities (naming) in 9.1.3.1: Promoted to Stable (TSG-SMIS-SCR00319)

NAS Head Profile

- promoted 12.1.3.8.1 to Stable (TSG-SMIS-SCR00319)

References

- Updated reference to DMTF DSP1054 Indications Profile

Self-Contained NAS Profile:

Promoted 13.1.3.1, 13.1.3. 2 (TSG-SMIS-SCR00319)

Annex A (informative) SMI-S Information Model

- DMTF's CIM schema version changed to 2.45.0. (TSG meeting voice vote)

#### Comments

- Editorial notes and DRAFT material were hidden.

## **Revision 5**

### **Date**

9 October 2015

### **SCRs Incorporated and other changes**

Multiple profiles: Addressed SMI-S 1.7.0 Revision 4 TSG ballot comments that were strictly editorial and were approved by voice vote of the TSG.

Filesystem Quotas Profile

- Fixed queries in CIM table 173

References

- Removed DSP0214.
- Removed year from DSP1034

### **Comments**

- Editorial notes were hidden.

Suggestion for changes or modifications to this document should be sent to the SNIA Storage Management Initiative Technical Steering Group (SMI-TSG) at <http://www.snia.org/feedback/>

## USAGE

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

- 1) Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration, and,
- 2) Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing [tcmd@snia.org](mailto:tcmd@snia.org). Please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Copyright (c) 2015, The Storage Networking Industry Association.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of The Storage Networking Industry Association (SNIA) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## **DISCLAIMER**

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to <http://www.snia.org/feedback/>.

Copyright © 2003-2015 SNIA. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners.

Portions of the CIM Schema are used in this document with the permission of the Distributed Management Task Force (DMTF). The CIM classes that are documented have been developed and reviewed by both the SNIA and DMTF Technical Working Groups. However, the schema is still in development and review in the DMTF Working Groups and Technical Committee, and subject to change.

## INTENDED AUDIENCE

This document is intended for use by individuals and companies engaged in developing, deploying, and promoting interoperable multi-vendor SANs through the Storage Networking Industry Association (SNIA) organization.

## CHANGES TO THE SPECIFICATION

Each publication of this specification is uniquely identified by a three-level identifier, comprised of a version number, a release number and an update number. The current identifier for this specification is version 1.7.0. Future publications of this specification are subject to specific constraints on the scope of change that is permissible from one publication to the next and the degree of interoperability and backward compatibility that should be assumed between products designed to different publications of this standard. The SNIA has defined three levels of change to a specification:

- **Major Revision:** A major revision of the specification represents a substantial change to the underlying scope or architecture of the SMI-S API. A major revision results in an increase in the version number of the version identifier (e.g., from version 1.x.x to version 2.x.x). There is no assurance of interoperability or backward compatibility between releases with different version numbers.
- **Minor Revision:** A minor revision of the specification represents a technical change to existing content or an adjustment to the scope of the SMI-S API. A minor revision results in an increase in the release number of the specification's identifier (e.g., from x.1.x to x.2.x). Minor revisions with the same version number preserve interoperability and backward compatibility.
- **Update:** An update to the specification is limited to minor corrections or clarifications of existing specification content. An update will result in an increase in the third component of the release identifier (e.g., from x.x.1 to x.x.2). Updates with the same version and minor release levels preserve interoperability and backward compatibility.

## TYPOGRAPHICAL CONVENTIONS

### Maturity Level

In addition to informative and normative content, this specification includes guidance about the maturity of emerging material that has completed a rigorous design review but has limited implementation in commercial products. This material is clearly delineated as described in the following sections. The typographical convention is intended to provide a sense of the maturity of the affected material, without altering its normative content. By recognizing the relative maturity of different sections of the standard, an implementer should be able to make more informed decisions about the adoption and deployment of different portions of the standard in a commercial product.

This specification has been structured to convey both the formal requirements and assumptions of the SMI-S API and its emerging implementation and deployment lifecycle. Over time, the intent is that all content in the specification will represent a mature and stable design, be verified by extensive implementation experience, assure consistent support for backward compatibility, and rely solely on content material that has reached a similar level of maturity. Unless explicitly labeled with one of the subordinate maturity levels defined for this specification, content is assumed to satisfy these requirements and is referred to as "Finalized". Since much of the evolving specification

content in any given release will not have matured to that level, this specification defines three subordinate levels of implementation maturity that identify important aspects of the content's increasing maturity and stability. Each subordinate maturity level is defined by its level of implementation experience, its stability and its reliance on other emerging standards. Each subordinate maturity level is identified by a unique typographical tagging convention that clearly distinguishes content at one maturity model from content at another level.

## Experimental Maturity Level

No material is included in this specification unless its initial architecture has been completed and reviewed. Some content included in this specification has complete and reviewed design, but lacks implementation experience and the maturity gained through implementation experience. This content is included in order to gain wider review and to gain implementation experience. This material is referred to as “Experimental”. It is presented here as an aid to implementers who are interested in likely future developments within the SMI specification. The contents of an Experimental profile may change as implementation experience is gained. There is a high likelihood that the changed content will be included in an upcoming revision of the specification. Experimental material can advance to a higher maturity level as soon as implementations are available. Figure 1 is a sample of the typographical convention for Experimental content.

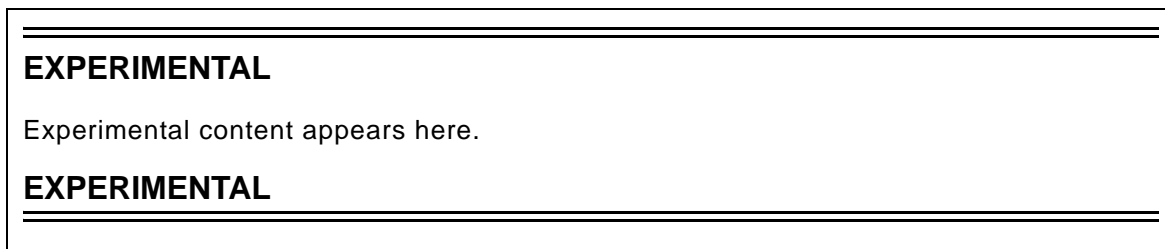


Figure 1 - Experimental Maturity Level Tag

## Implemented Maturity Level

Profiles for which initial implementations have been completed are classified as “Implemented”. This indicates that at least two different vendors have implemented the profile, including at least one provider implementation. At this maturity level, the underlying architecture and modeling are stable, and changes in future revisions will be limited to the correction of deficiencies identified through additional implementation experience. Should the material become obsolete in the future, it must be deprecated in a minor revision of the specification prior to its removal from subsequent releases. Figure 2 is a sample of the typographical convention for Implemented content.

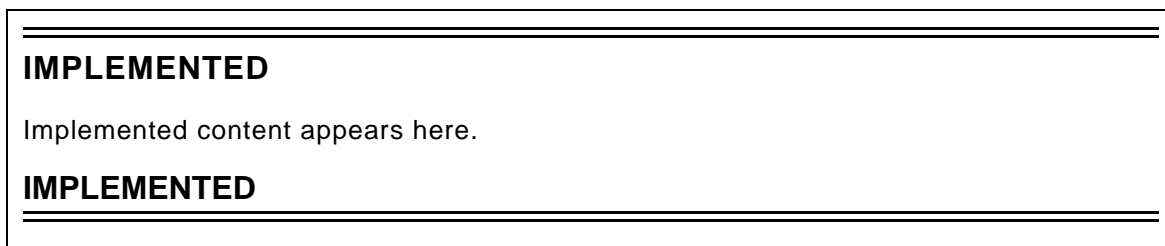


Figure 2 - Implemented Maturity Level Tag

## Stable Maturity Level

Once content at the Implemented maturity level has garnered additional implementation experience, it can be tagged at the Stable maturity level. Material at this maturity level has been implemented by three different vendors, including both a provider and a client. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a minor revision to the specification. Material at this maturity level that has been deprecated may only be removed from the specification as part of a major revision. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. As a result, Profiles at or above the Stable



maturity level shall not rely on any content that is Experimental. Figure 3 is a sample of the typographical convention for Implemented content.

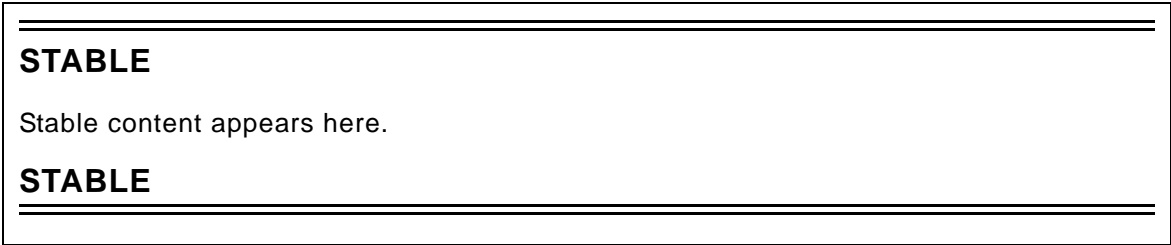


Figure 3 - Stable Maturity Level Tag

**Finalized Maturity Level**

Content that has reached the highest maturity level is referred to as “Finalized.” In addition to satisfying the requirements for the Stable maturity level, content at the Finalized maturity level must solely depend upon or refine material that has also reached the Finalized level. If specification content depends upon material that is not under the control of the SNIA, and therefore not subject to its maturity level definitions, then the external content is evaluated by the SNIA to assure that it has achieved a comparable level of completion, stability, and implementation experience. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a major revision to the specification. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. Over time, it is hoped that all specification content will attain this maturity level. Accordingly, there is no special typographical convention, as there is with the other, subordinate maturity levels. Unless content in the specification is marked with one of the typographical conventions defined for the subordinate maturity levels, it should be assumed to have reached the Finalized maturity level.

**Deprecated Material**

Non-Experimental material can be deprecated in a subsequent revision of the specification. Sections identified as “Deprecated” contain material that is obsolete and not recommended for use in new development efforts. Existing and new implementations may still use this material, but shall move to the newer approach as soon as possible. The maturity level of the material being deprecated determines how long it will continue to appear in the specification. Implemented content shall be retained at least until the next revision of the specialization, while Stable and Finalized material shall be retained until the next major revision of the specification. Providers shall implement the deprecated elements as long as it appears in the specification in order to achieve backward compatibility. Clients may rely on deprecated elements, but are encouraged to use non-deprecated alternatives when possible.

Deprecated sections are documented with a reference to the last published version to include the deprecated section as normative material and to the section in the current specification with the replacement. Figure 4 contains a sample of the typographical convention for deprecated content.

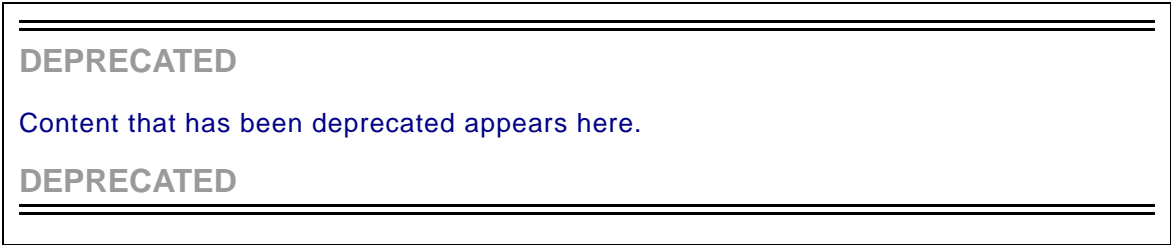


Figure 4 - Deprecated Tag



## Contents

Revision History .....	2
List of Figures .....	15
List of Tables .....	17
Foreword .....	25
1 Scope .....	27
2 Normative References .....	29
2.1 General .....	29
2.2 References under development .....	29
2.3 Other references .....	29
3 Terms, Definitions, Symbols, Abbreviations, and Conventions .....	31
3.1 General .....	31
3.2 Terms and Definitions .....	31
4 File Export Profile .....	33
4.1 Description .....	33
4.2 Health and Fault Management Consideration .....	35
4.3 Cascading Considerations .....	35
4.4 Methods of the Profile .....	35
4.5 Use Cases .....	36
4.6 CIM Elements .....	36
5 File Export Manipulation Profile .....	43
5.1 Description .....	43
5.2 Health and Fault Management Considerations .....	50
5.3 Cascading Considerations .....	51
5.4 Methods of the Profile .....	51
5.5 Use Cases .....	63
5.6 CIM Elements .....	64
6 File Server Manipulation Profile .....	79
6.1 Description .....	79
6.2 Health and Fault Management Consideration .....	84
6.3 Cascading Considerations .....	85
6.4 Methods of the Profile .....	85
6.5 Use Cases .....	93
6.6 CIM Elements .....	94
7 File Storage Profile .....	111
7.1 Description .....	111
7.2 Health and Fault Management Consideration .....	112
7.3 <a href="#">Cascading Considerations</a> .....	112
7.4 Methods of the Profile .....	114
7.5 Client Considerations and Recipes .....	115
7.6 CIM Elements .....	115
8 Filesystem Profile .....	117
8.1 Description .....	117
8.2 Health and Fault Management Consideration .....	120
8.3 Methods of the Profile .....	121
8.4 Use Cases .....	121
8.5 CIM Elements .....	121
9 Filesystem Manipulation Profile .....	135
9.1 Description .....	135

9.2	Health and Fault Management Considerations .....	142
9.3	Methods of the Profile .....	144
9.4	Use Cases.....	162
9.5	CIM Elements.....	163
10	Filesystem Performance Profile.....	187
10.1	Description .....	187
10.2	Implementation.....	188
10.3	Methods of the Profile .....	193
10.4	Use Cases.....	198
10.5	CIM Elements.....	201
11	Filesystem Quotas Profile.....	227
11.1	Description .....	227
11.2	Health and Fault Management Considerations .....	230
11.3	Methods of the Profile .....	230
11.4	Use Cases.....	233
11.5	CIM Elements.....	239
12	NAS Head Profile .....	245
12.1	Description .....	245
12.2	Health and Fault Management Considerations.....	253
12.3	Methods of the Profile .....	254
12.4	Use Cases.....	255
12.5	CIM Elements.....	255
13	Self-Contained NAS Profile .....	263
13.1	Description .....	263
13.2	Health and Fault Management Considerations .....	271
13.3	Cascading Considerations .....	272
13.4	Methods of the Profile .....	272
13.5	Use Cases.....	273
13.6	CIM Elements.....	273
14	NAS Network Port Profile .....	279
14.1	Description .....	279
14.2	Implementation.....	279
14.3	Health and Fault Management Considerations.....	284
14.4	Cascading Considerations .....	285
14.5	Methods .....	285
14.6	Use Cases.....	286
14.7	CIM Elements.....	286
15	Host Filesystem Profile.....	299
15.1	Description .....	299
15.2	Implementation.....	301
15.3	Methods of the Profile .....	304
15.4	Use Cases.....	305
15.5	CIM Elements.....	309
16	Filesystem Replication Services Profile .....	325
16.1	Description .....	325
16.2	Implementation.....	341
16.3	Methods .....	343
16.4	Use Cases.....	374
16.5	CIM Elements.....	375

Annex A (informative) SMI-S Information Model.....	403
Annex B (Informative) State Transitions from Storage to File Shares .....	405



## LIST OF FIGURES

Figure 1 - Experimental Maturity Level Tag .....	8
Figure 2 - Implemented Maturity Level Tag .....	8
Figure 3 - Stable Maturity Level Tag .....	9
Figure 4 - Deprecated Tag .....	9
Figure 5 - File Export Instance .....	34
Figure 6 - File Export Manipulation Profile Instance .....	44
Figure 7 - Capabilities and Settings for Exported File Share Creation.....	47
Figure 8 - FileShares and Simple Identity Management.....	49
Figure 9 - File Server Classes and Associations (Read only view).....	81
Figure 10 - File Server Configuration classes and association .....	83
Figure 11 - File Storage Instance.....	111
Figure 12 - Cascading File Storage .....	113
Figure 13 - Filesystem Instance .....	118
Figure 14 - LocalFileSystem Creation Instance Diagram.....	136
Figure 15 - Capabilities and Settings for Filesystem Creation .....	141
Figure 16 - Filesystem Performance Profile Summary Instance Diagram .....	189
Figure 17 - Filesystem Quotas Instance Diagram.....	230
Figure 18 - NAS Head Profiles.....	247
Figure 19 - NAS Head Instance .....	248
Figure 20 - NAS Storage Instance .....	250
Figure 21 - Self-Contained NAS Profiles .....	265
Figure 22 - Self-Contained NAS Instance .....	266
Figure 23 - NAS Storage Instance .....	268
Figure 24 - NAS Support for Front-end Network Ports .....	280
Figure 25 - Optional NAS TCP Interface Modeling .....	281
Figure 26 - Mandatory NAS Ethernet Port Modeling .....	282
Figure 27 - Host Filesystem Profiles and Package .....	300
Figure 28 - Host Filesystem Instance Diagram.....	301
Figure 29 - Host Filesystem support for Cascading.....	303
Figure 30 - Replication Service Discovery .....	327
Figure 31 - Local File System Replication .....	329
Figure 32 - Remote File System Replication.....	330
Figure 33 - Group Instance Diagram .....	331
Figure 34 - Associated Group and Elements .....	332
Figure 35 - One-to-Many Association .....	333
Figure 36 - Sample CopyState and ProgressStatus Transitions.....	336
Figure 37 - Local Replication with ReplicationEntity .....	337
Figure 38 - Remote replication with ReplicationEntity.....	338
Figure 39 - Multi-Hop Replication .....	339
Figure 40 - SettingDefineState.....	339
Figure 41 - SynchronizationAspect Instance Diagram .....	340
Figure 42 - Filesystem Replication Service support for Cascading .....	342

Figure 43 - Cascading and Replication Groups .....	343
Figure B.1 State Transitions From LogicalDisk to FileShare .....	406



## LIST OF TABLES

Table 1 - Related Profiles for File Export.....	33
Table 2 - FileShare OperationalStatus .....	35
Table 3 - CIM Elements for File Export.....	36
Table 4 - SMI Referenced Properties/Methods for CIM_CIFSShare (Exported File Share).....	37
Table 5 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapa- bilities to FileShare) .....	38
Table 6 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (FileShare) .....	38
Table 7 - SMI Referenced Properties/Methods for CIM_ExportedFileShareSetting (Setting) .....	39
Table 8 - SMI Referenced Properties/Methods for CIM_FileShare (Exported File Share) .....	39
Table 9 - SMI Referenced Properties/Methods for CIM_FileShareSettingData (FileShare).....	40
Table 10 - SMI Referenced Properties/Methods for CIM_HostedShare.....	40
Table 11 - SMI Referenced Properties/Methods for CIM_NFSShare (Exported File Share).....	40
Table 12 - SMI Referenced Properties/Methods for CIM_SAPAvailableForFileShare .....	41
Table 13 - SMI Referenced Properties/Methods for CIM_SharedElement.....	41
Table 14 - Related Profiles for File Export Manipulation .....	43
Table 15 - Operational Status for FileExport Service .....	50
Table 16 - FileExportManipulation Methods .....	51
Table 17 - Parameters for Extrinsic Method ExportedFileShareCapabilities.CreateGoalSettings .....	53
Table 18 - Parameters for Extrinsic Method FileExportService.CreateExportedShare .....	55
Table 19 - Parameters for Extrinsic Method FileExportService.ModifyExportedShare .....	58
Table 20 - Parameters for Extrinsic Method FileExportService.ReleaseExportedShare.....	60
Table 21 - Parameters for Extrinsic Method FileExportService.AssignPrivilegeToExportedShare .....	61
Table 22 - Parameters for Extrinsic Method AccountManagementService.CreateUserContact .....	62
Table 23 - Parameters for Extrinsic Method AccountManagementService.GetUserContacts.....	63
Table 24 - SMI-S File Export Supported Capabilities Patterns.....	63
Table 25 - CIM Elements for File Export Manipulation .....	64
Table 26 - SMI Referenced Properties/Methods for CIM_AccountManagementService.....	66
Table 27 - SMI Referenced Properties/Methods for CIM_AssignedIdentity .....	66
Table 28 - SMI Referenced Properties/Methods for CIM_AssociatedPrivilege .....	66
Table 29 - SMI Referenced Properties/Methods for CIM_CIFSShare (Exported File Share).....	67
Table 30 - SMI Referenced Properties/Methods for CIM_ConcreteDependency .....	68
Table 31 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (FES Configuration) .....	68
Table 32 - SMI Referenced Properties/Methods for CIM_ElementSettingData (FileShare Setting).....	68
Table 33 - SMI Referenced Properties/Methods for CIM_ExportedFileShareCapabilities (FES Capabilities) .....	69
Table 34 - SMI Referenced Properties/Methods for CIM_ExportedFileShareSetting (FileShare Setting).....	70
Table 35 - SMI Referenced Properties/Methods for CIM_ExportedFileShareSetting (Pre-defined).....	71
Table 36 - SMI Referenced Properties/Methods for CIM_FileExportCapabilities (FES Configuration) .....	72
Table 37 - SMI Referenced Properties/Methods for CIM_FileExportService .....	73
Table 38 - SMI Referenced Properties/Methods for CIM_FileShare (Exported File Share) .....	74
Table 39 - SMI Referenced Properties/Methods for CIM_FileStorage (Subelement).....	74
Table 40 - SMI Referenced Properties/Methods for CIM_HostedService .....	75
Table 41 - SMI Referenced Properties/Methods for CIM_HostedShare.....	75
Table 42 - SMI Referenced Properties/Methods for CIM_Identity .....	75
Table 43 - SMI Referenced Properties/Methods for CIM_LogicalFile (Subelement).....	76
Table 44 - SMI Referenced Properties/Methods for CIM_NFSShare (Exported File Share).....	76
Table 45 - SMI Referenced Properties/Methods for CIM_SAPAvailableForFileShare .....	77
Table 46 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement .....	77

Table 47 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (Pre-defined).....	77
Table 48 - SMI Referenced Properties/Methods for CIM_SharedElement.....	78
Table 49 - SMI Referenced Properties/Methods for CIM_UserContact.....	78
Table 50 - Supported Profiles for File Server Manipulation .....	79
Table 51 - Operational Status for File Server ComputerSystem .....	85
Table 52 - Array Element Mappings for TemplateGoalSettings and SupportedGoalSettings .....	86
Table 53 - Parameters for Extrinsic Method FileServerCapabilities.CreateGoalSettings .....	87
Table 54 - Parameters for Extrinsic Method FileServerConfigurationService.CreateFileServer .....	88
Table 55 - Parameters for Extrinsic Method FileServerConfigurationService.ModifyFileServer .....	90
Table 56 - Parameters for Extrinsic Method FileServerConfigurationService.DeleteFileServer.....	91
Table 57 - Parameters for Extrinsic Method FileServerConfigurationService.AddIPInterface.....	91
Table 58 - Parameters for Extrinsic Method FileServerConfigurationService.ModifyIPInterface .....	92
Table 59 - Parameters for Extrinsic Method FileServerConfigurationService.DeleteIPInterface.....	93
Table 60 - CIM Elements for File Server Manipulation .....	94
Table 61 - SMI Referenced Properties/Methods for CIM_CIFSSettingData.....	96
Table 62 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (FileServerSettings to CIFS-SettingData).....	97
Table 63 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (FileServerSettings to DNS-SettingData).....	98
Table 64 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (FileServerSettings to IPInterfaceSettingData).....	98
Table 65 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (FileServerSettings to NFS-SettingData).....	98
Table 66 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (FileServerSettings to NIS-SettingData).....	99
Table 67 - SMI Referenced Properties/Methods for CIM_DNSSettingData .....	99
Table 68 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (FileServerConfigurationService to FileServerCapabilities) .....	99
Table 69 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (FileServerConfigurationService to FileServerConfigurationCapabilities).....	100
Table 70 - SMI Referenced Properties/Methods for CIM_ElementSettingData (ComputerSystem FileServer to FileServerSettings) .....	100
Table 71 - SMI Referenced Properties/Methods for CIM_ElementSettingData (IPInterfaceSettingData to IP-ProtocolEndpoint) .....	100
Table 72 - SMI Referenced Properties/Methods for CIM_FileServerCapabilities.....	101
Table 73 - SMI Referenced Properties/Methods for CIM_FileServerConfigurationCapabilities .....	101
Table 74 - SMI Referenced Properties/Methods for CIM_FileServerConfigurationService .....	102
Table 75 - SMI Referenced Properties/Methods for CIM_FileServerSettings .....	103
Table 76 - SMI Referenced Properties/Methods for CIM_HostedDependency .....	104
Table 77 - SMI Referenced Properties/Methods for CIM_HostedService (Hosting Computer System to FileServerConfigurationService) .....	104
Table 78 - SMI Referenced Properties/Methods for CIM_IPInterfaceSettingData .....	104
Table 79 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.).....	105
Table 80 - SMI Referenced Properties/Methods for CIM_NetworkVLAN .....	106
Table 81 - SMI Referenced Properties/Methods for CIM_NFSSettingData.....	106
Table 82 - SMI Referenced Properties/Methods for CIM_NISSettingData .....	107
Table 83 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (CIFSSettingData).....	107
Table 84 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (DNSSettingData) .....	107
Table 85 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (FileServerSettings).....	108
Table 86 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (IPInterfaceSettingData) .....	108

Table 87 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (NFSSettingData) .....	108
Table 88 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (NISSettingData) .....	109
Table 89 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (ComputerSystem FileServer to FileServerSettings) .....	109
Table 90 - Cascaded Storage .....	114
Table 91 - CIM Elements for File Storage .....	115
Table 92 - SMI Referenced Properties/Methods for CIM_ResidesOnExtent .....	115
Table 93 - Related Profiles for Filesystem .....	117
Table 94 - Filesystem OperationalStatus .....	120
Table 95 - CIM Elements for Filesystem .....	121
Table 96 - SMI Referenced Properties/Methods for CIM_Dependency (Uses Directory Services From) .....	123
Table 97 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to LocalFileSystem) .....	124
Table 98 - SMI Referenced Properties/Methods for CIM_ElementSettingData (FileSystem) .....	124
Table 99 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Local Access Required) .....	124
Table 100 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (LocalFileSystem) .....	125
Table 101 - SMI Referenced Properties/Methods for CIM_FileStorage .....	125
Table 102 - SMI Referenced Properties/Methods for CIM_FileSystemSetting .....	126
Table 103 - SMI Referenced Properties/Methods for CIM_HostedDependency (Local Access Required) .....	127
Table 104 - SMI Referenced Properties/Methods for CIM_HostedFileSystem (LocalFileSystem) .....	127
Table 105 - SMI Referenced Properties/Methods for CIM_LocalAccessAvailableToFS .....	128
Table 106 - SMI Referenced Properties/Methods for CIM_LocallyAccessibleFileSystemSetting .....	128
Table 107 - SMI Referenced Properties/Methods for CIM_LocalFileSystem .....	131
Table 108 - SMI Referenced Properties/Methods for CIM_LogicalFile .....	134
Table 109 - Related Profiles for Filesystem Manipulation .....	135
Table 110 - LocalFileSystem OperationalStatus .....	142
Table 111 - Filesystem Manipulation Methods that cause Instance Creation, Deletion or Modification .....	144
Table 112 - Parameters for Extrinsic Method FileSystemCapabilities.CreateGoalSettings .....	146
Table 113 - Parameters for Extrinsic Method FileSystemCapabilities.GetRequiredStorageSize .....	147
Table 114 - Parameters for Extrinsic Method LocallyAccessibleFileSystemCapabilities.CreateGoalSettings .....	150
Table 115 - Parameters for Extrinsic Method FileSystemConfigurationService.CreateFileSystem .....	153
Table 116 - Parameters for Extrinsic Method FileSystemConfigurationService.ModifyFileSystem .....	158
Table 117 - Parameters for Extrinsic Method FileSystemConfigurationService.DeleteFileSystem .....	161
Table 118 - Filesystem Manipulation Supported Capabilities Patterns .....	162
Table 119 - CIM Elements for Filesystem Manipulation .....	163
Table 120 - SMI Referenced Properties/Methods for CIM_Dependency (Uses Directory Services From) .....	166
Table 121 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (FS Configuration Capabilities) .....	167
Table 122 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Local Access Configuration Capabilities) .....	167
Table 123 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Default) .....	167
Table 124 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Non-Default) .....	168
Table 125 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Attached to Filesystem) .....	168
Table 126 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Local Access Required) .....	168
Table 127 - SMI Referenced Properties/Methods for CIM_FileSystemCapabilities .....	169
Table 128 - SMI Referenced Properties/Methods for CIM_FileSystemConfigurationCapabilities .....	169
Table 129 - SMI Referenced Properties/Methods for CIM_FileSystemConfigurationService .....	172
Table 130 - SMI Referenced Properties/Methods for CIM_FileSystemSetting (Attached to FileSystem) .....	173
Table 131 - SMI Referenced Properties/Methods for CIM_FileSystemSetting (Predefined FS Settings) .....	175

Table 132 - SMI Referenced Properties/Methods for CIM_HostedDependency (Attached to File System).....	176
Table 133 - SMI Referenced Properties/Methods for CIM_HostedDependency (Predefined Capabilities).....	176
Table 134 - SMI Referenced Properties/Methods for CIM_HostedDependency (Predefined Setting) .....	177
Table 135 - SMI Referenced Properties/Methods for CIM_HostedFileSystem.....	177
Table 136 - SMI Referenced Properties/Methods for CIM_HostedService .....	177
Table 137 - SMI Referenced Properties/Methods for CIM_LocalAccessAvailableToFS .....	178
Table 138 - SMI Referenced Properties/Methods for CIM_LocalFileSystem .....	178
Table 139 - SMI Referenced Properties/Methods for CIM_LocallyAccessibleFileSystemCapabilities .....	180
Table 140 - SMI Referenced Properties/Methods for CIM_LocallyAccessibleFileSystemSetting .....	182
Table 141 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (Predefined FS Settings).....	185
Table 142 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities (Predefined Local Access Settings).....	186
Table 143 - Related Profiles for Filesystem Performance .....	187
Table 144 - Summary of Element Types by Profile .....	190
Table 145 - Creation, Deletion and Modification Methods in the Filesystem Performance Profile .....	193
Table 146 - Summary of Statistics Support by Element .....	198
Table 147 - Formulas and Calculations - Calculated Statistics for a Time Interval.....	200
Table 148 - Filesystem Performance Profile Supported Capabilities Patterns .....	201
Table 149 - CIM Elements for Filesystem Performance .....	201
Table 150 - SMI Referenced Properties/Methods for CIM_AssociatedFileSystemStatisticsManifestCollection (Client defined collection) .....	204
Table 151 - SMI Referenced Properties/Methods for CIM_AssociatedFileSystemStatisticsManifestCollection (Provider defined collection) .....	204
Table 152 - SMI Referenced Properties/Methods for CIM_ElementCapabilities .....	205
Table 153 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Exported File Share Stats) .....	205
Table 154 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Exporting Port Stats) .....	206
Table 155 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Local Filesystem Stats) .....	206
Table 156 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (OTHER Element Type Stats) .....	207
Table 157 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticalData .....	207
Table 158 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticsCapabilities .....	213
Table 159 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticsManifest (Client Defined) .....	214
Table 160 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticsManifest (Provider Support).....	217
Table 161 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticsManifestCollection (Client Defined).....	219
Table 162 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticsManifestCollection (Provider Defined) .....	220
Table 163 - SMI Referenced Properties/Methods for CIM_FileSystemStatisticsService.....	220
Table 164 - SMI Referenced Properties/Methods for CIM_HostedCollection (Client Defined).....	222
Table 165 - SMI Referenced Properties/Methods for CIM_HostedCollection (Default).....	222
Table 166 - SMI Referenced Properties/Methods for CIM_HostedCollection (Provider Supplied).....	223
Table 167 - SMI Referenced Properties/Methods for CIM_HostedService .....	223
Table 168 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of client defined collection) .....	224
Table 169 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of predefined collection) .....	224
Table 170 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of statistics collection).....	224
Table 171 - SMI Referenced Properties/Methods for CIM_StatisticsCollection.....	225

Table 172 - Related Profiles for FileSystem Quotas.....	227
Table 173 - CIM Elements for FileSystem Quotas.....	239
Table 174 - SMI Referenced Properties/Methods for CIM_FSDomainIdentity .....	240
Table 175 - SMI Referenced Properties/Methods for CIM_FSQuotaAppliesToElement .....	240
Table 176 - SMI Referenced Properties/Methods for CIM_FSQuotaAppliesToPrincipal.....	240
Table 177 - SMI Referenced Properties/Methods for CIM_FSQuotaAppliesToTree.....	241
Table 178 - SMI Referenced Properties/Methods for CIM_FSQuotaCapabilities.....	241
Table 179 - SMI Referenced Properties/Methods for CIM_FSQuotaConfigEntry.....	242
Table 180 - SMI Referenced Properties/Methods for CIM_FSQuotaIndication .....	242
Table 181 - SMI Referenced Properties/Methods for CIM_FSQuotaManagementService .....	243
Table 182 - SMI Referenced Properties/Methods for CIM_FSQuotaReportRecord .....	243
Table 183 - Related Profiles for NAS Head .....	245
Table 184 - InstModification Events for ComputerSystem.....	251
Table 185 - InstModification Events for LogicalDisk .....	252
Table 186 - Bellwether AlertIndication Events for ComputerSystem .....	252
Table 187 - Bellwether AlertIndication Events for LogicalDisk.....	253
Table 188 - Standard Messages used by NAS Head .....	254
Table 189 - CIM Elements for NAS Head.....	255
Table 190 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System).....	257
Table 191 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Virtual File Server) .....	257
Table 192 - SMI Referenced Properties/Methods for CIM_ConcreteComponent.....	258
Table 193 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (ImplementationCapabilities to Service) .....	258
Table 194 - SMI Referenced Properties/Methods for CIM_HostedDependency .....	258
Table 195 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapabilities) .....	259
Table 196 - SMI Referenced Properties/Methods for CIM_LogicalDisk (LD for FS).....	259
Table 197 - SMI Referenced Properties/Methods for CIM_StorageExtent (Primordial Imported Extent).....	260
Table 198 - SMI Referenced Properties/Methods for CIM_SystemDevice (Logical Disks) .....	261
Table 199 - SMI Referenced Properties/Methods for CIM_SystemDevice (Storage Extents).....	261
Table 200 - Related Profiles for Self-contained NAS System.....	263
Table 201 - InstModification Events for ComputerSystem.....	269
Table 202 - InstModification Events for LogicalDisk .....	270
Table 203 - Bellwether AlertIndication Events for ComputerSystem .....	271
Table 204 - Bellwether AlertIndication Events for LogicalDisk.....	271
Table 205 - Standard Messages used by NAS Head .....	272
Table 206 - CIM Elements for Self-contained NAS System.....	273
Table 207 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System).....	275
Table 208 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Virtual File Server) .....	275
Table 209 - SMI Referenced Properties/Methods for CIM_HostedDependency .....	276
Table 210 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapabilities) .....	276
Table 211 - SMI Referenced Properties/Methods for CIM_LogicalDisk (Disk for FS) .....	277
Table 212 - SMI Referenced Properties/Methods for CIM_SystemDevice (Logical Disks) .....	277
Table 213 - Related Profiles for NAS Network Port.....	279
Table 214 - InstModification Events for NetworkPort.....	283
Table 215 - InstModification Events for ProtocolEndpoint .....	284
Table 216 - Bellwether AlertIndication Events for NetworkPort .....	284
Table 217 - NetworkPort OperationalStatus .....	285
Table 218 - ProtocolEndpoint OperationalStatus.....	285

Table 219 - Standard Messages used by NAS Head .....	285
Table 220 - CIM Elements for NAS Network Port.....	286
Table 221 - SMI Referenced Properties/Methods for CIM_BindsTo (CIFS or NFS).....	288
Table 222 - SMI Referenced Properties/Methods for CIM_BindsTo (TCP) .....	288
Table 223 - SMI Referenced Properties/Methods for CIM_BindsToLANEndpoint .....	289
Table 224 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (CIFS or NFS to Net- workPort) .....	289
Table 225 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (LANEndpoint to Net- workPort) .....	289
Table 226 - SMI Referenced Properties/Methods for CIM_ElementSettingData (IPInterfaceSettingData to IP- ProtocolEndpoint) .....	290
Table 227 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (CIFS or NFS).....	290
Table 228 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (IP).....	290
Table 229 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (LAN) .....	291
Table 230 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (TCP) .....	291
Table 231 - SMI Referenced Properties/Methods for CIM_IPInterfaceSettingData.....	291
Table 232 - SMI Referenced Properties/Methods for CIM_IPProtocolEndpoint .....	292
Table 233 - SMI Referenced Properties/Methods for CIM_LANEndpoint.....	293
Table 234 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.).....	294
Table 235 - SMI Referenced Properties/Methods for CIM_NetworkPort.....	294
Table 236 - SMI Referenced Properties/Methods for CIM_NetworkVLAN .....	295
Table 237 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint (CIFS or NFS) .....	296
Table 238 - SMI Referenced Properties/Methods for CIM_SystemDevice (Network Ports).....	296
Table 239 - SMI Referenced Properties/Methods for CIM_TCPProtocolEndpoint .....	297
Table 240 - Related Profiles for Host Filesystem.....	299
Table 241 - Discovery of the Filesystem Volumes.....	305
Table 242 - Expansion of a Filesystem.....	306
Table 243 - Replication of a Filesystem.....	306
Table 244 - Quiesce a Filesystem .....	307
Table 245 - Unquiesce a Filesystem.....	307
Table 246 - Filesystem quiesce timeout .....	308
Table 247 - Retrieve File Information.....	308
Table 248 - CIM Elements for Host Filesystem .....	309
Table 249 - SMI Referenced Properties/Methods for CIM_AllocatedResources.....	311
Table 250 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Shadow).....	312
Table 251 - SMI Referenced Properties/Methods for CIM_Dependency (Systems) .....	312
Table 252 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (FS Configuration Capabili- ties) .....	313
Table 253 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (ImplementationCapabilities to Service) .....	313
Table 254 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (FilesystemConfigura- tionService to Host Filesystem RegisteredProfile) .....	313
Table 255 - SMI Referenced Properties/Methods for CIM_FileSystemConfigurationCapabilities .....	314
Table 256 - SMI Referenced Properties/Methods for CIM_FileSystemConfigurationService.....	315
Table 257 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources) .....	315
Table 258 - SMI Referenced Properties/Methods for CIM_HostedService .....	316
Table 259 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapa- bilities) .....	316
Table 260 - SMI Referenced Properties/Methods for CIM_LocalFileSystem .....	316
Table 261 - SMI Referenced Properties/Methods for CIM_LogicalDisk (Shadow) .....	318

Table 262 - SMI Referenced Properties/Methods for CIM_LogicalFile.....	319
Table 263 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (LogicalDisk) .....	320
Table 264 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources).....	320
Table 265 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources).....	320
Table 266 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint (Shadow) .....	321
Table 267 - SMI Referenced Properties/Methods for CIM_RemoteResources .....	321
Table 268 - SMI Referenced Properties/Methods for CIM_ResidesOnExtent.....	322
Table 269 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement .....	322
Table 270 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement.....	322
Table 271 - SMI Referenced Properties/Methods for CIM_StorageExtent (Primordial Imported Extent) .....	323
Table 272 - SMI Referenced Properties/Methods for CIM_SystemDevice (LogicalDisks) .....	323
Table 273 - Related Profiles for Filesystem Replication Services .....	325
Table 274 - Key Components .....	326
Table 275 - Comparing SyncTypes .....	328
Table 276 - CopyStatus Values .....	334
Table 277 - Indications .....	341
Table 278 - Extrinsic Method for Group Management.....	344
Table 279 - Extrinsic Method for Replication Management .....	344
Table 280 - Extrinsic Method for Getting Supported Capabilities .....	345
Table 281 - Selected CreateElementReplica optional parameters .....	349
Table 282 - Selected CreateGroupReplica optional parameters .....	351
Table 283 - Selected CreateListReplica optional parameters.....	353
Table 284 - SyncTypes .....	363
Table 285 - Mode.....	363
Table 286 - Locality .....	363
Table 287 - ReplicationTypes .....	363
Table 288 - Features .....	364
Table 289 - Group Features.....	366
Table 290 - Consistency .....	367
Table 291 - Operations .....	367
Table 292 - Comparison of Similar Operations.....	369
Table 293 - SettingsDefineState Operations .....	370
Table 294 - Thin Provisioning Features.....	370
Table 295 - Components .....	371
Table 296 - Default Consistency.....	371
Table 297 - Default Group Persistency.....	371
Table 298 - Copy Methodologies.....	372
Table 299 - Target Element Suppliers .....	372
Table 300 - ThinProvisioningPolicy.....	373
Table 301 - Connection Features .....	373
Table 302 - Storage Compression Features.....	374
Table 303 - CIM Elements for Filesystem Replication Services .....	375
Table 304 - SMI Referenced Properties/Methods for CIM_AllocatedResources.....	379
Table 305 - SMI Referenced Properties/Methods for CIM_ElementCapabilities .....	379
Table 306 - SMI Referenced Properties/Methods for CIM_FileSystemReplicationServiceCapabilities.....	380
Table 307 - SMI Referenced Properties/Methods for CIM_FileSystemSynchronized .....	382
Table 308 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ForProtocolEndpoint) .....	385
Table 309 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ForRemoteServiceAccess- Point) .....	386

Table 310 - SMI Referenced Properties/Methods for CIM_HostedCollection (Allocated Resources) .....	386
Table 311 - SMI Referenced Properties/Methods for CIM_HostedCollection (Between ComputerSystem and RemoteReplicationCollection) .....	386
Table 312 - SMI Referenced Properties/Methods for CIM_HostedCollection (Between ComputerSystem and ReplicationGroup).....	387
Table 313 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources) .....	387
Table 314 - SMI Referenced Properties/Methods for CIM_HostedService .....	388
Table 315 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources).....	388
Table 316 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection) .....	388
Table 317 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources).....	389
Table 318 - SMI Referenced Properties/Methods for CIM_OrderedMemberOfCollection.....	389
Table 319 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint .....	389
Table 320 - SMI Referenced Properties/Methods for CIM_RemoteReplicationCollection.....	390
Table 321 - SMI Referenced Properties/Methods for CIM_RemoteResources .....	391
Table 322 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint .....	391
Table 323 - SMI Referenced Properties/Methods for CIM_ReplicaPoolForStorage.....	392
Table 324 - SMI Referenced Properties/Methods for CIM_ReplicationEntity .....	393
Table 325 - SMI Referenced Properties/Methods for CIM_ReplicationGroup .....	393
Table 326 - SMI Referenced Properties/Methods for CIM_ReplicationService .....	394
Table 327 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData .....	395
Table 328 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement .....	398
Table 329 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection).....	398
Table 330 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity) .....	398
Table 331 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup) .....	399
Table 332 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect).....	399
Table 333 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (Between storage object and SynchronizationAspect) .....	400
Table 334 - SMI Referenced Properties/Methods for CIM_SharedSecret.....	400
Table 335 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect .....	401



## FOREWORD

The Filesystems part of the *Storage Management Technical Specification* contains Profiles and other clauses for management of devices and programs that support filesystems. A filesystem is a specific formatting of storage for storing and accessing files on external storage. This part describes how filesystems are created, modified and deleted, as well as how they can be found and reported. This part also describe modeling for how filesystems are exported for access from remote systems. The filesystem profiles use information from other parts of the Storage Management Technical Specifications. Specifically, they reference profiles in the Common Profiles and the Block Devices parts of the specification. This part describes how these profiles are used in filesystem profiles.

### Parts of this Standard

This standard is subdivided in the following parts:

- *Storage Management Technical Specification, Part 1 Overview, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 5 Filesystems, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 6 Fabric, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 7 Host Elements, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 8 Media Libraries, 1.7.0 Rev 5*

### SNIA Web Site

Current SNIA practice is to make updates and other information available through their web site at <http://www.snia.org>

### SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at <http://www.snia.org/feedback/> or by mail to the Storage Networking Industry Association, 4360 ArrowsWest Drive, Colorado Springs, Colorado 80907, U.S.A.



## 1 Scope

The Filesystems part of the *Storage Management Technical Specification* defines management profiles for Autonomous (top level) profiles for programs and devices whose central function is providing support and access to file data. In addition, it provides documentation of component profiles that deal with filesystems and management interface functions that may be used by other autonomous profiles not included in this part of the specification.

There is an informative annex that describes how storage is mapped from block storage to file shares exported by the filesystem and the mechanisms involved in that establishing those mappings. This annex is recommended for getting an overview of how the filesystem models work.

This version of the Filesystems part of the Storage Management Technical Specification includes two autonomous profiles:

- The NAS Head Profile

This profile defines the model and functions of a NAS device that exports file shares to remote users and gets its storage from a SAN (array devices attached to the NAS Head device).

- The Self-Contained NAS Profile

This profile defines the model and functions of a NAS device that exports file shares to remote users, but gets its storage from disk drives that are internal to the NAS device (instead of externally attached arrays).

In addition to these autonomous profiles, this part of the specification defines a number of component profiles, which are used by the autonomous NAS profiles and might also be used by other autonomous profiles that feature filesystem elements and services. The component profiles defined in this version of the specification include:

- The File Export (component) Profile

This component profile defines the elements used to model the exporting of filesystems or directories for any autonomous profile that exports file data to remote systems.

- The File Export Manipulation (component) Profile

This component profile defines the elements used to model the services for creating, modifying and deleting the file shares (the representation of exported filesystems or directories) for any autonomous profile that provides manipulation of exported filesystems or directories.

- The File Storage (component) Profile

This component profile defines the elements used to model the storage of filesystems on logical disks. This profile does not have services for maintaining the mapping of filesystems to logical disks. These services are addressed in the Filesystem Manipulation Profile.

- The Filesystem (component) Profile

This component profile defines the elements used to model filesystems and its related elements, such as logical files, directories and information on how the filesystem is addressed when mounted to a specific system. The services for defining and maintaining the information in the Filesystem Profile are contained in the Filesystem Manipulation Profile.

- The Filesystem Manipulation (component) Profile

This component profile defines the elements used to model the services for creating, modifying and deleting filesystems and their related elements.

- The Filesystem Quotas (component) Profile

This component profile defines the elements used to model the elements associated with creating, maintaining and reporting on quotas on various filesystem elements.

## 2 Normative References

### 2.1 General

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

### 2.2 References under development

*Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*

*Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*

*Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*

### 2.3 Other references

DMTF DSP1034 Simple Identity Management Profile 1.1.0

[http://dmtof.org/sites/default/files/standards/documents/DSP1034\\_1.1.0.pdf](http://dmtof.org/sites/default/files/standards/documents/DSP1034_1.1.0.pdf)

DMTF DSP1054 Indications Profile 1.2.2

[http://www.dmtf.org/sites/default/files/standards/documents/DSP1054\\_1.2.2.pdf](http://www.dmtf.org/sites/default/files/standards/documents/DSP1054_1.2.2.pdf)

MS-SMB2: Server Message Block (SMB) Protocol Versions 2 and 3

<https://msdn.microsoft.com/en-us/library/cc246482.aspx>

IETF ISSN: 2070-1721, Network File System (NFS) Version 4 Minor Version 1 Protocol,

<https://tools.ietf.org/html/rfc5661>

ISO/IEC 14776-452, SCSI Primary Commands - 3 (SPC-3) [ANSI INCITS.351-2005]



## 3 Terms, Definitions, Symbols, Abbreviations, and Conventions

### 3.1 General

For the purposes of this document, the terms, definitions, symbols, abbreviations, and conventions given in *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5* and the following apply.

### 3.2 Terms and Definitions

#### 3.2.1

##### **CIFS**

Common Internet File System

#### 3.2.2

##### **Directory**

A subtree within a filesystem

A directory may contain files or other directories.

#### 3.2.3

##### **File**

A logical file in a filesystem

#### 3.2.4

##### **file server**

a system configuration that supports the exporting of files and files systems

Note 1 to entry: A file server may be a virtual system element.

#### 3.2.5

##### **file share**

sharing protocols applied to a directory. A directory is exported to remote users through a file share

#### 3.2.6

##### **filesystem**

a filesystem in which files are named and placed logically for storage and retrieval

#### 3.2.7

##### **FS quota**

a quota (hard or soft limit) placed on filesystem resource usage

#### 3.2.8

##### **logical disk**

block storage on which filesystems are built

Note 1 to entry: A logical disk would be formatted for a particular filesystem.

#### 3.2.9

##### **NAS**

Network Attached Storage

In the context of this specification this refers to devices that serve files to a network

#### 3.2.10

##### **NAS head**

a NAS device that gets its physical storage from one or more arrays that are externally attached to the NAS device

### **3.2.11**

#### **NFS**

Network File System

### **3.2.12**

#### **Self-Contained NAS**

a NAS device that has its own internal (to the NAS device) storage

### **3.2.13**

#### **quota**

a hard or soft limit defined for users, user groups or resource collections on the amount of resources that may be consumed



---



---

**STABLE**
**4 File Export Profile****4.1 Description****4.1.1 Synopsis****Profile Name:** File Export (Component Profile)**Version:** 1.7.0**Organization:** SNIA**Central Class:** FileShare**Scoping Class:** ComputerSystem**Related Profiles:** Table 336 describes the related profiles for File Export.**Table 336 - Related Profiles for File Export**

Profile Name	Organization	Version	Requirement	Description
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

**4.1.2 Overview**

The File Export Profile is a component profile for autonomous profiles that support exporting of filesystems. Specifically, in this release of SMI-S, this includes the NAS Head and Self-Contained NAS Profiles. In some of these autonomous profiles the File Export is required. In others it may not be. See the parent profile to see if this profile is required or not.

**4.1.3 Implementation**

Figure 5: "File Export Instance" illustrates the classes mandatory for modeling the export of File Shares for the filesystem profiles. This profile is supported by the Self-contained NAS and the NAS Head Profiles. Figure 5 shows the ComputerSystem that hosts the LocalFileSystem ("filesystem host") as different from the ComputerSystem hosting the FileShare ("File server"). While they may be different ComputerSystems, they may also be the same ComputerSystem instance.

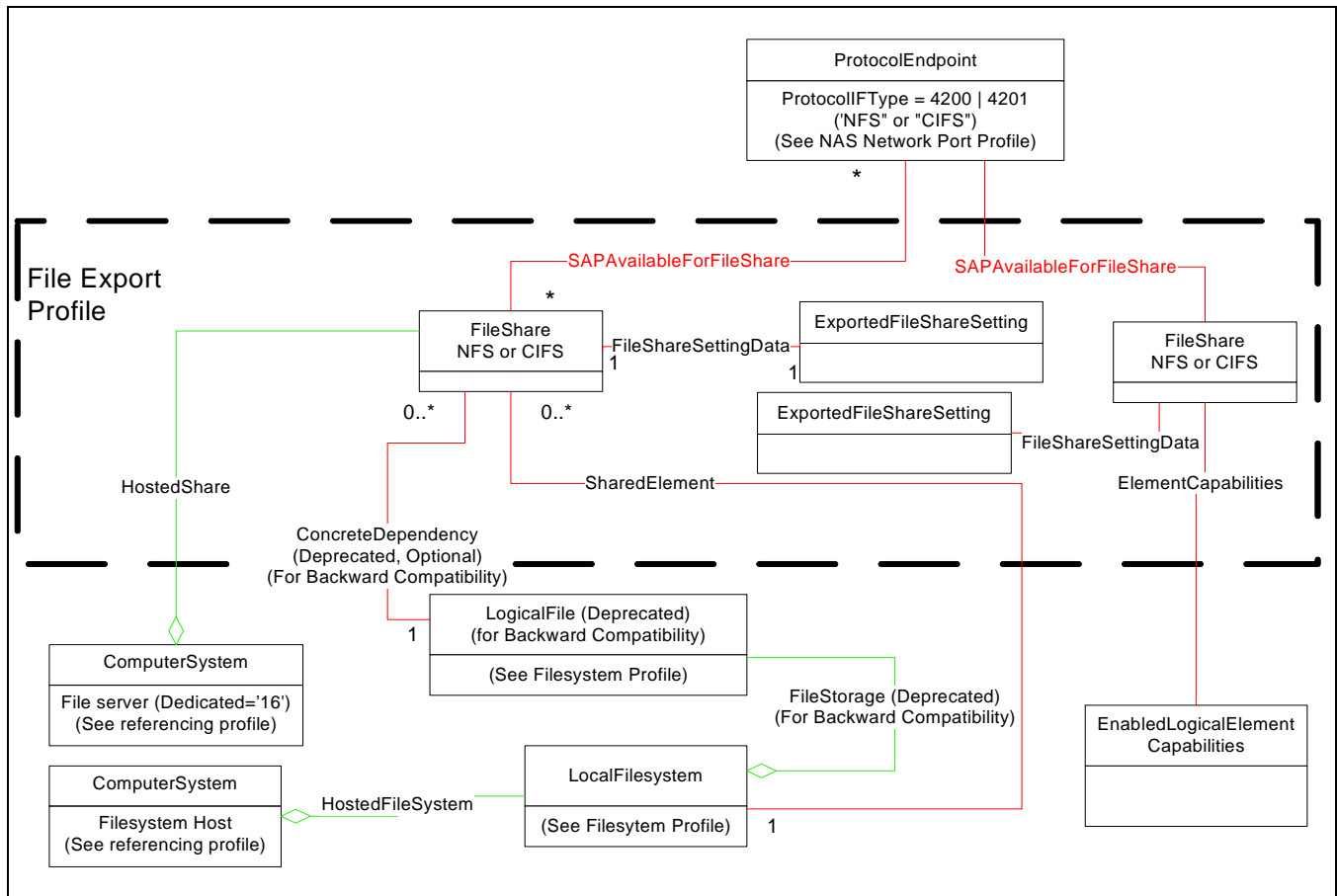


Figure 5 - File Export Instance

The referencing profile shall model any File Shares that have been exported to the network. A File Share shall be represented as a FileShare instance with associations to the ComputerSystem that hosts the share (via HostedShare), to the ExportedFileShareSetting (via FileShareSettingData) and to the ProtocolEndpoint (via SAPAvailableForFileShare) through which the Share can be accessed.

NOTE In Figure 5 the FileShare shown is intended to represent a subclass of CIM\_FileShare (e.g., CIFSShare or NFSShare). It is not intended to imply that either should be represented by CIM\_FileShare (which does not indicate the type of file share).

## EXPERIMENTAL

The FileShare also has a SharedElement association to the LocalFilesystem on which the share is based.

## EXPERIMENTAL

The FileShare may also have an ElementCapabilities association to an EnabledLogicalUnitCapabilities to identify naming and requested state change capabilities.

#### 4.1.3.1 Associations to FileShare

The SAPAvailableForFileShare is a many to many association. That is, multiple FileShares may be exported through the same ProtocolEndpoint and multiple ProtocolEndpoints may support the same FileShare (CIFSShare or NFSShare).

The SharedElement association between the FileShare (CIFSShare or NFSShare) and a LocalFileSystem is many to one association. Zero or more FileShares may be associated to one LocalFileSystem. But each FileShare (CIFSShare or NFSShare) shall only reference one LocalFileSystem.

The FileShareSettingData association between the FileShare (CIFSShare or NFSShare) and the ExportedFileShareSetting is a one to one association. That is, a FileShare (CIFSShare or NFSShare) shall have an ExportedFileShareSetting and that ExportedFileShareSetting shall be associated to exactly one FileShare (CIFSShare or NFSShare).

#### 4.1.3.2 Element Naming

The name of a FileShare may be changed. The existence of the EnabledLogicalElementCapabilities instance associated to the FileShare indicates that the FileShare can be named. If ElementNameEditSupported is set to TRUE, then the ElementName of the associated FileShare may be modified. The ElementNameMask property provides the regular expression that expresses the limits of the name; see 4.7.x for the class definition for EnabledLogicalElementCapabilities for details for this property.

### 4.2 Health and Fault Management Consideration

The File Export Profile supports state information (e.g., OperationalStatus) on the following element of the model:

- FileShares that are exported (See section 4.2.1)

#### 4.2.1 OperationalStatus for FileShares

Table 2 shows FileShare operationalStatus.

**Table 2 - FileShare OperationalStatus**

OperationalStatus	Description
OK	FileShare is online
Error	FileShare has a failure. This could be due to a filesystem failure.
Stopped	FileShare is disabled
Unknown	

### 4.3 Cascading Considerations

Not defined in this standard

### 4.4 Methods of the Profile

#### 4.4.1 Extrinsic Methods of the Profile

Not defined in this standard

#### 4.4.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

#### 4.5 Use Cases

To list existing FileShares on the system, a client shall be able to find FileShares attached to a system (e.g., a file server) by doing an association traversal from the ComputerSystem that represents the system using the HostedShare association.

#### 4.6 CIM Elements

Table 3 describes the CIM elements for File Export.

**Table 3 - CIM Elements for File Export**

Element Name	Requirement	Description
4.6.1 CIM_CIFSShare (Exported File Share)	Optional	Represents the CIFS sharing characteristics of a particular file element.
4.6.2 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to FileShare)	Conditional	Conditional Requirement: EnabledLogicalElementCapabilities. Expressed the ability for the file share to be named or have its state changed.
4.6.3 CIM_EnabledLogicalElementCapabilities (FileShare)	Optional	This class is used to express the naming and possible requested state change possibilities for file shares.
4.6.4 CIM_ExportedFileShareSetting (Setting)	Mandatory	The configuration settings for an Exported FileShare that is a setting for a FileShare (CIFSShare or NFSShare) available for exporting.
4.6.5 CIM_FileShare (Exported File Share)	Mandatory	Represents the sharing characteristics of a particular file element.
4.6.6 CIM_FileShareSettingData (FileShare)	Mandatory	Experimental. Associates a FileShare (CIFSShare or NFSShare) and ExportedFileShareSetting elements.
4.6.7 CIM_HostedShare	Mandatory	Represents that a shared element is hosted by a File Server Computer System.
4.6.8 CIM_NFSShare (Exported File Share)	Optional	Represents the NFS sharing characteristics of a particular file element.
4.6.9 CIM_SAPAvailableForFileShare	Mandatory	Represents the association between a ProtocolEndpoint to the file share that is being accessed through that SAP.
4.6.10 CIM_SharedElement	Mandatory	Associates a FileShare (CIFSShare or NFSShare) to the LocalFileSystem on which it is based.

**Table 3 - CIM Elements for File Export**

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileShare AND SourceInstance.CIM_FileShare::OperationalStatus <> PreviousInstance.CIM_FileShare::OperationalStatus	Optional	CQL -Change of Status of a FileShare. PreviousInstance is optional, but may be supplied by an implementation of the Profile.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FileShare	Mandatory	CQL -Creation of a FileShare element.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FileShare	Mandatory	CQL -Deletion of a FileShare element.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM16'	Conditional	FileShare is degraded (if hardware implementation can support).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM17'	Mandatory	FileShare issues have been cleared (if FSM16 is supported).

**4.6.1 CIM\_CIFSShare (Exported File Share)**

The CIM\_CIFSShare is a subclass of CIM\_FileShare. It is optional, since an implementation may instantiate either (or both) of CIM\_CIFSShare or CIM\_NFSShare.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 4 describes class CIM\_CIFSShare (Exported File Share).

**Table 4 - SMI Referenced Properties/Methods for CIM\_CIFSShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section 4.6.5 CIM_FileShare (Exported File Share).
ElementName		Mandatory	See the ElementName definition in section 4.6.5 CIM_FileShare (Exported File Share).
Name		Mandatory	See the Name definition in section 4.6.5 CIM_FileShare (Exported File Share).
SharingDirectory		Mandatory	See the SharingDirectory definition in section 4.6.5 CIM_FileShare (Exported File Share).
OperationalStatus		Mandatory	See the OperationalStatus definition in section 4.6.5 CIM_FileShare (Exported File Share).
Description	N	Optional	See the Description definition in section 4.6.5 CIM_FileShare (Exported File Share).

**4.6.2 CIM\_ElementCapabilities (EnabledLogicalElementCapabilities to FileShare)**

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static  
Requirement: Optional

Table 5 describes class CIM\_ElementCapabilities (EnabledLogicalElementCapabilities to FileShare).

**Table 5 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (EnabledLogicalElementCapabilities to FileShare)**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The capabilities object associated with the file share.
ManagedElement		Mandatory	The FileShare.

#### 4.6.3 CIM\_EnabledLogicalElementCapabilities (FileShare)

Experimental.

Created By: Static  
Modified By: Static  
Deleted By: Static  
Requirement: Optional

Table 6 describes class CIM\_EnabledLogicalElementCapabilities (FileShare).

**Table 6 - SMI Referenced Properties/Methods for CIM\_EnabledLogicalElementCapabilities (FileShare)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	The moniker for the instance.
ElementNameEditSupported		Mandatory	Denotes whether a file share can be named.
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See MOF for details.
RequestedStatesSupported		Optional	Expresses the states to which this file share may be changed using the RequestStateChange method. If this property, it may be assumed that the state may not be changed.
GetElementNameCapabilities()		Conditional	Conditional requirement: Required if File Export Manipulation is implemented.

#### 4.6.4 CIM\_ExportedFileShareSetting (Setting)

Created By: External  
Modified By: External  
Deleted By: External  
Requirement: Mandatory

Table 7 describes class CIM\_ExportedFileShareSetting (Setting).

**Table 7 - SMI Referenced Properties/Methods for CIM\_ExportedFileShareSetting (Setting)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	A unique ID for the setting.
ElementName		Mandatory	A user-friendly name for the Setting.
FileSharingProtocol		Mandatory	The file sharing protocol supported by this share. NFS (2) and CIFS (3) are the supported values.
ProtocolVersions		Mandatory	An array of the versions of the supported file sharing protocol. A share may support multiple versions of the same protocol.
InitialEnabledState	N	Optional	Valid values are '1 2 3 7 8 9' for ('Other'   'Enabled'   'Disabled'   'In Test'   'Deferred'   'Quiesce').
OtherEnabledState	N	Optional	This should be filled in if the InitialEnabledState is '1'.
DefaultUserIdSupported	N	Optional	Valid values are '2 3 4' for ('No Default User Id'   'System-Specified Default User Id'   'Share-Specified Default User Id').
RootAccess	N	Optional	Valid values are '2 3' for ('No Root Access'   'Allow Root Access').
AccessPoints	N	Optional	Valid values are '2 3 4 5' for ('None'   'Service Default'   'All'   'Named Points').

#### 4.6.5 CIM\_FileShare (Exported File Share)

SMI-S treats CIM\_FileShare as an abstract class. It is mandatory because an implementation shall instantiate either (or both) CIM\_CIFSShare or CIM\_NFSShare.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 8 describes class CIM\_FileShare (Exported File Share).

**Table 8 - SMI Referenced Properties/Methods for CIM\_FileShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	A unique id for the FileShare element.
ElementName		Mandatory	This shall be a user friendly name for the FileShare.
Name		Mandatory	This shall be an opaque string that uniquely identifies the path to the directory or file.
SharingDirectory		Mandatory	Indicates if the shared element is a file or a directory. This is useful when importing but less so when exporting.
OperationalStatus		Mandatory	The OperationalStatus of the FileShare as defined in section 4.2.1.
Description	N	Optional	This a comment describing the file share.

#### 4.6.6 CIM\_FileShareSettingData (FileShare)

Experimental.

Created By: External  
 Modified By: Static  
 Deleted By: External  
 Requirement: Mandatory

Table 9 describes class CIM\_FileShareSettingData (FileShare).

**Table 9 - SMI Referenced Properties/Methods for CIM\_FileShareSettingData (FileShare)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The FileShare (CIFSShare or NFSShare).
SettingData		Mandatory	The settings define on creation of the FileShare.

#### 4.6.7 CIM\_HostedShare

Created By: External  
 Modified By: External  
 Deleted By: External  
 Requirement: Mandatory

Table 10 describes class CIM\_HostedShare.

**Table 10 - SMI Referenced Properties/Methods for CIM\_HostedShare**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The Share that is hosted by a Computer System.
Antecedent		Mandatory	The Computer System that hosts the FileShare. This can be the top level or non-top level system, or a virtual file server. But it shall be a File Server (Dedicated='16').

#### 4.6.8 CIM\_NFSShare (Exported File Share)

The CIM\_NFSShare is a subclass of CIM\_FileShare. It is optional, since an implementation may instantiate either (or both) of CIM\_CIFSShare or CIM\_NFSShare.

Created By: External  
 Modified By: External  
 Deleted By: External  
 Requirement: Optional

Table 11 describes class CIM\_NFSShare (Exported File Share).

**Table 11 - SMI Referenced Properties/Methods for CIM\_NFSShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section 4.6.5 CIM_FileShare (Exported File Share).
ElementName		Mandatory	See the ElementName definition in section 4.6.5 CIM_FileShare (Exported File Share).



**Table 11 - SMI Referenced Properties/Methods for CIM\_NFSShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	See the Name definition in section 4.6.5 CIM_FileShare (Exported File Share).
SharingDirectory		Mandatory	See the SharingDirectory definition in section 4.6.5 CIM_FileShare (Exported File Share).
OperationalStatus		Mandatory	See the OperationalStatus definition in section 4.6.5 CIM_FileShare (Exported File Share).
Description	N	Optional	See the Description definition in section 4.6.5 CIM_FileShare (Exported File Share).

**4.6.9 CIM\_SAPAvailableForFileShare**

Created By: External

Modified By: Static

Deleted By: External

Requirement: Mandatory

Table 12 describes class CIM\_SAPAvailableForFileShare.

**Table 12 - SMI Referenced Properties/Methods for CIM\_SAPAvailableForFileShare**

Properties	Flags	Requirement	Description & Notes
FileShare		Mandatory	The file share that is made available through a SAP. In the File Export profile, these are FileShares configured for either export.
AvailableSAP		Mandatory	The Service Access Point that is available to this FileShare. This shall have a value of '4200' (NFS) or '4201' (CIFS).

**4.6.10 CIM\_SharedElement**

Created By: External

Modified By: Static

Deleted By: External

Requirement: Mandatory

Table 13 describes class CIM\_SharedElement.

**Table 13 - SMI Referenced Properties/Methods for CIM\_SharedElement**

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	The LocalFileSystem that is exporting some contained file or directory as a FileShare.
SameElement		Mandatory	The FileShare (CIFSShare or NFSShare) that exposes a contained file or directory of the LocalFileSystem as an exported object.

**STABLE**



---

---

## STABLE

### 5 File Export Manipulation Profile

#### 5.1 Description

##### 5.1.1 Synopsis

**Profile Name:** File Export Manipulation (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** FileExportService

**Scoping Class:** File server ComputerSystem element (with Dedicated property containing "16")

**Related Profiles:** Table 14 describes the related profiles for File Export Manipulation.

**Table 14 - Related Profiles for File Export Manipulation**

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	
File Export	SNIA	1.7.0	Mandatory	
Simple Identity Management	DMTF	1.1.0	Mandatory	See DSP1034, version 1.1.0.
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

##### 5.1.2 Overview

The File Export Manipulation Profile is a component profile of autonomous profiles that support filesystems. It makes use of elements of the Filesystem profiles and supports creation, modification and deletion of FileShares that are exported by the File Export Profile. A number of other profiles also make use of elements of the filesystem profile and will be referred to in this specification as "filesystem related profiles" -- these include but are not limited to the Filesystem Profile, the Filesystem Manipulation Profile, the File Export Profile, the NAS Head Profile, the Self-Contained NAS Profile, and so on.

In this release of SMI-S, the autonomous profiles that use the File Export Manipulation Profile are the NAS Head and Self-Contained NAS Profiles.

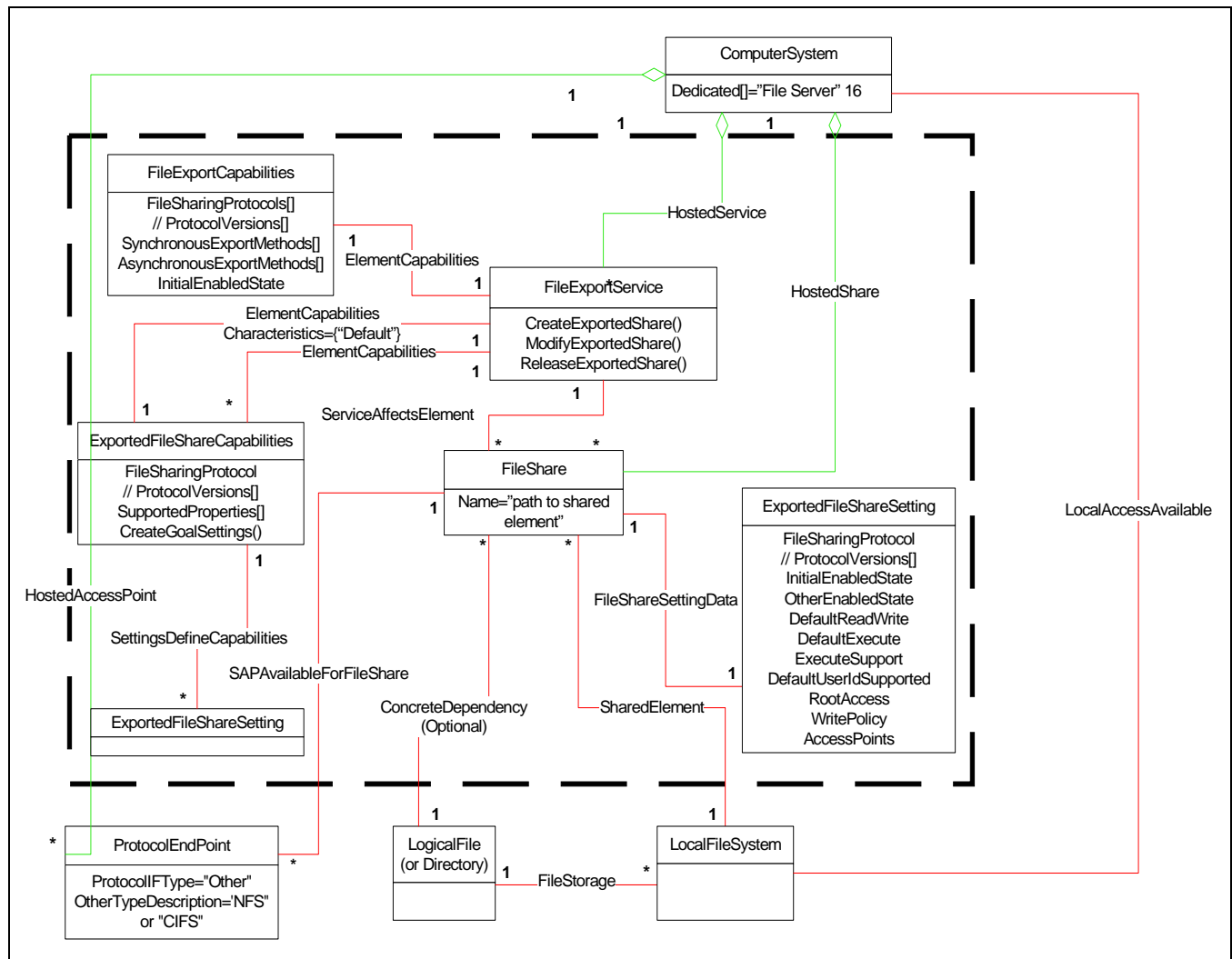
Annex B, "(Informative) State Transitions from Storage to File Shares" describes the states that a storage element, initially an unused LogicalDisk, goes through before it can be exported as a CIFS or NFS file share. The Filesystem Manipulation Profile provides the methods to create the filesystem as a LocalFileSystem and make it locally accessible at a file server ComputerSystem (associated to the file server ComputerSystem via the LocalAccessAvailableToFS association). This profile (the File Export Manipulation Profile) provides the methods to "Export a file share" from the file server that allows the file server to share its contents with remote operational users. Sharing the contents of a LocalFileSystem can be from the root directory or some contained internal directory, or some contained internal file. When a directory (root or otherwise) is shared, all files and sub-directories of that directory are also automatically shared (recursively). The semantics of sharing are ultimately controlled by the Authorization profiles and by the filesystem implementation, so sharing cannot violate the access rules specified internally to the filesystem. In addition to specifying the object (file or directory) to be shared, the filesystem implementation shall specify the protocol to use (CIFS, NFS, or other protocol) for sharing.

SMI-S uses a FileShare (CIFSShare or NFSShare) element to represent the externally accessible file share. A SharedElement association will exist between the FileShare (CIFSShare or NFSShare) and the LocalFileSystem. The FileShare.Name property indicates the shared object (it is the filesystem-specific path to the contained file or directory that is being shared). The format of Name is specific to the filesystem type indicated by the associated FileSystemSetting.ActualFileSystemType property; the LocalFileSystem.PathnameSeparatorString property indicates the "separator string" that may be used to split the PathName into the components of a hierarchical path name from the root of the associated filesystem (indicated by the LocalFileSystem).

### 5.1.3 Instance Diagrams

#### 5.1.3.1 File Export Creation classes and associations

Figure 6 illustrates the constructs involved with creating a FileShare (CIFSShare or NFSShare) for a File



**Figure 6 - File Export Manipulation Profile Instance**

Export Profile. This summarizes the mandatory classes and associations for this profile. Specific areas are discussed in later sections.

The FileExportService provides configuration support for exporting elements ('files' and 'directories') of a LocalFileSystem as FileShare (CIFSShare or NFSShare) elements. A FileExportService is hosted by the

file server ComputerSystem that exports the directories/files (these would be the file server ComputerSystems in the Filesystem profile that were given local access to the filesystem). FileShares are accessed through ServiceAccessPoint(s) hosted by the file server ComputerSystem. FileShares are associated with the FileExportService via ServiceAffectsElement and with the ServiceAccessPoint(s) via SAPAvailableToElement.

If a filesystem-related profile supports the File Export Manipulation Profile, it shall have at least one FileExportService element. This FileExportService shall be hosted on the top level ComputerSystem of the File Export Profile (which shall be a file server ComputerSystem element in the filesystem related profiles). The methods offered are CreateFileShare, ModifyFileShare, and ReleaseFileShare.

Associated to the FileExportService (via ElementCapabilities) shall be one FileExportCapabilities element that describes the capabilities of the service. It identifies the methods supported, whether the methods support Job Control or not, the protocols that the created file share can support, and whether or not the file share shall be made available after creation.

For each file sharing protocol that is supported, there shall be one ExportedFileShareCapabilities element that defines the range of capabilities supported for that particular file sharing protocol. The ExportedFileShareCapabilities elements shall be associated via ElementCapabilities to the FileExportService. One of the ExportedFileShareCapabilities may be identified as a default (by setting the property ElementCapabilities.IsDefault). The default ExportedFileShareCapabilities element also indicates the default file sharing protocol to be supported. These defaults apply if any of the extrinsic methods of the FileExportService are invoked with a NULL value for the Capabilities parameter.

Each ExportedFileShareCapabilities element is defined by a set of ExportedFileShareSettings that are associated to it by the SettingsDefineCapabilities association. These ExportedFileShareSettings may be structured to indicate a range of supported and unsupported property values and shall have the same value for the FileSharingProtocol property as the ExportedFileShareCapabilities element.

For the convenience of clients the File Export Manipulation Profile shall populate a set of "pre-defined" ExportedFileShareSettings for each of the ExportedFileShareCapabilities. These shall be associated to the ExportedFileShareCapabilities via the SettingsDefineCapabilities association -- the association shall have its SettingsDefineCapabilities.PropertyPolicy property set to "Correlated" and theSettingsDefineCapabilities.ValueRole property set to "Supported".

NOTE That they are pre-defined and therefore exist at all times does not imply that these ExportedFileShareSettings must be made persistent by the implementation.

The ExportedFileShareCapabilities instance supports the CreateGoalSettings method, described in detail in , "Table 16 shows methods and instances for FileExportManipulation.". This method supports establishing one client-defined ExportedFileShareSettings (as a goal).

CreateGoalSettings takes an array of embedded SettingData elements as the input TemplateGoalSettings and SupportedGoalSettings parameters and may generate an array of embedded SettingData elements as the output SupportedGoalSettings parameter. However, this profile only uses a single embedded ExportedFileShareSettings element in the input parameters (both TemplateGoalSettings and SupportedGoalSettings) and generate a single valid embedded ExportedFileShareSettings element as output (SupportedGoalSettings). If a client supplies a NULL ExportedFileShareSettings (i.e., the empty string) as input to this method, the returned ExportedFileShareSettings structure shall be a default setting for the parent ExportedFileShareCapabilities. If the input (the embedded ExportedFileShareSettings) is not NULL, the method may return a "best fit" to the requested setting. The client may iterate on the CreateGoalSettings method until it acquires a setting that suits its needs. This embedded settings structure may then be used when the CreateFileShare or ModifyFileShare methods are invoked. The details of how iterative negotiation can work are discussed in 5.4.1.1, "ExportedFileShareCapabilities.CreateGoalSettings". Note that the file sharing protocol indicated by the FileSharingProtocol property is invariant in all of these interactions. It is an error if the client changes the FileSharingProtocol property for a Setting and submits it as a goal to the Capabilities element that provided the original Setting.

Note that it is not possible to guarantee that negotiation will terminate with an agreed upon setting and therefore a fall-back mechanism is needed. This profile does not require negotiation – an implementation may support only a set of pre-defined correlated point settings that a client can preload and use without modification. The implementation could also support the only settings whose properties are selectable from an arithmetic progression or from a fixed enumeration, so that the client may construct a goal setting that is guaranteed to be supported without negotiation.

NOTE That a client has obtained a goal setting supported by the implementation does not guarantee that a create or modify request will succeed. Such a setting only specifies a supported static configuration not that the current dynamic environment has the resources to implement a specific request.

Armed with the goal (the embedded `ExportedFileShareSettings` element), a reference to a `LocalFileSystem`, and a path to a file or directory contained within that `LocalFileSystem`, the client can now use the `CreateFileShare` method to create the file share for export. The `CreateFileShare` method creates a `FileShare` element, and a new `ExportedFileShareSettings` instance as well as several necessary associations. These associations are:

- `HostedFileShare` association between the `FileShare` and the file server `ComputerSystem` that hosts it.
- `SharedElement` association between the `FileShare` and the `LocalFileSystem`. In addition, the `FileShare` element specifies the pathname for the shared element (file or directory) relative to the root of the `LocalFileSystem` (using the `Name` property).

---

---

## EXPERIMENTAL)

- `FileShareSettingData` to associate the `FileShare` to the `ExportedFileShareSetting` defined for it

---

---

## EXPERIMENTAL

- For backward compatibility with previous releases of SMI-S:
  - The file or directory of the `LocalFileSystem` that is being shared is represented as a `LogicalFile`
  - A `FileStorage` association is created between the `LogicalFile` and the `LocalFileSystem`
  - A `ConcreteDependency` association is created between the `FileShare` and the `LogicalFile`.
- In addition, optional parameters to the method can cause other classes to be created:
  - `DefaultUserId` could create a `Privilege` (see 5 File Export Manipulation Profile of *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*) associated to the `FileShare` as `AuthorizationTarget` and to a `UserIdentity` as `AuthorizationSource`
  - `RootAccessHosts` array parameter could create root access `Privileges` from a number of remote `Host ComputerSystems` to the `FileShare` (also using the Simple Identity Management Profile)
  - `AccessPointPorts` array parameter could create `SAPAvailableForFileShare` associations to a number of `ProtocolEndpoints` representing TCP/IP ports through which access is provided to this `FileShare`.

To determine if the implementation supports supplying the `ElementName` during the creation of a `FileShare` and to determine the supported methods to modify the `ElementName` of the existing `FileShare`, invoke the method `ExportedFileShareCapabilities.GetElementNameCapabilities`.

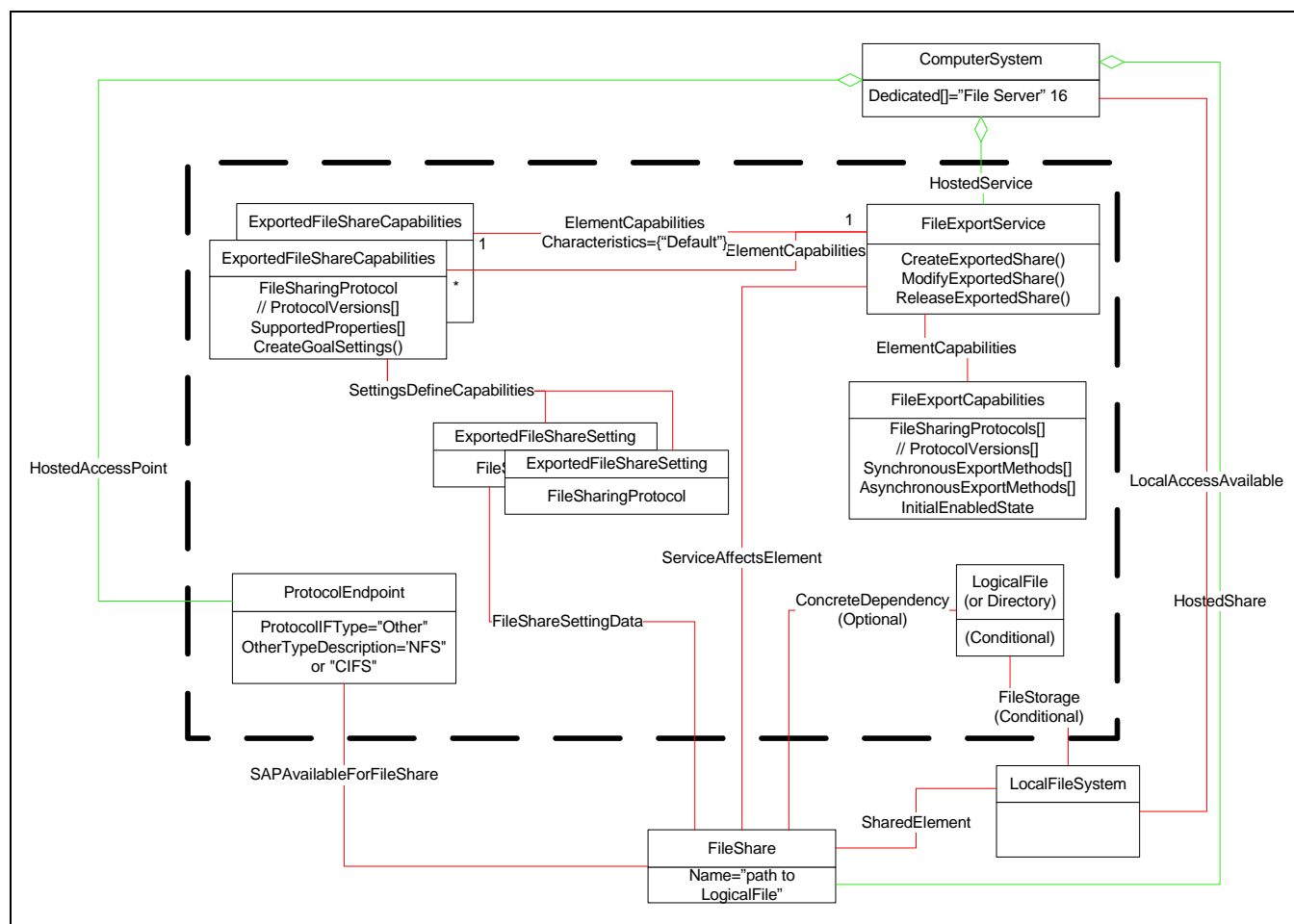
The `ReleaseFileShare` method is straightforward -- it deletes the `FileShare` and the `ExportedFileShareSetting`, and the associations to those elements (`HostedFileShare`, the `FileShareSettingData` element, `SharedElement`, all the `SAPAvailableForFileShare` associations and all `Privileges` that reference this `FileShare` as an `AuthorizationTarget`). Any `ComputerSystem` elements created to represent remote hosts with root access to this `FileShare` that have no further references may

also be removed. A LogicalFile element associated to the LocalFileSystem via FileStorage will not necessarily be deleted (the implementation may keep track of the other users of this element and be able to delete it). For similar reasons, a ProtocolEndPoint created as a result of being specified in the AccessPointPorts parameter may not be deleted. In both these cases, if the element has no associations other than the scoping one (FileStorage to LocalFileSystem for LogicalFile and HostedAccessPoint to ComputerSystem for ProtocolEndPoint) the provider may stop surfacing it at any time.

The `ModifyFileShare` method modifies an existing `FileShare` -- this requires a new `ExportedFileShareSetting` element to be used as a goal. But not any `ExportedFileShareSetting` will do: the client shall use the `ExportedFileShareCapabilities.CreateGoalSettings` method which would have been used to create the file share, or an appropriate compatible `ExportedFileShareCapabilities` instance. The `CreateGoalSettings` method is used to establish a new `ExportedFileShareSetting` goal (as with the original file share creation, it may be necessary to iterate on the `CreateGoalSettings` method). Since only properties controlled by Settings can be changed by `ModifyFileShare`, elements surfaced as a side-effect of creating or modifying a file share (i.e., any `ComputerSystems` created to represent remote hosts with root access or an `ProtocolEndPoints` created to represent access points for the share, or any user id created as a default user id) cannot be deleted, though new ones can be created and/or added), the effect of `ModifyFileShare` is to change some properties of the `FileShare` or of the associated `ExportedFileShareSetting`.

### 5.1.3.2 Finding File Export Services, Capabilities and Pre-defined Settings

When creating a file share the first step is to determine what can be created. Figure 7 illustrates an instance diagram showing the elements that shall exist for supporting fileshare creation.



### Figure 7 - Capabilities and Settings for Exported File Share Creation

At least one FileExportService shall exist if the Filesystem Profile has implemented the File Export Manipulation Profile. The instance(s) of this service can be found by following the HostedService association and filtering on the target class of FileExportService.

NOTE If no service is found from the Top Level file server ComputerSystem, the client should look for other component file server ComputerSystems that may be hosting the service. This is not recommended, but permitted.

An instance of the FileExportCapabilities shall be associated to the FileExportService via the ElementCapabilities association. A client should follow this association (filtering on the result value of "CIM\_FileExportCapabilities") to inspect the configuration capabilities that are supported. The client would choose between the file sharing protocols specified in the array property FileSharingProtocol.

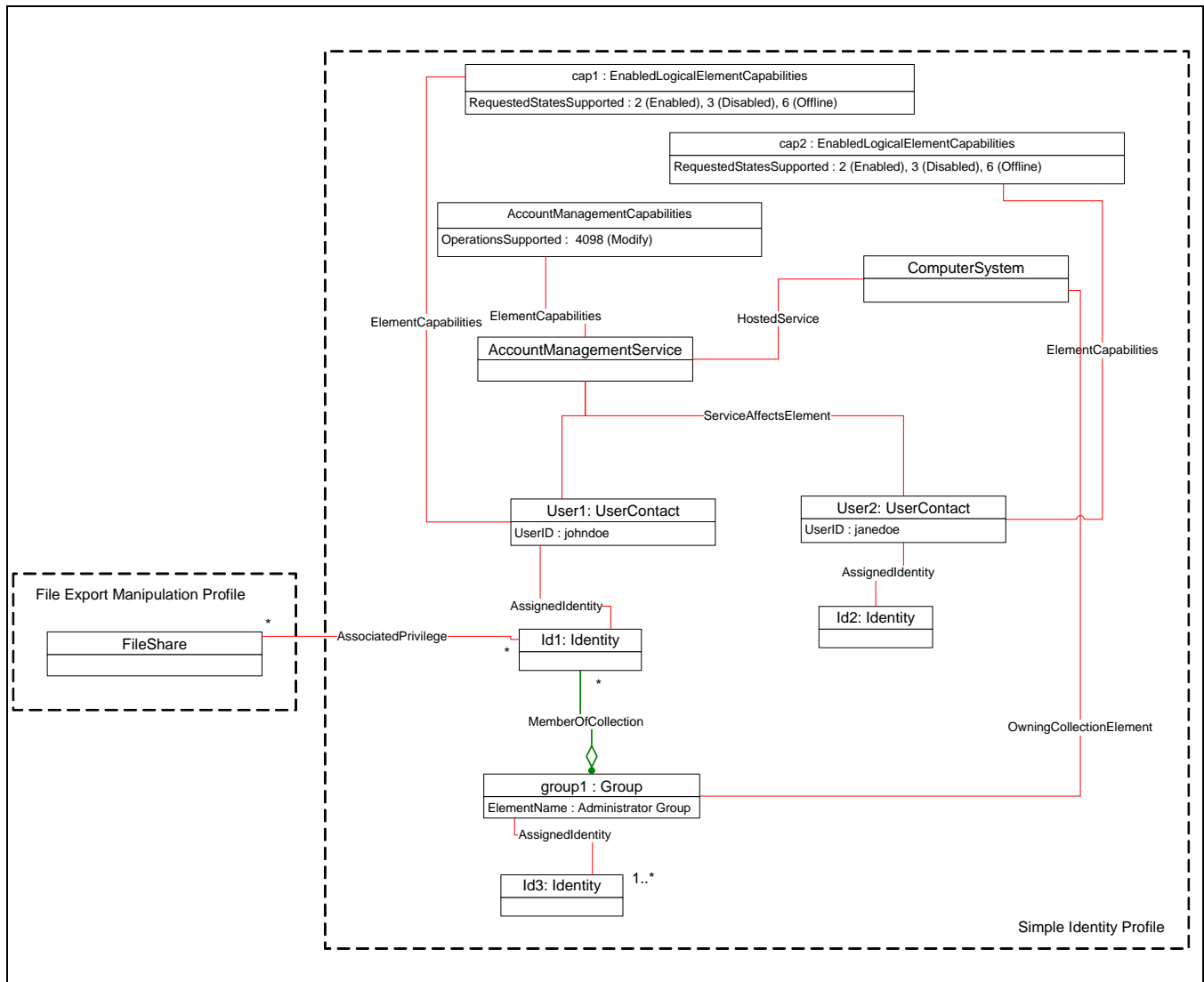
For each entry in the FileSharingProtocol array, there shall be one instance of ExportedFileShareCapabilities with the same value for the FileSharingProtocol property that shall be associated to the FileExportService using the ElementCapabilities association (filtering on the result value of "CIM\_ExportedFileShareCapabilities"). This ExportedFileShareCapabilities element shall specify the supported capabilities for that FileSharingProtocol using a collection of ExportedFileShareSetting elements. These ExportedFileShareSettings shall be associated with the ExportedFileShareCapabilities via SettingsDefineCapabilities.

An implementation may support a set of pre-defined ExportedFileShareSetting that clients could use directly if desired. The SettingsDefineCapabilities association from the ExportedFileShareCapabilities to the pre-defined ExportedFileShareSettings shall have the PropertyPolicy property be "Correlated", the ValueRole property be "Supported" and the ValueRange property be "Point". Other pre-defined combinations of property values may be specified by ExportedFileShareSetting whose SettingsDefineCapabilities association has the PropertyPolicy be "Independent", ValueRole property be "Supported" and the ValueRange array property contain "Minimums", "Maximums", or "Increment". These settings can be used by the client to compose ExportedFileShareSetting that are more likely to be directly usable.



### 5.1.3.3 Modeling for FileShare Access Control Lists

Figure 8 illustrates the modeling for FileShare ACLs (Access Control Lists) using the DMTF Simple Identity Management Profile.



**Figure 8 - FileShares and Simple Identity Management**

An implementation shall support "Read" (5) and "Write" (6) for CIM\_AssociatedPrivilege.Activities[]

To assign an ID with a privilege to a share, a client will invoke the method AssignPrivilegeToExportedShare providing a list of Identities, the Share and the Activities.

Groups are optional and only shown as informational, but a client will need to know to check for Accounts, UserContact or Group in the case an implementation has the support. If a client traverses AssignedIdentity from an Identity, the client could receive one of these three types of instances.

The AssociatedPrivilege class contains the following properties:

- Subject
- Target

- UseKey
- PrivilegeGranted (shall support at least true)
- Activities

The modification and deletion of AssociatedPrivilege can be done by intrinsic methods (ModifyInstance and DeleteInstance). The UserID property of CIM\_UserContact shall be persisted by the implementation as provided by the user when calling CIM\_AccountManagementService.CreateUserContact.

## 5.2 Health and Fault Management Considerations

The key elements of this profile are the FileExportService and the file server ComputerSystem. For the computer system, see 22.2.5 Computer System Operational Status in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*.

### 5.2.1 OperationalStatus for FileExportService

Table 15 shows operational status for FileExport services.

**Table 15 - Operational Status for FileExport Service**

Primary OperationalStatus	Description
2 "OK"	The service is running with good status
3 "Degraded"	The service is operating in a degraded mode. This could be due to the health state of the underlying file server, or of the storage being degraded or in error.
4 "Stressed"	The services resources are stressed
5 "Predictive Failure"	The service might fail because some resource or component is predicted to fail
6 "Error"	An error has occurred causing the service to become unavailable. Operator intervention through SMI-S to restore the service may be possible.
6 "Error"	An error has occurred causing the service to become unavailable. Automated recovery may be in progress.
7 "Non-recoverable Error"	The service is not functioning. Operator intervention through SMI-S will not fix the problem.
8 "Starting"	The service is in process of initialization and is not yet available operationally.
9 "Stopping"	The service is in process of stopping, and is not available operationally.
10 "Stopped"	The service cannot be accessed operationally because it is stopped -- if this did not happened because of operator intervention or happened in real-time, the OperationalStatus would have been "Lost Communication" rather than "Stopped".
11 "In Service"	The service is offline in maintenance mode, and is not available operationally.
13 "Lost Communications"	The service cannot be accessed operationally -- if this happened because of operator intervention it would have been "Stopped" rather than "Lost Communication".
14 "Aborted"	The service is stopped but in a manner that may have left it in an inconsistent state.
15 "Dormant"	The service is offline; and the reason for not being accessible is unknown.
16 "Supporting Entity in Error"	The service is in an error state, or may be OK but not accessible, because a supporting entity is not accessible.

### 5.3 Cascading Considerations

Not Applicable.

### 5.4 Methods of the Profile

#### 5.4.1 Extrinsic Methods of the Profile

Table 16 shows methods and instances for FileExportManipulation.

**Table 16 - FileExportManipulation Methods**

Method	Created Instances	Deleted Instances	Modified Instances
CreateExportedShare	FileShare (Export) ExportedFileShareSetting FileShareSettingData HostedShare SharedElement SAPAvailableForFileShare ServiceAffectsElement LogicalFile (or Directory) <i>(for bc to 1.1)</i> ProtocolEndPoint	N/A	N/A
ModifyExportedShare			ExportedFileShareSetting FileShare (Export) ProtocolEndPoint
ReleaseExportedShare	N/A	FileShare (Export) ExportedFileShareSetting FileShareSettingData HostedShare SharedElement ServiceAffectsElement ProtocolEndPoint LogicalFile	N/A
AssignPrivilegeToExportedShare	AssociatedPrivilege	N/A	N/A
CreateGoalSettings	N/A	N/A	N/A
GetElementNameCapabilities	N/A	N/A	N/A
CreateUserContact	UserContact Identity	N/A	N/A
GetUserContacts	N/A	N/A	N/A

##### 5.4.1.1 ExportedFileShareCapabilities.CreateGoalSettings

This extrinsic method of the ExportedFileShareCapabilities class validates support for a caller-proposed ExportedFileShareSetting passed as the TemplateGoalSettings parameter. This profile restricts the usage of this method to a single entry array for both TemplateGoalSettings and SupportedGoalSettings parameters.

If the input `TemplateGoalSettings` is `NULL` or the empty string, this method returns a single default `ExportedFileShareSetting` in the `SupportedGoalSettings` array. Both `TemplateGoalSettings` and `SupportedGoalSettings` are string arrays containing embedded instances of type `ExportedFileShareSetting`. As such, these settings do not exist in the implementation but are the responsibility of the client.

If the `TemplateGoalSettings` specifies values that cannot be supported, this method shall return an appropriate error and should return a best match for a `SupportedGoalSettings`.

The client and the implementation can engage in a negotiation process that may require iterative calls to this method. To assist the implementation in tracking the progress of the negotiation, the client may pass previously returned values of `SupportedGoalSettings` as the new input value of `SupportedGoalSettings`. The implementation may determine that a step has not resulted in progress if the input and output values of `SupportedGoalSettings` are the same. A client may infer from the same result that the `TemplateGoalSettings` must be modified.

#### 5.4.1.1.1 Client Considerations

It is important to understand that the client is acting as an agent for a human user -- either a "system" administrator, or other entity with administrative privileges with respect to the filesystem, the filesystem host, or the file server or the file share. During negotiation, the client will show the current state to the user -- the `SupportedGoalSettings` received to date (either the latest or some subset), the `TemplateGoalSettings` proposed (the most recent, but possibly more). But the administrator needs a representation of what is available, possibly the range or sets of values that the different setting properties can take. Some decisions are assumed to have been made already, such as the file-sharing protocol to be used or the filesystem element to be shared or the resources allocated for providing local access to the filesystem from the file server.

The `SettingsDefineCapabilities` association from the selected `Capabilities` element provides the information for the client to lay out these options. "Point" settings can be identified using `ExportedFileShareSettings` -- these points can be further qualified to indicate whether these are supported (or not), and even whether they represent some ideal point in the space -- a "minimum", or a "maximum", or an "optimal" point. Other settings can provide ranges for properties -- by specifying a minimum, a maximum, and an increment an arithmetic progression of values can be specified (a continuous range can be specified with a zero increment). Specifying a set of supported values for a property that do not follow some pattern is possible, if a bit tedious.

The set of settings associated via `SettingsDefineCapabilities` are expected to be quite stable -- real systems do not continually vary the functionality they can support. Such variations do occur -- for instance, if a new PCMCIA card is added to a running system -- and the best way for a client to be able to add these to the set of choices presented to a user is to subscribe to indications on new `Capabilities` elements and new instances of `SettingsDefineCapabilities`.

There is no guarantee that a negotiation will terminate successfully with the client and the implementation achieving agreement. The implementation may support some simpler mechanisms, short of fully-fledged negotiation, that would be used by a client to obtain an acceptable `TemplateGoalSettings`. The following two use cases are easily covered:

- 1) Client only considers only the canned settings specified exactly. The canned settings are specified by the `ExportedFileShareSetting` elements that are associated to the `ExportedFileShareCapabilities` via `SettingDefinesCapabilities` association with the following property values:
  - `SettingDefinesCapabilities.PropertyPolicy` = "Correlated"
  - `SettingDefinesCapabilities.ValueRole` = "Supported"
  - `SettingDefinesCapabilities.ValueRange` = "Point"

- 2) Client considers canned settings that range over values specified using minimum/increment/maximum for the Setting properties. For example, these could be specified by the ExportedFileShareSetting elements that are associated to the ExportedFileShareCapabilities via SettingDefinesCapabilities association with the following property values:

- SettingDefinesCapabilities.ValueRange = "Minimums" or "Maximums" or "Increments"
- The PropertyPolicy and ValueRole properties of SettingDefinesCapabilities will be appropriately specified

#### 5.4.1.1.2 Signature and Parameters of CreateGoalSettings

Table 17 describes the parameters for Extrinsic Method ExportedFileShareCapabilities.CreateGoalSettings.

**Table 17 - Parameters for Extrinsic Method ExportedFileShareCapabilities.CreateGoalSettings**

Parameter Name	Qualifier	Type	Description & Notes
TemplateGoalSettings[]	IN	string	EmbeddedInstance ("ExportedFileShareSetting") TemplateGoalSettings is a string array containing embedded instances of class ExportedFileShareSetting, or a derived class. This parameter specifies the client's requirements and is used to locate matching settings that the implementation can support.
SupportedGoalSettings[]	INOUT	string	EmbeddedInstance ("ExportedFileShareSetting") SupportedGoalSettings is a string array containing embedded instances of class ExportedFileShareSetting, or a derived class. On input, it specifies a previously returned set of Settings that the implementation could support. On output, it specifies a new set of Settings that the implementation can support. If the output set is identical to the input set, both client and implementation may conclude that this is the best match for the TemplateGoalSettings that is available. If the output does not match the input and the non-NULL output does not match the non-NULL TemplateGoalSettings, then the method shall return "Alternative Proposed". If the output is NULL, the method shall return an "Failed".
<b>Normal Return</b>			
Status		uint32	"Success", "Failed", "Timeout", "Alternative Proposed"
<b>Error Returns</b>			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

#### 5.4.1.2 ExportedFileShareCapabilities.GetElementNameCapabilities

This method indicates if ElementName can be specified as a part of invoking an appropriate method of FileExportService to create a new FileShare. Additional, the returned data includes the methods that can be used to modify the ElementName of existing FileShare.

```

uint32 GetElementNameCapabilities(
    [OUT,
    ValueMap { "2", "3", "4", "..", "32768..65535" },
    Values { "ElementName can be supplied during creation",
    "ElementName can be modified with InvokeMethod",
    "ElementName can be modified with intrinsic method",
    "DMTF Reserved", "Vendor Specific" }]
    uint32 SupportedFeatures[],
    [OUT] string ElementNameMask,
    [OUT] uint16 MaxElementNameLen);

```

The parameters are:

- **SupportedFeatures:** This OUT parameter is an array indicating what methods can accept the element name for creation or modification of a FileShare. For example, the value of "ElementName can be supplied during creation" indicates the method such as `CreateExportedShare( )` accepts the `ElementName` when creating a new FileShare. An empty array indicates `ElementNaming` for `ElementType` is not supported.
- **MaxElementNameLen:** This OUT parameter specifies the maximum supported `ElementName` length.
- **ElementNameMask:** This OUT parameter expresses the restrictions on `ElementName`. The mask is expressed as a regular expression. See DMTF standard ABNF with the Management Profile Specification Usage Guide, Annex C for the regular expression syntax permitted. Since the `ElementNameMask` can describe the maximum length of the `ElementName`, any length defined in the regexp is in addition to the restriction defined in `MaxElementNameLen` (causing the smaller value to be the maximum length). If NULL, it indicates no restrictions on the `ElementName`.

#### 5.4.1.3 FileExportService.CreateExportedShare

This extrinsic method creates a FileShare providing access to a contained sub-element of a `LocalFileSystem` (either a `LogicalFile` or its sub-class `Directory`). A reference to the created FileShare is returned as the output parameter `TheShare`. This FileShare element is hosted by the same file server `ComputerSystem` that hosts the `FileExportService`. The `LocalFileSystem` whose element is exported shall be locally accessible to the file server `ComputerSystem` (and need not be hosted by it), as represented by the `LocalAccessAvailableToFS` association from the file server `ComputerSystem` to the `LocalFileSystem`.

The `LocalFileSystem` whose sub-element is being exported is specified by the input parameter `Root`. The input string parameter `SharedElementPath` specifies a pathname from the root directory of the `Root` to the sub-element to be exported. If `SharedElementPath` is NULL or the empty string, it specifies the root directory of `Root`. The format of `SharedElementPath` is implementation-specific -- the most common format is as a sequence of directory names separated by a character or short string indicated by the `FileSystemSetting.PathNameSeparatorString` property.

NOTE The root directory of `Root` is the base from which a path to the `LogicalFile` being shared is specified. In the simplest and possibly the most common case, the `LogicalFile` element is the root directory of `Root` and the path is NULL or the empty string.

The desired settings for the FileShare are specified by the `Goal` parameter (a string-valued `EmbeddedInstance` object of class `ExportedFileShareSetting`). An `ExportedFileShareSetting` element shall be created that represents the settings of the created FileShare and will be associated via `FileShareSettingData` to the FileShare. (This `ExportedFileShareSetting` may be identical to the `Goal` or may be its equivalent). The created element shall be returned as the output `Goal` parameter.

The input `Goal` parameter can be NULL, in which case the `ExportedFileShareSetting` associated with the default `ExportedFileShareCapabilities` of the `FileExportService` is used as the goal. In that case, the following references to `Goal` are to the output value of the `Goal` parameter.

#### 5.4.1.3.1 Signature and Parameters of CreateExportedShare

Table 18 shows parameters for Extrinsic Method FileExportService.CreateExportedShare.

**Table 18 - Parameters for Extrinsic Method FileExportService.CreateExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
ElementName	IN	string	An end user relevant name for the FileShare being created. If NULL, then a system-supplied default name can be used.  The value shall be stored in the 'ElementName' property for the created element.
Comment	IN	string	An end user relevant comment for the FileShare being created. If NULL, then a system-supplied default comment can be used.  The value shall be stored in the 'Description' property for the created element.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
Root	IN, REF	LocalFileSystem	A reference indicating a LocalFileSystem element whose sub-element is being exported. The LocalFileSystem shall be locally available (either explicitly or implicitly) to the file server ComputerSystem that hosts the FileExportService.
SharedElementPath	IN, OUT	string	An opaque string representing a path to the shared element from the root directory of the FileSystem indicated by the Root parameter. The format of this is as a sequence of directory names (from the \"root\") separated by the PathNameSeparatorString property.  Multiple paths could lead to the same element but the access rights or other privileges could be specific to the path. The client needs to specify the path.  If SharedElementPath is NULL or is the empty string, it indicates the \"root\" directory of the filesystem indicated by Root.  The value shall be stored in the 'Name' property for the created element.
Goal	IN, OUT, EI	string	EmbeddedInstance ("ExportedFileShareSetting")  The client-specified requirements for how the specified FileShare element is to be shared or exported by the FileExportService. This is an element of the ExportedFileShareSetting class, or a derived class, encoded as a string-valued embedded object parameter. If NULL or the empty string, the default configuration will be specified by the FileExportService.
TheShare	OUT, REF	CIM_FileShare	If successful, this returns a reference to the created file share.

**Table 18 - Parameters for Extrinsic Method FileExportService.CreateExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
DefaultUserId	IN, OUT, REF, NULL allowed	CIM_identity	<p>A reference to a concrete derived class of CIM_Identity that indicates the user id to use for default access to this share. A NULL value on input indicates that no user id is requested. A NULL value on output indicates that no user id has been assigned, even by default.</p> <p>A default user id per share is not supported by the CIFS Protocol so this is ignored if the Goal specifies creating a CIFSShare.</p>
RootAccessHosts[]	IN, OUT, URI, NULL allowed	string	<p>An array of strings that specify the hosts that have root access to this Share, if the ExportedFileShareSetting.RootAccess property is set to 'Allow Root Access'. Each entry specifies a host by a URI. All entries up to the first empty string are allowed root access; the entries after the first empty string are denied root access. If this parameter is NULL, root access will be denied to all hosts, effectively overriding the value of the property ExportedFileShareSetting.RootAccess. If the first entry is the empty string, root access will be allowed from all hosts, and subsequent entries will be denied root access. This property needs to be an array of URIs because the remote host may not be known to the provider and therefore a reference to the host may not exist.</p> <p>Root Access is not supported by the CIFS Protocol so this is ignored if the Goal specifies creating a CIFSShare.</p>
AccessPointPorts[]	IN, OUT, REF, NULL allowed	CIM_ServiceAccessPoints	<p>An array of references to the ProtocolEndpoints that can connect to this Share, if the ExportedFileShareSetting.AccessPoints property is set to 'Named Ports'.</p> <p>If the parameter is NULL, all access points will be denied access, effectively overriding the value of the property ExportedFileShareSetting.AccessPoints.</p> <p>If the array has multiple entries and the first entry in the array is NULL, all access points supported by the service will be supported, and subsequent entries will be denied access.</p> <p>Any AccessPoints granted access via this parameter will also be associated to this share with SAPAvailableForFileShare. If the AccessPoint is not already enabled it will appear in a disabled state.</p> <p>The CIFS protocol does not support multiple ProtocolEndpoints, so this is ignored if the Goal specifies creating a CIFSShare.</p>
<b>Normal Return</b>			
Status	OUT	uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"



**Table 18 - Parameters for Extrinsic Method FileExportService.CreateExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
<b>Error Returns</b>			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

#### 5.4.1.4 FileExportService.ModifyExportedShare

This extrinsic method modifies a FileShare element that is exporting a contained sub-element of a LocalFileSystem (either a LogicalFile or its sub-class Directory). The FileShare is specified by the reference parameter TheShare. TheShare cannot be NULL and it shall be hosted by the same file server ComputerSystem that hosts the FileExportService. The input parameters Root and SharedElementPath shall be NULL or shall be the same as the corresponding parameters when the FileShare was created (i.e., these cannot be changed using this method).

The precise constraint is that the sub-element shall not be changed even if the Root and SharedElementPath are different. For instance, this would allow a different path that leads to the same sub-element. However, this profile does not allow this flexibility.

The new desired settings for the FileShare are specified by the input parameter Goal (a string-valued EmbeddedInstance object of class ExportedFileShareSetting). An ExportedFileShareSetting element that represents the settings of the created FileShare (either identical to the Goal or its equivalent) will be associated via FileShareSettingData to the FileShare. The implementation shall modify the existing ExportedFileShareSetting. The Setting that is actually established will be returned as the output parameter Goal.

Goal parameter is required and cannot be NULL.

If Goal.AccessPoints="Named Points", then the non-NULL input parameter AccessPointPorts will be an array of references to ProtocolEndpoints that provide access to this FileShare. This will be represented by creating instances of the SAPAvailableForFileShare association between the FileShare and the specified ProtocolEndpoint. Any existing specification of access points to the FileShare will be overridden. If AccessPointPorts is NULL, the existing specification will not be changed.

NOTE This changes the type of the AccessPointPorts parameter from a string array to an array of references to ProtocolEndpoints (or more generally to ServiceAccessPoints).

The input parameters InUseOptions and WaitTime provide for delaying or aborting the execution of this method if the FileShare is in use and the method requires that no operations be in progress during that execution.

NOTE The WaitTime and InUseOptions parameters are supported if the ExportedFileShareCapabilities.SupportedProperties includes the "RequireInUseOptions" option. This requires a change to the MOF that may not show up in this document as enumerations are not documented in the spec.

#### 5.4.1.4.1 Signature and Parameters of ModifyExportedShare

Table 19 shows parameters for Extrinsic Method FileExportService.ModifyExportedShare.

**Table 19 - Parameters for Extrinsic Method FileExportService.ModifyExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
ElementName	IN	string	A new end-user relevant name for the FileShare being modified. If NULL or the empty string, the existing name stored in the 'ElementName' property for the created element shall not be changed.
Comment	IN	string	A new end-user relevant comment for the FileShare being modified. If NULL or the empty string, the existing comment stored in the 'Description' property will not be changed.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
Root	IN, OUT, REF	CIM_ManagedElement	A reference indicating a LocalFileSystem element whose sub-element is being exported. In the ModifyExportedShare method, this shall not indicate a different filesystem from the one indicated when the file share was created (even if the reference is to a different instance of LocalFileSystem).  If Root is NULL on input it is ignored.  As an OUT parameter, a reference to the LocalFileSystem is returned.
SharedElementPath	IN, OUT	string	A string representing a path to the shared element from the root directory of the LocalFileSystem indicated by Root.  The ModifyExportedShare method cannot be used to change the object indicated by the path, but the path itself can be different as multiple paths could lead to the same element. Such a change may have side-effects, for instance, the access rights or other privileges could be specific to the path.  If SharedElementPath is NULL, it indicates no change to the current path. If SharedElementPath consists of a single empty string, it indicates the root directory of the Filesystem indicated by Root.  As an OUT parameter, the current path is returned.  The value shall be stored in the 'Name' property for the created element.
Goal	IN, OUT, EI	string	EmbeddedInstance ("ExportedFileShareSetting")  The client-specified requirements for how the specified FileShare element is to be shared or exported by the FileExportService. This is an element of the ExportedFileShareSetting class, or a derived class, encoded as a string-valued embedded instance parameter. If NULL or the empty string, the current setting will be re-applied.  As an OUT parameter, the current Setting is returned.
TheShare	IN, OUT, REF	CIM_FileShare	As an IN Parameter, it specifies the share that is to be modified or whose settings are being queried. As an OUT Parameter, this specifies the share if the request is successful.

**Table 19 - Parameters for Extrinsic Method FileExportService.ModifyExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
DefaultUserId	IN, OUT, REF, NULL allowed,	CIM_identity	<p>As an IN parameter, this is a reference to a concrete derived class of CIM_Identity that indicates the user id to use for default access to this share. A NULL value indicates no change to the existing user id, if one has been specified. The provider is expected to surface this access using the Simple Identity Management Profile. As an OUT Parameter, this returns a reference to the current DefaultUserId.</p> <p>A default user per share is not supported by the CIFS Protocol so this is ignored if the file share is a CIFSShare.</p>
RootAccessHosts[]	IN, OUT, URI, NULL allowed	string	<p>An array of strings that specify the hosts that have root access to this Share, if the ExportedFileShareSetting.RootAccess property is set to 'Allow Root Access'. Each entry specifies a host by a URI. The set of hosts specified is added to the existing set of hosts with root access.</p> <p>If this parameter is NULL, root access will be denied to all hosts, including the ones currently allowed root access, effectively overriding the value of the property ExportedFileShareSetting.RootAccess.</p> <p>Each entry specifies a host by a URI. All entries up to the first empty string are allowed root access; the entries after the first empty string are denied root access.</p> <p>If the first entry is the empty string, root access will continue to be allowed from the existing hosts, and subsequent entries in the array will be denied root access.</p> <p>This property needs to be an array of URIs because the remote host may not be known to the provider and therefore a reference to the host may not exist.</p> <p>Root Access is not supported by the CIFS Protocol so this is ignored if the Goal specifies creating a CIFSShare.</p>
AccessPointPorts[]	IN, OUT, REF, NULL Allowed	CIM_ServiceAccessPoints	<p>An array of references to the ProtocolEndpoints that can connect to this Share, if the ExportedFileShareSettings.AccessPoints property is set to 'Named Ports'. The set of access points specified in the array is added to the existing set of access points.</p> <p>If the parameter is NULL, all access points will be denied access, effectively overriding the value of the property ExportedFileShareSetting.AccessPoints.</p> <p>If the first entry in the array is NULL, existing access points supported by the service will be supported, and subsequent entries in the array will be access points that are denied access.</p> <p>Any AccessPoints granted access via this parameter will also be associated to this share with SAPAvailableForFileShare. If the AccessPoint is not already enabled it will appear in a disabled state.</p> <p>The CIFS protocol does not support multiple ProtocolEndpoints, so this is ignored if the Goal specifies creating a CIFSShare.</p>
InUseOptions	IN	uint16	<p>An enumerated integer that specifies the action that the provider should take if the FileShare is still in use when this request is made. The WaitTime parameter indicates the specified time used for this function.</p> <p>This option is only relevant if the FileShare needs to be made unavailable while the request is being executed.</p>

**Table 19 - Parameters for Extrinsic Method FileExportService.ModifyExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
WaitTime	IN	uint32	An integer that indicates the time (in seconds) that the provider needs to wait before executing this request if it cannot be done while the FileShare is in use. If WaitTime is not zero, the method will create a job, if supported by the provider, and return immediately. If the provider does not support asynchronous jobs, there is a possibility that the client could time-out before the job is completed.  The combination of InUseOptions = '4' and WaitTime = '0' (the default) is interpreted as 'Wait (forever) until Quiescence, then Execute Request' and will be performed asynchronously if possible.
<b>Normal Return</b>			
Status	OUT	uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
<b>Error Returns</b>			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

#### 5.4.1.5 FileExportService.ReleaseExportedShare

This is an extrinsic method that will delete a FileShare specified by the parameter TheShare and delete any associated instances and associations that are no longer needed. The deleted instances will include the Directory (or LogicalFile) if it had been created only for the purpose of representing the shared sub-element.

NOTE Deleting the Directory or LogicalFile deletes only the representation of the file or directory for management and does not delete the underlying operational element

The deleted associations include HostedShare, FileShareSettingData, and any elements and associations created to support the DefaultUserId, RootAccessHosts, and AccessPointPorts parameters. In addition, the ExportedFileShareSetting will be deleted if appropriate.

The input parameters InUseOptions and WaitTime provide for delaying or aborting the execution of this method if the FileShare is in use and the method requires that no operations be in progress during that execution.

NOTE The WaitTime and InUseOptions parameters are supported if the ExportedFileShareCapabilities.SupportedProperties includes the "RequireInUseOptions" option.

##### 5.4.1.5.1 Signature and Parameters of ReleaseExportedShare

Table 20 shows parameters for Extrinsic Method FileExportService.ReleaseExportedShare.

**Table 20 - Parameters for Extrinsic Method FileExportService.ReleaseExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
TheShare	IN, OUT, REF	CIM_FileShare	As an IN Parameter, it specifies the share that is to be modified or whose settings are being queried. As an OUT Parameter, this specifies the share if the request is successful.

**Table 20 - Parameters for Extrinsic Method FileExportService.ReleaseExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
InUseOptions	IN	uint16	An enumerated integer that specifies the action that the provider should take if the FileShare is still in use when this request is made. The WaitTime parameter indicates the specified time used for this function.  This option is only relevant if the FileShare needs to be made unavailable while the request is being executed.
WaitTime	IN	uint32	An integer that indicates the time (in seconds) that the provider needs to wait before executing this request if it cannot be done while the FileShare is in use. If WaitTime is not zero, the method will create a job, if supported by the provider, and return immediately. If the provider does not support asynchronous jobs, there is a possibility that the client could time-out before the job is completed.  The combination of InUseOptions = '4' and WaitTime = '0' (the default) is interpreted as 'Wait (forever) until Quiescence, then Execute Request' and will be performed asynchronously if possible.
<b>Normal Return</b>			
Status	OUT	uint32	ValueMap{}, Values{}  "Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
<b>Error Returns</b>			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

**5.4.1.6 FileExportService.AssignPrivilegeToFileShare**

This is an extrinsic method that grants the specified Identities permissions to the FileShare specified by "TheShare". The method creates any AssociatedPrivilege instances that are necessary to represent the requested permissions.

Table 21 shows parameters for Extrinsic Method FileExportService.AssignPrivilegeToExportedShare.

**Table 21 - Parameters for Extrinsic Method FileExportService.AssignPrivilegeToExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
Identities	IN, REF	CIM_Identity[]	The list of Identities to assign privilege to share
Activities	IN	uint16[]	The Activities to assign to the share. The Activities are defined in the CIM_AssociatedPrivilege.Activities property
TheShare	IN, REF	CIM_FileShare	The FileShare to assign the privileges
<b>Normal Return</b>			
Status	OUT	uint32	0 ("Completed with No Error")
<b>Error</b>			
			1 ("Not Supported")

**Table 21 - Parameters for Extrinsic Method FileExportService.AssignPrivilegeToExportedShare**

Parameter Name	Qualifier	Type	Description & Notes
			2 ("Failed")
			3 ("Activities Not Supported")
			4 ("Identity Not Found")
			5 ("File Share Not Found")

**5.4.1.6.1 Signature and Parameters of AccountManagementService.CreateUserContact**

Table 22 shows parameters for Extrinsic Method AccountManagementService.CreateUserContact.

**Table 22 - Parameters for Extrinsic Method AccountManagementService.CreateUserContact**

Parameter Name	Qualifier	Type	Description & Notes
System	IN, REF	CIM_ComputerSystem	The scoping ComputerSystem in which to create the UserContact
UserContactTemplate	IN	string	UserContactTemplate is a template for the desired UserContact to be created.
Identities	OUT, REF	CIM_Identity[]	Reference to the instances of CIM_Identity when the method returns a value of 0. These instances may not be created as a part of the execution of this method. These instances may already exist prior to the invocation of this method.
UserContact	OUT, REF	CIM_UserContact	Reference to the instance of CIM_UserContact created when the method returns a value of 0
<b>Normal Return</b>			
Status	OUT	int32	0 ("Completed with No Error")
<b>Error Return</b>			
			1 ("Not Supported")
			2 ("Failed")

**5.4.1.7 AccountManagementService.GetUserContacts**

This is an extrinsic method that retrieves instances of UserContact that have the specified UserID. Upon successful completion of the method, it returns an array of UserContact instances that are already associated to instances of Identity through AssignedIdentity.

#### 5.4.1.7.1 Signature and Parameters of AccountManagementService.GetUserContacts

Table 23 shows parameters for Extrinsic Method AccountManagementService.GetUserContacts.

**Table 23 - Parameters for Extrinsic Method AccountManagementService.GetUserContacts**

Parameter Name	Qualifier	Type	Description & Notes
UserID	IN	String	The UserID for the CIM_UserContact instances to be retrieved
UserContacts	OUT, REF	CIM_UserContact[]	An array of references to the UserContact instances that have the supplied UserID
<b>Normal Return</b>			
Status	OUT	int32	0 ("Completed with No Error")
<b>Error Return</b>			
			1 ("Not Supported")
			2 ("Failed")

#### 5.4.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

### 5.5 Use Cases

Not defined in this standard.

#### 5.5.1 File Export Manipulation Supported Capabilities Patterns

Table 24 lists the capabilities patterns that are formally recognized by SMI-S 1.2.0 for determining capabilities of various implementations:

**Table 24 - SMI-S File Export Supported Capabilities Patterns**

FileSharingProtocols	SynchronousMethods	AsynchronousMethods	InitialExportState
NFS, CIFS	Export Creation, Export Modification, Export Deletion	Null	Any state is valid.
NFS, CIFS	Null	Export Creation, Export Modification, Export Deletion	Any state is valid.
NFS, CIFS	Null	Null	Null

## 5.6 CIM Elements

Table 25 describes the CIM elements for File Export Manipulation.

**Table 25 - CIM Elements for File Export Manipulation**

Element Name	Requirement	Description
5.6.1 CIM_AccountManagementService	Mandatory	Creates and manages UserContacts on behalf of other SecurityServices
5.6.2 CIM_AssignedIdentity	Mandatory	Associates an Identity to a specific ManagedElement (in this case UserContact), whose trust and account information is represented
5.6.1 CIM_AccountManagementService	Mandatory	Represents an entry in the access control list for the referenced file share
5.6.4 CIM_CIFSShare (Exported File Share)	Optional	Represents the CIFS sharing characteristics of a particular file element.
5.6.5 CIM_ConcreteDependency	Optional	Represents an association between a (CIFSShare or NFSShare) FileShare element and the actual shared LogicalFile or Directory on which it is based. This is provided for backward compatibility with previous releases of SMI-S.
5.6.6 CIM_ElementCapabilities (FES Configuration)	Mandatory	Associates the File Export Service to at least one ExportedFileShareCapabilities element that indicates that support is available for managing an exported FileShare for at least one of the file sharing protocols recognized by this profile. These include: "NFS"/2, "CIFS"/3, "DAFS"/4, "WebDAV"/5, "HTTP"/6, or "FTP"/7.
5.6.7 CIM_ElementSettingData (FileShare Setting)	Mandatory	Associates a (CIFSShare or NFSShare) FileShare and ExportedFileShareSetting elements.
5.6.8 CIM_ExportedFileShareCapabilities (FES Capabilities)	Mandatory	This element represents the Capabilities of the File Export Service for managing FileShares of a specific file sharing protocol (and version). The file sharing protocol is specified by the FileSharingProtocol and ProtocolVersions properties.
5.6.9 CIM_ExportedFileShareSetting (FileShare Setting)	Mandatory	The configuration settings for an Exported FileShare; i.e., a setting for a FileShare available for exporting.  This setting may have been created or modified by the extrinsic methods of this profile. Note that CIFS allows in-band creation, modification, or deletion of FileShares; also, some systems might define preexistent FileShares. All of these will be surfaced.
5.6.10 CIM_ExportedFileShareSetting (Pre-defined)	Mandatory	This element represents a predefined configuration settings for exported FileShares that is used to define a Capabilities element associated with the FileExportService.
5.6.11 CIM_FileExportCapabilities (FES Configuration)	Mandatory	This element represents the management capabilities of the File Export Service.
5.6.12 CIM_FileExportService	Mandatory	The File Export Service provides the methods to create and export file elements as shares.
5.6.13 CIM_FileShare (Exported File Share)	Mandatory	Represents the sharing characteristics of a particular file element.



**Table 25 - CIM Elements for File Export Manipulation**

Element Name	Requirement	Description
5.6.14 CIM_FileStorage (Subelement)	Conditional	Conditional requirement: Required if parent profile is NAS Head. or Required if parent profile is a Self-contained NAS System.  Represents that a file or directory that is made available for export is contained by a LocalFileSystem specified as a dangling reference.
5.6.15 CIM_HostedService	Mandatory	Associates the File Export Service to the hosting File Server Computer System.
5.6.16 CIM_HostedShare	Mandatory	Represents that a shared element is hosted by a ComputerSystem.
5.6.17 CIM_Identity	Mandatory	Represents a ManagedElement that acts as a security principal within the scope in which it is defined and authenticated
5.6.18 CIM_LogicalFile (Subelement)	Conditional	Conditional requirement: Required if parent profile is NAS Head. or Required if parent profile is a Self-contained NAS System.  A LogicalFile (or Directory subclass) that is a sub-element of a LocalFileSystem that is made available for export via a fileshare hosted on a ComputerSystem. This is included for backward compatibility with previous releases of SMI-S.
5.6.19 CIM_NFSShare (Exported File Share)	Optional	Represents the NFS sharing characteristics of a particular file element.
5.6.20 CIM_SAPAvailableForFileShare	Mandatory	Represents the association between a ServiceAccessPoint to the shared element that is being accessed through that SAP.
5.6.21 CIM_ServiceAffectsElement	Mandatory	Associates the File Export Service to the elements that the service manages (such as a FileShare configured for exporting a LogicalFile).
5.6.22 CIM_SettingsDefineCapabilities (Pre-defined)	Optional	Represents the association between a ExportedFileShareCapabilities and a predefined ExportedFileShareSetting element that specifies what the Capabilities can support.
5.6.23 CIM_SharedElement	Mandatory	Associates a (CIFSShare or NFSShare) FileShare to the LocalFileSystem on which it is based.
5.6.24 CIM_UserContact	Mandatory	Contains the identifying information of a user
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FileShare	Mandatory	Creation of an exported file share.  This indication returns the newly created FileShare.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FileShare	Mandatory	Deletion of an exported file share.  This indication returns the model path to the deleted file share and its unique instance id. (Question: Should this return the pathname of the shared directory as well?) Note that a model path is like a CIM object path but not exactly the same.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileShare AND SourceInstance.CIM_FileShare::OperationalStatus <> PreviousInstance.CIM_FileShare::OperationalStatus	Mandatory	CQL -Change of state of a FileShare.  PreviousInstance is optional, but may be supplied by an implementation of the profile.

### 5.6.1 CIM\_AccountManagementService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 26 describes class CIM\_AccountManagementService

**Table 26 - SMI Referenced Properties/Methods for CIM\_AccountManagementService**

Properties	Flags	Requirement	Description & Notes
CreateUserContact()		Mandatory	Creates a UserContact for the specified system. Upon successful completion of the method, it returns a UserContact instance associated to an instance of Identity through AssignedIdentity. An Identity instance may be created by the method or may be one that already existed that represented the user.
GetUserContacts()		Mandatory	Retrieves instances of UserContact that have the specified UserID. Upon successful completion of the method, it returns an array of UserContact instances that are already associated to instances of Identity.

### 5.6.2 CIM\_AssignedIdentity

Created By: Extrinsic CIM\_AccountManagementService.CreateUserContact

Modified By: N/A

Deleted By: N/A

Requirement: Mandatory

Table 27 describes class CIM\_AssignedIdentity

**Table 27 - SMI Referenced Properties/Methods for CIM\_AssignedIdentity**

Properties	Flags	Requirement	Description & Notes
IdentityInfo	KEY, REF	Mandatory	The Identity instance referenced by this AssignedIdentity
ManagedElement	KEY, REF	Mandatory	The UserContact referenced by this AssignedIdentity

### 5.6.3 CIM\_AssociatedPrivilege

Created By: Extrinsic: CIM\_FileExportService.AssignPrivilegeToExportedShare

Modified By: N/A

Deleted By: Intrinsic: DeleteInstance

Requirement: Mandatory

Table 28 describes class CIM\_AssociatedPrivilege

**Table 28 - SMI Referenced Properties/Methods for CIM\_AssociatedPrivilege**

Properties	Requirement	Description & Notes
UseKey	Mandatory	Used to uniquely identify an AssociatedPrivilege instances among others with the same Subject/Target values. This field shall be populated and persisted by the implementation.
Subject	Mandatory	The ManagedElement (in this case Identity) to which privileges are granted
Target	Mandatory	The ManagedElement (in this case FileShare) to which privileges apply.

**Table 28 - SMI Referenced Properties/Methods for CIM\_AssociatedPrivilege**

Properties	Requirement	Description & Notes
DeleteInstance()	Mandatory	Deletes the AssociatedPrivilege instance, for the purpose of revoking user (identity) access to the resource (file share)
Activities[]	Mandatory	An enumeration indicating the activities that are granted or denied. The values Read (5) and Write (6) must be supported.

#### 5.6.4 CIM\_CIFSShare (Exported File Share)

The CIM\_CIFSShare is a subclass of CIM\_FileShare. It is optional, since an implementation may instantiate either (or both) of CIM\_CIFSShare or CIM\_NFSShare.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 29 describes class CIM\_CIFSShare (Exported File Share).

**Table 29 - SMI Referenced Properties/Methods for CIM\_CIFSShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section 4.6.5 CIM_FileShare (Exported File Share).
ElementName		Mandatory	See the ElementName definition in section 4.6.5 CIM_FileShare (Exported File Share).
Name		Mandatory	See the Name definition in section 4.6.5 CIM_FileShare (Exported File Share).
SharingDirectory		Mandatory	See the SharingDirectory definition in section 4.6.5 CIM_FileShare (Exported File Share).
OperationalStatus		Mandatory	See the OperationalStatus definition in section 4.6.5 CIM_FileShare (Exported File Share).
Description	N	Optional	See the Description definition in section 4.6.5 CIM_FileShare (Exported File Share).

#### 5.6.5 CIM\_ConcreteDependency

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Optional

Table 30 describes class CIM\_ConcreteDependency.

**Table 30 - SMI Referenced Properties/Methods for CIM\_ConcreteDependency**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The LogicalFile that is being shared.
Dependent		Mandatory	The (CIFSShare or NFSShare) Share that represents the LogicalFile being shared.

### 5.6.6 CIM\_ElementCapabilities (FES Configuration)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 31 describes class CIM\_ElementCapabilities (FES Configuration).

**Table 31 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (FES Configuration)**

Properties	Flags	Requirement	Description & Notes
Characteristics		Mandatory	If this array enum includes the value mapped to "Default", it indicates that the ExportedFileShareCapabilities element identified by this association is the default to be used for any extrinsic method of the associated FileExportService element.
Capabilities		Mandatory	The FileExportCapabilities. The FileSharingProtocol in these capabilities shall be 2 (NFS), 3 (CIFS), 4 (DAFS), 5 (WebDAV), 6 (HTTP) or 7 (FTP).
ManagedElement		Mandatory	The FileExportService.

### 5.6.7 CIM\_ElementSettingData (FileShare Setting)

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 32 describes class CIM\_ElementSettingData (FileShare Setting).

**Table 32 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (FileShare Setting)**

Properties	Flags	Requirement	Description & Notes
IsCurrent	N	Optional	Is always true in this version of the profile because we only support one setting per share. However support for the other flags, specifically, IsDefault and IsNext, could be added in future releases.

**Table 32 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (FileShare Setting)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The (CIFSShare or NFSShare) FileShare used for exporting an element.
SettingData		Mandatory	A Setting that specifies possible configurations of the FileShare. In this version, we default this to isCurrent="true".

**5.6.8 CIM\_ExportedFileShareCapabilities (FES Capabilities)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 33 describes class ExportedFileShareCapabilities (FES Capabilities)

**Table 33 - SMI Referenced Properties/Methods for CIM\_ExportedFileShareCapabilities (FES Capabilities)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for a capability of a File Export Service.
ElementName		Mandatory	A provider supplied user-Friendly Name for this Capabilities element.
FileSharingProtocol		Mandatory	This identifies the single file sharing protocol (e.g., NFS or CIFS) that this Capabilities represents.
ProtocolVersions		Mandatory	An array of strings listing the versions of the file sharing protocol that can be supported via these capabilities. All settings associated with this Capabilities must have a non-empty subset of these values in the corresponding CIM_ExportedFileShareSetting.ProtocolVersions property. At this point there is no standard mechanism for naming versions of CIFS or NFS.
SupportedProperties		Mandatory	This is the list of configuration properties (of ExportedFileShareSetting) that are supported for specification at creation time by this Capabilities element.  Properties that can appear in this array are: "DefaultReadWrite" ("2"), "DefaultExecute" ("3"), "DefaultUserId" ("4"), "RootAccess" ("5"), "WritePolicy" ("6"), "AccessPoints" ("7"), and "InitialEnabledState" ("8").
CASupported		Optional	This property applies to CIFS/SMB shares only. If it is true, it means that "Continuous Availability" is supported for CIFS shares. Continuous Availability (CA) - Client/Server mediated recovery from network and server failure with application transparency. Like Multi-Channel IO, this feature is somewhat analogous to capabilities available in NFSv4.
CreateGoalSettings()		Mandatory	This extrinsic method supports the creation of a ExportedFileShareSetting that is a supported variant of a ExportedFileShareSetting passed in as an embedded IN parameter TemplateGoalSettings[0]. The method returns the supported ExportedFileShareSetting as an embedded OUT parameter SupportedGoalSettings[0].

**5.6.9 CIM\_ExportedFileShareSetting (FileShare Setting)**

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 34 describes class ExportedFileShareSetting (FileShare Setting).

**Table 34 - SMI Referenced Properties/Methods for CIM\_ExportedFileShareSetting (FileShare Setting)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique ID for the Setting.
ElementName		Mandatory	A client-defined user-friendly name for the Setting.
FileSharingProtocol		Mandatory	The file sharing protocol supported by this share. NFS (2) and CIFS (3) are the supported values.
ProtocolVersions		Mandatory	An array of strings listing the versions of the file sharing protocol that this share can support. This property shall exist and have at least one entry. At this point there is no standard mechanism for naming versions of CIFS or NFS.
InitialEnabledState	N	Optional	<p>This indicates the enabled/disabled states initially set for a created FileShare by this Setting element.</p> <p>Valid values are "1" ("Other"), "2" ("Enabled"), "3" ("Disabled"), "7" ("In Test"), "8" ("Deferred") or "9" ("Quiesce")</p> <p>Note: We need to rethink the usage of this property once the file share has been created. Maybe it should apply to when the file share is re-activated when the share or system is rebooted after a shutdown. With the current definition, neither this nor OtherEnabledState make sense.</p>
OtherEnabledState	N	Optional	This should be filled in if the InitialEnabledState is "1" ("Other").
DefaultUserIdSupported	N	Optional	<p>Indicates whether the associated FileShare will use a default user id to control access to the share if the id of the importing client is not provided.</p> <p>Note: The resulting access privileges shall be surfaced using the Simple Identity Management Profile.</p> <p>Valid values are "2" ("No Default User Id"), "3" ("System-Specified Default User Id") or "4" ("Share-Specified Default User Id").</p>
RootAccess	N	Optional	<p>Indicates whether the associated FileShare will support default access privileges to administrative users from specified hosts.</p> <p>Valid values are "2" ("No Root Access") or "3" ("Allow Root Access").</p>
AccessPoints	N	Optional	<p>An enumerated value that specifies the service access points that are available to this FileShare element by default (to be used by clients for connections). Any ServiceAccessPoint elements that actually connect to this FileShare element will be associated to it by a CIM_SAPAvailableForFileShare association.</p> <p>Valid values are "2" ("None"), "3" ("Service Default"), "4" ("All") or "5" ("Named Points").</p>

#### 5.6.10 CIM\_ExportedFileShareSetting (Pre-defined)

Created By: Static\_or\_External

Modified By: External  
Deleted By: External  
Requirement: Optional

Table 35 describes class ExportedFileShareSetting (Pre-defined).

**Table 35 - SMI Referenced Properties/Methods for CIM\_ExportedFileShareSetting (Pre-defined)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for this Setting element.
ElementName		Mandatory	A provider supplied user-friendly name for this Setting element.
FileSharingProtocol		Mandatory	The file sharing protocol to which this Setting element applies. The entries in the ProtocolVersions property identify the specific versions of the protocol that are supported. This profile only supports "NFS" (2) and "CIFS" (3).
ProtocolVersions		Mandatory	An array of strings listing the versions of the file sharing protocol that this share can support. This property shall exist and have at least one entry. At this point there is no standard mechanism for naming versions of CIFS or NFS.
InitialEnabledState		Optional	This indicates the enabled/disabled states initially set for a created FileShare by this Setting element.  Valid values are "1" ("Other"), "2" ("Enabled"), "3" ("Disabled"), "7" ("In Test"), "8" ("Deferred") or "9" ("Quiesce").
OtherEnabledState		Optional	A vendor-specific description of the initial enabled state of a created fileshare if InitialEnabledState=1("Other").
DefaultUserIdSupported		Optional	Indicates whether a FileShare created or modified by using this Setting element will use a default user id to control access to the share if the id of the importing client is not provided.  Note: The resulting access privileges shall be surfaced using the Simple Identity Managment Profile.  Valid values are "2" ("No Default User Id"), "3" ("System-Specified Default User Id") or "4" ("Share-Specified Default User Id").
RootAccess		Optional	Indicates whether a FileShare created or modified by using this Setting element will support default access privileges to administrative users from specific hosts specified at creation time.  Valid values are "2" ("No Root Access") or "3" ("Allow Root Access").
AccessPoints		Optional	An enumerated value that specifies the service access points that are available to a FileShare created or modified by using this Setting element by default (to be used by clients for connections). These default access points can always be overridden by the privileges explicitly defined by a supported authorization mechanism(s). Any ServiceAccessPoints that actually connect to this share will be associated to it by CIM_SAPAvailableForFileShare.  Valid values are "2" ("None"), "3" ("Service Default"), "4" ("All") or "5" ("Named Points").
CASupported		Optional	This property applies to CIFS/SMB shares only. If it is true, it means that "Continuous Availability" is supported for CIFS shares. Continuous Availability (CA) - Client/Server mediated recovery from network and server failure with application transparency. Like Multi-Channel IO, this feature is somewhat analogous to capabilities available in NFSv4.

**Table 35 - SMI Referenced Properties/Methods for CIM\_ExportedFileShareSetting (Pre-defined)**

Properties	Flags	Requirement	Description & Notes
DefaultReadWrite		Optional	Indicates the default privileges that are supported for read and write authorization when creating or modifying a FileShare using this Setting element.
DefaultExecute		Optional	Indicates the default privileges that are supported for execute authorization when creating or modifying a FileShare using this Setting element.
ExecuteSupport		Optional	Indicates if the sharing mechanism provides specialized support for executing a shared element when creating or modifying a FileShare using this Setting element (for instance, does it provide paging support for text pages).
WritePolicy		Optional	Indicates whether writes through a FileShare (created or modified by using this Setting element) to the shared element will be handled synchronously or asynchronously by default.

**5.6.11 CIM\_FileExportCapabilities (FES Configuration)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 36 describes class FileExportCapabilities (FES Configuration).

**Table 36 - SMI Referenced Properties/Methods for CIM\_FileExportCapabilities (FES Configuration)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the capabilities of a File Export Service.
ElementName		Mandatory	A provider supplied user-friendly name for this Capabilities element.
FileSharingProtocol		Mandatory	An array listing all the protocols for file sharing supported by the FileExportService represented by this FileExportCapabilities element. Duplicate entries are permitted because the corresponding entry in the ProtocolVersions array property indicates the supported version of the protocol.  Each entry must correspond to an ExportedFileShareCapabilities element associated via ElementCapabilities to the FileExportService -- the FileSharingProtocol and ProtocolVersion properties of that element must match the entry.
ProtocolVersions		Mandatory	An array of strings listing the versions of the file sharing protocol in the corresponding CIM_ExportedFileShareCapabilities.ProtocolVersions property that can be supported by this FileExportService. At this point there is no standard mechanism for naming versions of CIFS or NFS.



**Table 36 - SMI Referenced Properties/Methods for CIM\_FileExportCapabilities (FES Configuration)**

Properties	Flags	Requirement	Description & Notes
SupportedSynchronousMethods	N	Mandatory	An array listing the extrinsic methods of the FileExportService that can be called synchronously.  Note: Every supported method shall be listed either in this property or in the SupportedAsynchronousMethods array property.
SupportedAsynchronousMethods	N	Mandatory	An array listing the extrinsic methods of the FileExportService that can be called synchronously.  Note: Every supported method shall be listed either in this property or in the SupportedSynchronousMethods array property.
InitialEnabledState		Optional	This represents the state of initialization of a FileShare on initial creation.
CASupported		Optional	This property applies to CIFS/SMB shares only. If it is true, it means that "Continuous Availability" is supported for CIFS shares. Continuous Availability (CA) - Client/Server mediated recovery from network and server failure with application transparency. Like Multi-Channel IO, this feature is somewhat analogous to capabilities available in NFSv4.

**5.6.12 CIM\_FileExportService**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 37 describes class FileExportService.

**Table 37 - SMI Referenced Properties/Methods for CIM\_FileExportService**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A provider supplied user-friendly name for this Service.
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the Service.
SystemName		Mandatory	The name of the Computer System hosting the Service.
CreationClassName		Mandatory	The CIM Class name of the Service.
Name		Mandatory	The unique name of the Service.
CreateExportedShare()		Mandatory	Create a FileShare element configured for exporting a file or directory as a share.
ModifyExportedShare()		Mandatory	Modify the configuration of a FileShare element setup to export a file or directory as a share.
ReleaseExportedShare()		Mandatory	Delete the FileShare element that is exporting a file or directory as a share, thus releasing that element.
AssignPrivilegeToExportedShare()		Mandatory	Grants the specified Identities permissions to the specified FileShare specified Creates any AssociatedPrivilege instances that are necessary to represent the requested permissions.

**5.6.13 CIM\_FileShare (Exported File Share)**

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 38 describes class CIM\_FileShare (Exported File Share).

**Table 38 - SMI Referenced Properties/Methods for CIM\_FileShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	A unique id for the FileShare element.
ElementName		Mandatory	This shall be a user friendly name for the FileShare.
Name		Mandatory	This shall be an opaque string that uniquely identifies the path to the directory or file.
SharingDirectory		Mandatory	Indicates if the shared element is a file or a directory. This is useful when importing but less so when exporting.
OperationalStatus		Mandatory	The OperationalStatus of the FileShare as defined in the Health and Fault Management Clause.

**5.6.14 CIM\_FileStorage (Subelement)**

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Required if parent profile is NAS Head. or Required if parent profile is a Self-contained NAS System.

Table 39 describes class CIM\_FileStorage (Subelement).

**Table 39 - SMI Referenced Properties/Methods for CIM\_FileStorage (Subelement)**

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	The file or directory that is made available for export.
GroupComponent		Mandatory	The local filesystem that contains the exported file or directory.

**5.6.15 CIM\_HostedService**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 40 describes class CIM\_HostedService.

**Table 40 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting Computer System.
Dependent		Mandatory	The FileExportService.

#### 5.6.16 CIM\_HostedShare

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 41 describes class CIM\_HostedShare.

**Table 41 - SMI Referenced Properties/Methods for CIM\_HostedShare**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The CIFS or NFS share that is hosted by a Computer System.
Antecedent		Mandatory	The Computer System that hosts a FileShare. It may be any system, but the system shall have Dedicated=16 (File Server).

#### 5.6.17 CIM\_Identity

Created By: Extrinsic: CIM\_AccountManagementService.CreateUserContact

Modified By: N/A

Deleted By: N/A

Requirement: Mandatory

Table 42 describes class CIM\_Identity.

**Table 42 - SMI Referenced Properties/Methods for CIM\_Identity**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key

#### 5.6.18 CIM\_LogicalFile (Subelement)

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Required if parent profile is NAS Head. or Required if parent profile is a Self-contained NAS System.

Table 43 describes class CIM\_LogicalFile (Subelement).

**Table 43 - SMI Referenced Properties/Methods for CIM\_LogicalFile (Subelement)**

Properties	Flags	Requirement	Description & Notes
CSCreationClassName		Mandatory	CIM Class of the Computer System that hosts the filesystem of this File.
CSName		Mandatory	Name of the Computer System that hosts the filesystem of this File.
FSCreationClassName		Mandatory	CIM Class of the LocalFileSystem on the Computer System that contains this File.
FSName		Mandatory	Name of the LocalFileSystem on the Computer System that contains this File.
CreationClassName		Mandatory	CIM Class of this instance of LogicalFile.
Name		Mandatory	Name of this LogicalFile.

#### 5.6.19 CIM\_NFSShare (Exported File Share)

The CIM\_NFSShare is a subclass of CIM\_FileShare. It is optional, since an implementation may instantiate either (or both) of CIM\_CIFSShare or CIM\_NFSShare.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 44 describes class CIM\_NFSShare (Exported File Share).

**Table 44 - SMI Referenced Properties/Methods for CIM\_NFSShare (Exported File Share)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section 4.6.5 CIM_FileShare (Exported File Share).
ElementName		Mandatory	See the ElementName definition in section 4.6.5 CIM_FileShare (Exported File Share).
Name		Mandatory	See the Name definition in section 4.6.5 CIM_FileShare (Exported File Share).
SharingDirectory		Mandatory	See the SharingDirectory definition in section 4.6.5 CIM_FileShare (Exported File Share).
OperationalStatus		Mandatory	See the OperationalStatus definition in section 4.6.5 CIM_FileShare (Exported File Share).
Description	N	Optional	See the Description definition in section 4.6.5 CIM_FileShare (Exported File Share).

#### 5.6.20 CIM\_SAPAvailableForFileShare

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 45 describes class CIM\_SAPAvailableForFileShare.

**Table 45 - SMI Referenced Properties/Methods for CIM\_SAPAvailableForFileShare**

Properties	Flags	Requirement	Description & Notes
FileShare		Mandatory	The element that is made available through a SAP. In the File Export Profile, these are (CIFSShare or NFSShare) FileShares configured for either export.
AvailableSAP		Mandatory	The ProtocolEndpoint that is available to this (CIFSShare or NFSShare) FileShare. This shall be 4200 (NFS) or 4201 (CIFS).

#### 5.6.21 CIM\_ServiceAffectsElement

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 46 describes class CIM\_ServiceAffectsElement.

**Table 46 - SMI Referenced Properties/Methods for CIM\_ServiceAffectsElement**

Properties	Flags	Requirement	Description & Notes
ElementEffects		Mandatory	In this profile, the service provides management for the element. We allow Other to support vendor extensions. The standard values are 1 (Other) and 5 (Manages).
OtherElementEffectsDescriptions		Mandatory	A description of other element effects that this association might be exposing.
AffectedElement		Mandatory	The (CIFSShare or NFSShare) FileShare.
AffectingElement		Mandatory	The FileExportService.

#### 5.6.22 CIM\_SettingsDefineCapabilities (Pre-defined)

Created By: Static\_or\_External

Modified By: External

Deleted By: External

Requirement: Optional

Table 47 describes class CIM\_SettingsDefineCapabilities (Pre-defined).

**Table 47 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (Pre-defined)**

Properties	Flags	Requirement	Description & Notes
PropertyPolicy		Mandatory	Used for the purpose of identifying "default" settings, as described in 5.1.3.1
ValueRole		Mandatory	
ValueRange		Mandatory	

**Table 47 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (Pre-defined)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	An Exported FileShare Capabilities element that is defined by a collection of ExportedFileShareSetting Settings.
PartComponent		Mandatory	A Exported FileShare Setting that provides a point or a partial definition for a Exported FileShare Capabilities element.

**5.6.23 CIM\_SharedElement**

Created By: Extrinsic: CreateExportedShare

Modified By: Extrinsic: ModifyExportedShare

Deleted By: Extrinsic: ReleaseExportedShare

Requirement: Mandatory

Table 48 describes class CIM\_SharedElement.

**Table 48 - SMI Referenced Properties/Methods for CIM\_SharedElement**

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	The LocalFileSystem that is sharing a directory or file through a FileShare alter ego.
SameElement		Mandatory	The (CIFSShare or NFSShare) FileShare that is the alter ego for a directory or file in a LocalFileSystem.

**5.6.24 CIM\_UserContact**

Created By: Extrinsic: CIM\_AccountManagementService.CreateUserContact

Modified By: N/A

Deleted By: N/A

Requirement: Mandatory

Table 49 describes class CIM\_UserContact.

**Table 49 - SMI Referenced Properties/Methods for CIM\_UserContact**

Properties	Flags	Requirement	Description & Notes
UserID		Mandatory	The login name for the user represented by UserContact. Implementations shall persist this value exactly as provided to CreateUserContact
Name		Mandatory	Name that uniquely identifies the UserContact instance. Whenever feasible, implementations should persist the value provided to CreateUserContact

---

---

**STABLE**

---

---

## STABLE

### 6 File Server Manipulation Profile

#### 6.1 Description

##### 6.1.1 Synopsis

**Profile Name:** File Server Manipulation (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** FileServerConfigurationService

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 50 describes the supported profiles for File Server Manipulation.

**Table 50 - Supported Profiles for File Server Manipulation**

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

##### 6.1.2 Overview

The File Server Manipulation Profile is a component profile of autonomous profiles that support filesystems. It makes use of elements of the filesystem component profiles and supports creation and deletion of Virtual File Servers and the modification of both virtual and non-virtual File Servers. A number of other profiles also make use of elements of the Filesystem Profile and will be referred to in this specification as “filesystem-related profiles” -- these include, but are not limited to, the Filesystem Profile, the Filesystem Manipulation Profile, the File Export Profile, the NAS Head Profile, the Self-Contained NAS Profile.

In this release of SMI-S, the autonomous profiles that use the File Server Manipulation Profile are the NAS Head and Self-Contained NAS Profiles.

A File Server is a computer system that is attached to a network and provides resources to allow client systems access to filesystem resources in the form of CIFS Shares and/or NFS Exports. A File Server can be either a physical computer system or can be a virtual system that is hosted by a physical computer system. A physical File Server can neither be created nor deleted but may have properties that can be modified via configuration actions. A virtual File Server can be created, deleted, and modified via configuration actions. The number of virtual File Servers that may be created is system dependent. This profile models both physical and virtual File Servers. Extrinsic methods are provided for the creation and deletion of virtual File Servers. Extrinsic methods are also provide for the modification of properties in both physical and virtual File Servers.

This profile supports viewing and configuring the following property “areas” of a File Server:

- NFS Exports
- CIFS Shares
- Ethernet port properties including VLAN tagging.

- DNS Settings
- NIS Settings

A given implementation may choose to support a strict read only view of the File Server configuration or may provide any combination of capabilities for modifying any and all of the above property areas for the File Server.

Throughout this profile, the term File Server will be synonymous for “ComputerSystem with Dedicated[]=”FileServer”. The term virtual File Server describes a File Server that has its “USAGE” property set to “Virtual File Server”. A non-virtual (physical) File Server cannot have its “USAGE” property set to “Virtual File Server”.

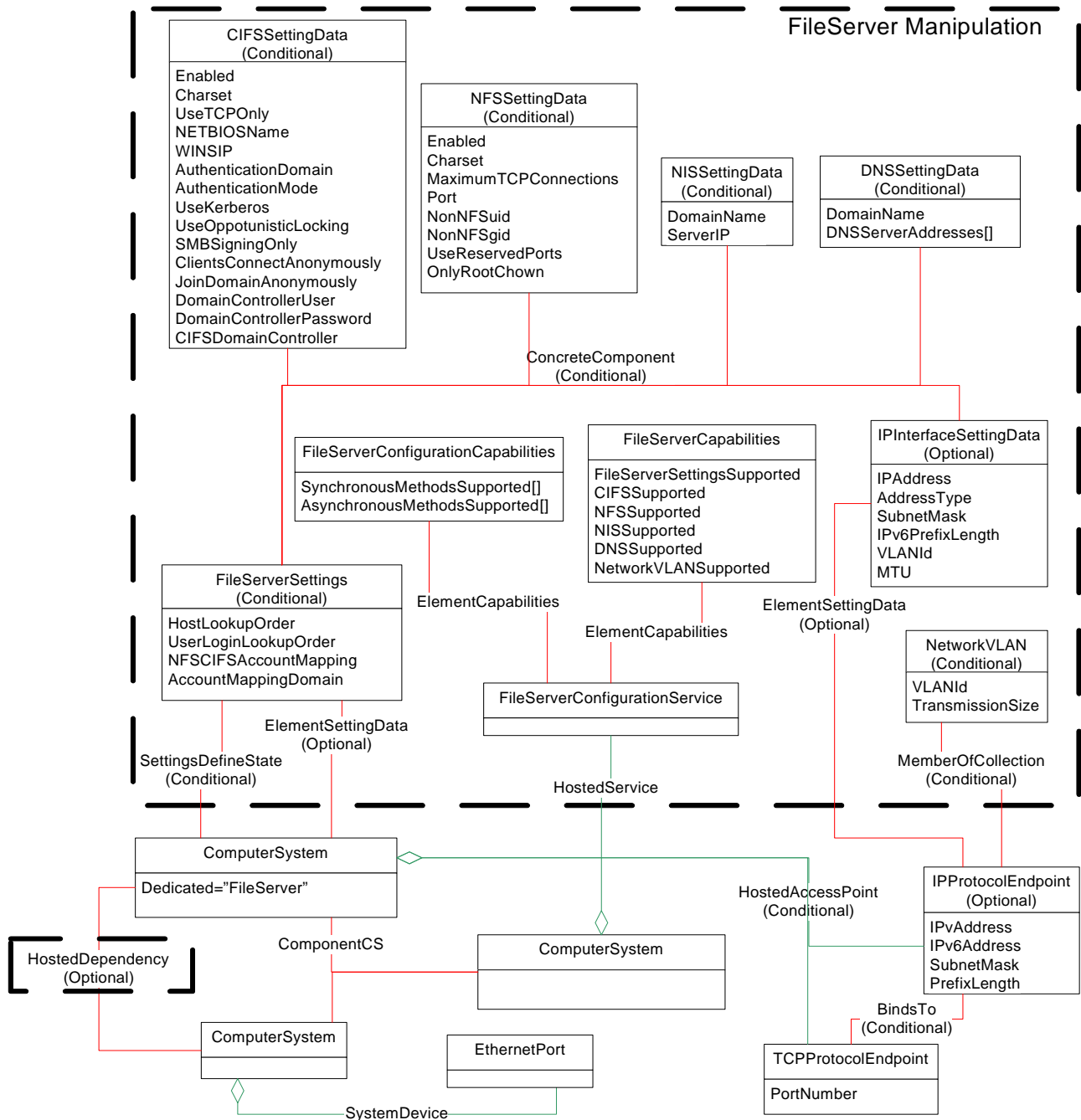
The profile models a File Server from a “read only” perspective and a “configuration” perspective. The read only perspective defines the objects and attributes that describe a File Server instance. The configuration perspective defines the permitted actions on the File Server for creating, deleting, and modifying instances. By providing these two perspectives, this profile takes the place of having two separate profiles.

### **6.1.3 Instance Diagrams**

#### **6.1.3.1 File Server classes and associations (read-only view)**

Figure 9 illustrates the constructs that are involved in defining a File Server. This summarizes the “read only” view of the classes and associations for this profile.



**Figure 9 - File Server Classes and Associations (Read only view)**

The File Server is modeled as a ComputerSystem whose Dedicated property is set to "FileServer" (16). There are two types of File Servers supported: Virtual File Servers and non-Virtual File Servers (which would be a physical File Server).

A Virtual File Server will have a HostedDependency association on another top level Computer System such as a NAS Head or Self-Contained NAS for example. This top level ComputerSystem has a HostedService association with FileServerConfigurationService, which provides the anchor point for the FileServerConfigurationCapabilities and FileServerCapabilities. These capabilities identify the level of support for File Servers by an implementation. For example, if the SynchronousMethodsSupported and

AsynchronousMethodsSupported are empty or NULL, then the implementation is a read-only implementation of the profile.

A File Server can also be the top level ComputerSystem. In that case, the Dedicated array would contain "FileServer" and either "NAS Head" or "Self-contained NAS". In this case, the File Server would be considered a non-Virtual File Server.

A Virtual File Server is hosted on a ComputerSystem. This may be a physical control unit or some other hardware system that has the EthernetPort through which the File Server will serve files via CIFS and/or NFS. The HostedDependency association is used to relate the Virtual File Server with the hosting ComputerSystem.

A non-Virtual File Server shall not have a HostedDependency association with another ComputerSystem. Instead, if the File Server ComputerSystem is not the top level system, then it shall have a ComponentCS association with the top level ComputerSystem.

FileServerSettings captures the settings of the File Server. It has ConcreteComponent associations with other setting data that capture the File Server's settings for CIFS, NFS, NIS, DNS, and its IP Interface(s). The minimal implementation only needs to support the File Server ComputerSystem because the FileServerSettings is conditionally supported. The FileServerCapabilities contains several booleans that tell a client the set of File Server related features that an implementation supports. The conditional associations associated with FileServerSettings are based on the values for these booleans.

The File Server has two separate associations with FileServerSettings. SettingsDefineState is used to represent the current state of the File Server's setting data while ElementSettingData is used to capture the setting data used to initially create or modify the File Server. In the read-only case, there will be no ElementSettingData association.

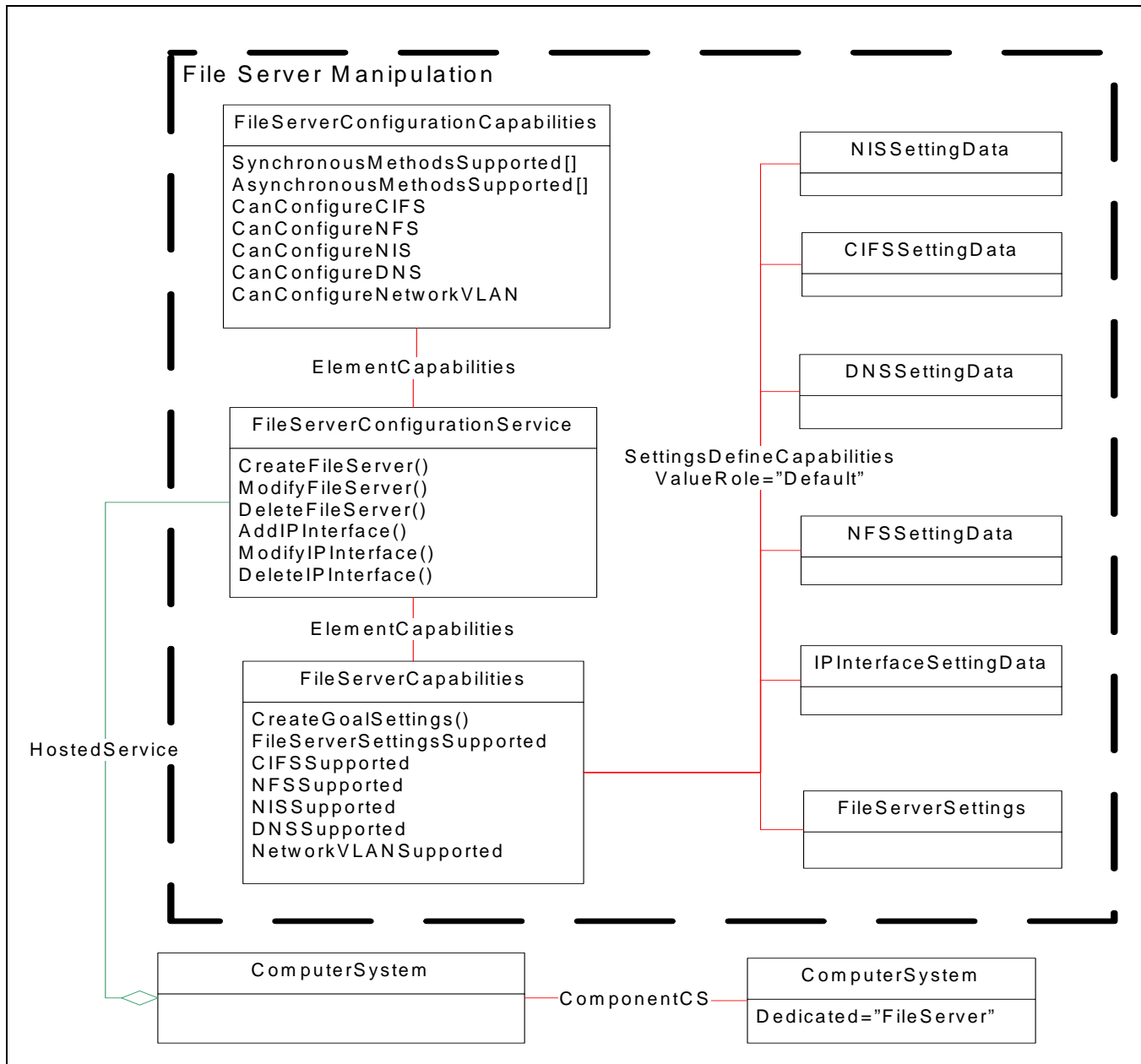
NOTE There is only an ElementSettingData between the IPInterfaceSettingData and the IPProtocolEndpoint. The IPProtocolEndpoint has at most one IPInterfaceSettingData and it represents the settings used to initially create or modify the IPProtocolEndpoint. Also note that multiple (CIFS or NFS) ProtocolEndpoints may be bound to a single IPProtocolEndpoint.

The NISSettingData and DNSSettingData if present are used to resolve hosts and user names when authenticating hosts and users.

The implementation can provide either a read-only view of the File Servers or may provide extrinsics for configuring existing and/or new File Servers.

A client can determine if a read-only implementation is provided by inspecting the two FileServerConfigurationCapabilities arrays SynchronousMethodsSupported and AsynchronousMethodsSupported. If they are both empty or null, then the implementation is read-only.

### 6.1.3.2 File Server Configuration classes and associations



**Figure 10 - File Server Configuration classes and association**

Figure 10 illustrates the constructs that are involved in configuring a File Server.

The top level ComputerSystem has a HostedService association with FileServerConfigurationService that defines the extrinsics that can be used to manage a File Server. There are 3 methods for managing a File Server and 3 methods for managing additional IPInterfaces for a given File Server.

FileServerConfigurationCapabilities lists the extrinsics that can be called synchronously or asynchronously. It is associated with the FileServerConfigurationService via the ElementCapabilities association. It also has several boolean properties that inform clients if the implementation is able to configure CIFS, NFS, NIS, DNS, and VLAN Tagging.

In addition to the set of booleans that indicate the set of File Server features supported by the implementation, `FileServerCapabilities` also provides one method `CreateGoalSettings` that can be used to arrive at a set of viable `SettingData` instances that can be used for creating or modifying a File Server. It also is associated with `FileServerConfigurationService` via `ElementCapabilities`. It may have associations with `SettingData` instances that reflect the Default settings for the File Server. The `SettingsDefineCapabilities` association (with `ValueRole="Default"`) is used to capture these default `SettingData` instances.

Only Virtual File Servers can be created or deleted. Non-Virtual File Servers can have properties modified, but cannot be deleted.

The extrinsic methods that create Virtual File Servers can take any combination of `SettingData` instances that are used to instantiate the File Server. The implementation can remember these initial `SettingData` instances via the `ElementSettingData` association between the File Server and `FileServerSettings`. After the File Server is created, the `SettingsDefineState` association between the File Server and `FileServerSettings` defines the actual settings of the File Server. Modifications to either a Virtual or non-Virtual File Server will be reflected in the `SettingData` instances associated via the `SettingsDefineState` association. A non-Virtual File Server may not have `SettingData` instances associated via the `ElementSettingData` association.

The `FileServerConfigurationCapabilities` instance contains several booleans that indicate if certain properties of the File Server can be configured or modified. For those properties that cannot be configured/modified, attempting to instantiate or modify them via a creation/modification extrinsic shall be an error.

If neither `CIFSSettingData` nor `NFSSettingData` are specified at creation time, and the implementation supports either or both of them, then instances shall be created by the implementation based on the settings in `FileServerCapabilities`. The "Enabled" property of the instances created will be set to "false".

When a Virtual File Server is created or when it has additional `IPInterfaces` associated with it, an instance of `NetworkVLAN` may be created if VLAN tagging should be associated with the `IPInterface`. `NetworkVLAN` instances are associated with the specific `IPProtocolEndpoint` to capture the VLAN tag to be used when doing I/O on that IP interface. The properties `VLANid` and `MTU` in `IPInterfaceSettingData` specify the values to use when creating the `NetworkVLAN` instance.

## **6.2 Health and Fault Management Consideration**

### **6.2.1 OperationalStatus for File Server ComputerSystem**

This section describes the operational status for Virtual File Servers. Non-Virtual File Server operation status information is covered in both the NAS Head and Self-Contained NAS Profiles.

A File Server's operational status will be influenced by the operational status of the `ComputerSystem` that is hosting it via `HostedDependency`. For example, if the hosting `ComputerSystem` is "Stopped", then the status of the File Server will be "Stopped". Providers must take this into account when formulating the status of the File Server.

Table 51 describes the operational status for File Server `ComputerSystem`.

**Table 51 - Operational Status for File Server ComputerSystem**

Primary OperationalStatus	Description
2 "OK"	The File Server is running with good status
3 "Degraded"	The File Server is operating in a degraded mode. This could be due to the health state of some component of the ComputerSystem, due to load by other applications, or due to the health state of backend or front-end network interfaces.
4 "Stressed"	The File Server resources are stressed
5 "Predictive Failure"	The File Server might fail because some resource or component is predicted to fail
6 "Error"	An error has occurred causing the File Server to become unavailable. Operator intervention through SMI-S to restore the service may be possible.
6 "Error"	An error has occurred causing the File Server to become unavailable. Automated recovery may be in progress.
7 "Non-recoverable Error"	The File Server is not functioning. Operator intervention through SMI-S will not fix the problem.
8 "Starting"	The File Server is in process of initialization and is not yet available operationally.
9 "Stopping"	The File Server is in process of stopping, and is not available operationally.
10 "Stopped"	The File Server cannot be accessed operationally because it is stopped -- if this did not happened because of operator intervention or happened in real-time, the OperationalStatus would have been "Lost Communication" rather than "Stopped".
11 "In Service"	The File Server is offline in maintenance mode, and is not available operationally.
13 "Lost Communications"	The File Server cannot be accessed operationally -- if this happened because of operator intervention it would have been "Stopped" rather than "Lost Communication".
14 "Aborted"	The File Server is stopped but in a manner that may have left it in an inconsistent state.
15 "Dormant"	The File Server is offline; and the reason for not being accessible is unknown.
16 "Supporting Entity in Error"	The File Server is in an error state, or may be OK but not accessible, because a supporting entity is not accessible.

### 6.3 Cascading Considerations

Not Applicable.

### 6.4 Methods of the Profile

This section describes each extrinsic method supported by this profile.

#### 6.4.1 Extrinsic Methods

##### 6.4.1.1 FileSystemCapabilities.CreateGoalSettings

This extrinsic method of the FileSystemCapabilities class validates support for a caller-proposed Settings.

The client shall pass six array elements in the TemplateGoalSettings parameter and six array elements in the SupportedGoalSettings parameter. Each array element represents a configurable aspect of a FileServer. A given array element in index “y” in TemplateGoalSettings will be of the same class/type as that in array element in index “y” in SupportedGoalSettings. As each array element in both parameters takes an EmbeddedInstance, this implies that they do not exist in the provider’s implementation but are the responsibility of the client to create and manage.

Any or all of the TemplateGoalSetting array elements may be the empty string to represent a NULL entry. This method will return a default CIM\_Settings subclass object in SupportedGoalSettings corresponding to each TemplateGoalSettings array element that is an empty string.

If any of the TemplateGoalSettings array elements specify values that cannot be supported, this method shall return an appropriate error and should return a best match in the corresponding SupportedGoalSettings array element.

When providing EmbeddedInstances as input for any of the SupportedGoalSettings array elements, the instance should specify a previously returned CIM\_Setting that the implementation could support. On output, this same array element specifies a new CIM\_Setting that the implementation can support. If the output array element is identical to the input array element, both client and implementation may conclude that this is the best match for that particular SupportedGoalSettings array element. If the output array elements do not match the corresponding TemplateGoalSettings array elements and if any of the input SupportedGoalSettings array elements do not match the output array elements provided in SupportedGoalSettings, then the method must return "Alternative Proposed". If any of the output array elements are empty strings (representing the fact that no valid CIM\_Setting could be found), the method must return an “Failed”.

The client and the implementation can engage in a negotiation process that may require iterative calls to this method. As stated above, to assist the implementation in tracking the progress of the negotiation, the client may pass previously returned values of SupportedGoalSettings array elements as new input values of SupportedGoalSettings. The implementation may determine that a step has not resulted in progress if the input and output values of any SupportedGoalSettings array elements are the same. A client may infer from the same result that the TemplateGoalSettings array element(s) must be modified.

The array elements in TemplateGoalSettings and SupportedGoalSettings shall have the index - EmbeddedInstance mappings shown in Table 52.

**Table 52 - Array Element Mappings for TemplateGoalSettings and SupportedGoalSettings**

Array Indice	EmbeddedInstance
0	FileServerSettings
1	IPInterfaceSettingData
2	CIFSSettingData
3	NFSSettingData
4	NISSettingData
5	CIM_DNSSettingData

Table 53 details of the method signature and return results.

**Table 53 - Parameters for Extrinsic Method FileServerCapabilities.CreateGoalSettings**

Parameter Name	Qualifier	Type	Description & Notes
TemplateGoalSettings[]	IN	string	<p>This contains an array of 6 elements, each of which being an EmbeddedInstance of a CIM_Setting subclass.</p> <p>Each of the array elements shall contain either an empty string to represent a "NULL" entry, or shall contain an EmbeddedInstance.</p> <p>Each array element contains a specific CIM_Setting subclass as follows:</p> <p>0: EmbeddedInstance ("CIM_FileServerSettings")</p> <p>1: EmbeddedInstance ("CIM_IPInterfaceSettingData")</p> <p>2: EmbeddedInstance ("CIM_CIFSSettingData")</p> <p>3: EmbeddedInstance ("CIM_NFSSettingData")</p> <p>4: EmbeddedInstance ("CIM_NISSettingData")</p> <p>5: EmbeddedInstance ("CIM_DNSSettingData")</p>
SupportedGoalSettings[]	INOUT	string	<p>This contains an array of 6 elements, each of which being an EmbeddedInstance of a CIM_Setting subclass.</p> <p>On input, each of the array elements shall contain either an empty string to represent a "NULL" entry, or shall contain an EmbeddedInstance. If it contains an EmbeddedInstance, then this instance specifies a previously returned CIM_Setting that the implementation could support. On output, it specifies a new CIM_Setting that the implementation can support.</p> <p>Each array element contains a specific CIM_Setting subclass as follows:</p> <p>0: EmbeddedInstance ("CIM_FileServerSettings")</p> <p>1: EmbeddedInstance ("CIM_IPInterfaceSettingData")</p> <p>2: EmbeddedInstance ("CIM_CIFSSettingData")</p> <p>3: EmbeddedInstance ("CIM_NFSSettingData")</p> <p>4: EmbeddedInstance ("CIM_NISSettingData")</p> <p>5: EmbeddedInstance ("CIM_DNSSettingData")</p>
Normal Return			
Status		uint32	<p>ValueMap{}, Values{}</p> <p>"Success",            "Not Supported",            "Unknown",            "Failed",            "Timeout",            "Invalid Parameter",            "Alternative Proposed"</p>
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

#### 6.4.1.2 Signature and Parameters of FileServerConfigurationService.CreateFileServer

This extrinsic creates a new FileServer. The method takes several “goal” parameters that represent different configurable aspects of the FileServer. Each of these parameters can be NULL, an empty string, or will contain an EmbeddedInstance.

If a given parameter is NULL or an empty string, a default instance will be selected by the provider using the corresponding element associated to the FileServerConfigurationService by the DefaultElementCapabilities association. This element that is used will be returned in the parameter.

When creating a new FileServer, the client can decide to what degree the new FileServer will be configured by providing the parameters of those aspects that should be configured. For example, to create a FileServer with a minimum configuration, the client could provide just the ElementName. The newly created FileServer will take on the configuration defaults as specified by the elements associated with FileServerService via the SettingsDefineCapabilities association (with ValueRole=”Default”). Later, the client may modify any of these default settings via the ModifyFileServer and ModifyIPInterface methods.

When creating a new FileServer, the client may associate a single IP Interface with the FileServer. If a client wishes to associate more than one IP Interface with the FileServer, the AddIPInterface method should be used. It allows the client to specify the additional IP information, Hosting ComputerSystem, and EthernetPort for the new IP Interface.

A client may change an existing IP Interface by using the ModifyIPInterface method. It allows the client to modify the IP Interface, Hosting ComputerSystem, and/or EthernetPort.

Table 54 details the parameters for Extrinsic Method FileServerConfigurationService.CreateFileServer.

**Table 54 - Parameters for Extrinsic Method FileServerConfigurationService.CreateFileServer**

Parameter Name	Qualifier	Type	Description & Notes
ElementName	IN	string	An end user relevant name for the File Server being created. The value shall be stored in the 'ElementName' property for the created element. This parameter shall not be NULL or the empty string.
Job	OUT, REF	CIM_Concrete Job	Reference to the job (may be null if job completed).
TheElement	OUT, REF	CIM_Compute rSystem	The newly created FileServer.
FileServerSettings	IN, OUT, EI, NULL allowed	string	EmbeddedInstance ("CIM_FileServerSettings") The FileServerSettings for the newly created FileServer. If NULL or the empty string, a default FileServerSettings shall be used and returned on output.
IPInterfaceSettingData	IN,OUT, EI, NULL allowed	string	EmbeddedInstance ("CIM_IPInterfaceSettingData") The IPInterfaceSettingData that specifies the IP Interface that the FileServer will use for servicing all CIFS and NFS requests. If NULL or the empty string, a default IPInterfaceSettingData shall be used and returned on output.



**Table 54 - Parameters for Extrinsic Method FileServerConfigurationService.CreateFileServer**

Parameter Name	Qualifier	Type	Description & Notes
CIFSSettingData	IN,OUT, EI, NULL allowed	string	<p>EmbeddedInstance ("CIM_CIFSSettingData")</p> <p>The CIFSSettingData that specifies the CIFS settings for the FileServer being created.</p> <p>If this is NULL, the FileServer shall not have CIFS enabled and the resulting CIFSSettingData instance created shall have its "Enabled" property set to false. The CIFSSettingData instance will be returned on output.</p>
NFSSettingData	IN,OUT, EI, NULL allowed	string	<p>EmbeddedInstance ("CIM_NFSSettingData")</p> <p>The NFSSettingData that specifies the NFS settings for the FileServer being created.</p> <p>If this is NULL, the FileServer shall not have NFS enabled and the resulting NFSSettingData instance created shall have its "Enabled" property set to false. The NFSSettingData instance will be returned on output.</p>
DNSSettingData	IN, EI, NULL allowed,	string	<p>EmbeddedInstance ("CIM_DNSSettingData")</p> <p>The DNSSettingData that specifies the DNS settings for the FileServer being created.</p> <p>If this is NULL, the FileServer shall not have access to a DNS server and a DNSSettingData instance shall not be instantiated for the FileServer.</p>
NISSettingData	IN, EI, NULL allowed,	string	<p>EmbeddedInstance ("CIM_DNSSettingData")</p> <p>The NISSettingData that specifies the NIS settings for the FileServer being created.</p> <p>If this is NULL, the FileServer shall not have access to a NIS server and a NISSettingData instance shall not be instantiated for the FileServer.</p>
NASComputerSystem	IN, REF	CIM_ComputerSystem	Either the NAS Head or Self-contained NAS system that the FileServer shall be a component system of.
HostingComputerSystem	IN, REF	CIM_ComputerSystem	The HostingComputerSystem identifies the ComputerSystem that will host the FileServer.
EthernetPort	IN, REF	CIM_EthernetPort	The EthernetPort identifies the hardware port that the File Server will use for IP mount requests.
Normal Return			
Status		uint32	<p>"Job Completed with No Error",</p> <p>"Failed",</p> <p>"Method Parameters Checked - Job Started"</p>
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

#### 6.4.1.3 Signature and Parameters of FileServerConfigurationService.ModifyFileServer

This extrinsic modifies the settings for an existing FileServer. All settings except IPInterfaceSettingData, Hosting ComputerSystem, and EthernetPort may be modified. To modify the IPInterfaceSettingData, Hosting ComputerSystem, and/or EthernetPort properties, use the ModifyIPInterface extrinsic.

Table 55 details the parameters for Extrinsic Method FileServerConfigurationService.ModifyFileServer.

**Table 55 - Parameters for Extrinsic Method FileServerConfigurationService.ModifyFileServer**

Parameter Name	Qualifier	Type	Description & Notes
FileServer	IN,OUT,REF	CIM_ComputerSystem	The FileServer that is to be modified.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
ElementName	IN, NULL allowed	string	An end user relevant name for the File Server being modified.
FileServerSettings	IN, NULL allowed	string	EmbeddedInstance ("CIM_FileServerSettings") If non-NULL, this specifies the new FileServerSettings for the FileServer If NULL, then the FileServerSettings of the FileServer shall not be modified.
CIFSSettingData	IN, NULL allowed,	string	EmbeddedInstance ("CIM_CIFSSettingData") IF non-NULL, this specifies the new CIFS settings for the FileServer. If the "Enabled" property set to false, CIFS will be disabled for the FileServer. If NULL, then the CIFS setting of the FileServer shall not be modified.
NFSSettingData	IN, NULL allowed,	string	EmbeddedInstance ("CIM_NFSSettingData") IF non-NULL, this specifies the new NFS settings for the FileServer. If the "Enabled" property set to false, NFS will be disabled for the FileServer. If NULL, then the NFS setting of the FileServer shall not be modified.
DNSSettingData	IN, NULL allowed,	string	EmbeddedInstance ("CIM_DNSSettingData") IF non-NULL, this specifies the new DNS settings for the FileServer. If NULL, then the DNS setting of the FileServer shall not be modified.
NISSettingData	IN, NULL allowed,	string	EmbeddedInstance ("CIM_DNSSettingData") IF non-NULL, this specifies the new NIS settings for the FileServer. If NULL, then the NIS setting of the FileServer shall not be modified.
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"

**Table 55 - Parameters for Extrinsic Method FileServerConfigurationService.ModifyFileServer**

Parameter Name	Qualifier	Type	Description & Notes
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.
CannotModify	OUT, Indication	CIM_Error	The FileServer is in a state in which it cannot be modified.

**6.4.1.4 Signature and Parameters of FileServerConfigurationService.DeleteFileServer**

This extrinsic deletes an existing FileServer.

Table 56 describes the parameters for Extrinsic Method FileServerConfigurationService.DeleteFileServer.

**Table 56 - Parameters for Extrinsic Method FileServerConfigurationService.DeleteFileServer**

Parameter Name	Qualifier	Type	Description & Notes
FileServer	IN,REF	CIM_ComputerSystem	The FileServer that is to be deleted.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
Error Returns			
CannotDelete	OUT, Indication	CIM_Error	The FileServer is in a state in which it cannot be deleted.

**6.4.1.5 Signature and Parameters of FileServerConfigurationService.AddIPInterface**

This extrinsic adds a new IPInterface to an existing FileServer. The FileServer will respond to requests issued to this new IP address. The number of IP addresses that a FileServer can respond on is system dependent and the use of CreateGoalSettings to verify a new IP address is recommended.

Table 57 describes the parameters for Extrinsic Method FileServerConfigurationService.AddIPInterface.

**Table 57 - Parameters for Extrinsic Method FileServerConfigurationService.AddIPInterface**

Parameter Name	Qualifier	Type	Description & Notes
FileServer	IN,OUT,REF	CIM_ComputerSystem	The FileServer to which the IPInterface will be added.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
IPInterfaceSettingData	IN	string	EmbeddedInstance ("CIM_IPInterfaceSettingData") The IPInterfaceSettingData that specifies the settings of the IP Interface to be added to the FileServer.

**Table 57 - Parameters for Extrinsic Method FileServerConfigurationService.AddIPInterface**

Parameter Name	Qualifier	Type	Description & Notes
HostingComputerSystem	IN, REF	CIM_ComputerSystem	The ComputerSystem that will host the File Server for the new IP Interface
EthernetPort	IN, REF	CIM_EthernetPort	The EthernetPort identifies the hardware port that the File Server will use for mount requests on the new IPAddress.
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

**6.4.1.6 Signature and Parameters of FileServerConfigurationService.ModifyIPInterface**

This extrinsic modifies an existing IPInterface associated with a FileServer. The IPInterfaceSettingData, the Hosting ComputerSystem, and/or the EthernetPort may be modified.

Table 58 describes the parameters for Extrinsic Method FileServerConfigurationService.ModifyIPInterface.

**Table 58 - Parameters for Extrinsic Method FileServerConfigurationService.ModifyIPInterface**

Parameter Name	Qualifier	Type	Description & Notes
FileServer	IN,OUT,REF	CIM_ComputerSystem	The FileServer from which the IPInterface will be modified.
IPInterfaceSettingData	IN,REF	CIM_IPInterfaceSettingData	The IPInterfaceSettingData that is to be modified.  This is used to identify which IPInterfaceSettingData instance to modify.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
NewIPInterfaceSettingData	IN, NULL allowed	string	EmbeddedInstance ("CIM_IPInterfaceSettingData")  If non-NULL, the IPInterfaceSettingData that will replace an existing IPInterfaceSettingData instance in the FileServer.  If NULL, then the IPInterfaceSettingData will not be modified.
HostingComputerSystem	IN, REF, NULL allowed	CIM_ComputerSystem	If non-NULL, the new ComputerSystem that will host the IPInterface.  If NULL, the current ComputerSystem hosting the IPInterface will remain unchanged.
EthernetPort	IN, REF, NULL allowed	CIM_EthernetPort	If non-NULL, the EthernetPort identifies the new hardware port for the IPInterface.  If NULL, the current EthernetPort setting will not be changed.
Normal Return			

**Table 58 - Parameters for Extrinsic Method FileServerConfigurationService.ModifyIPInterface**

Parameter Name	Qualifier	Type	Description & Notes
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value

**6.4.1.7 Signature and Parameters of FileServerConfigurationService.DeleteIPInterface**

This extrinsic deletes an existing IPInterface associated with a FileServer.

Table 59 describes the parameters for Extrinsic Method FileServerConfigurationService.DeleteIPInterface.

**Table 59 - Parameters for Extrinsic Method FileServerConfigurationService.DeleteIPInterface**

Parameter Name	Qualifier	Type	Description & Notes
FileServer	IN,OUT,REF	CIM_ComputerSystem	The FileServer from which the IPInterface will be deleted.
IPInterfaceSettingData	IN,REF	CIM_IPInterfaceSettingData	The IPInterfaceSettingData that is to be deleted. This is used to identify which IPInterfaceSettingData instance to delete from the FileServer.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value

**6.5 Use Cases**

Not defined in this standard.

## 6.6 CIM Elements

Table 60 describes the CIM elements for File Server Manipulation.

**Table 60 - CIM Elements for File Server Manipulation**

Element Name	Requirement	Description
6.6.1 CIM_CIFSSettingData	Conditional	Conditional requirement: CIFS shares are supported by the provider. This class contains the CIFS settings for the File Server.
6.6.2 CIM_ConcreteComponent (FileServerSettings to CIFSSettingData)	Conditional	Conditional requirement: CIFS shares are supported by the provider. Represents the association between a FileServerSettings and CIFSSettingData.
6.6.3 CIM_ConcreteComponent (FileServerSettings to DNSSettingData)	Conditional	Conditional requirement: The DNSSettingData is supported by the provider. Represents the association between a FileServerSettings and DNSSettingData.
6.6.4 CIM_ConcreteComponent (FileServerSettings to IPInterfaceSettingData)	Conditional	Conditional requirement: There is an instance of IPInterfaceSettingData. Represents the association between a FileServerSettings and IPInterfaceSettingData.
6.6.5 CIM_ConcreteComponent (FileServerSettings to NFSSettingData)	Conditional	Conditional requirement: NFS Exports are supported by the provider. Represents the association between a FileServerSettings and NFSSettingData.
6.6.6 CIM_ConcreteComponent (FileServerSettings to NISSettingData)	Conditional	Conditional requirement: NIS (Network Information System) is supported by the provider. Represents the association between a FileServerSettings and NISSettingData.
6.6.7 CIM_DNSSettingData	Conditional	Conditional requirement: The DNSSettingData is supported by the provider. This element represents the DNS setting data to be used by a file server.
6.6.8 CIM_ElementCapabilities (FileServerConfigurationService to FileServerCapabilities)	Mandatory	This associates the File Server Configuration Service to the Capabilities element that represents the capabilities supported by all File Servers.
6.6.9 CIM_ElementCapabilities (FileServerConfigurationService to FileServerConfigurationCapabilities)	Mandatory	This associates the File Server Configuration Service to the ConfigurationCapabilities element that represents the capabilities that it supports.
6.6.10 CIM_ElementSettingData (ComputerSystem FileServer to FileServerSettings)	Optional	Associates a File Server with the FileServerSettings that were used to initially create the File Server.
6.6.11 CIM_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint)	Optional	The IPProtocolEndpoint associated with the IPInterfaceSettingData.
6.6.12 CIM_FileServerCapabilities	Mandatory	The capabilities of the File Server.
6.6.13 CIM_FileServerConfigurationCapabilities	Mandatory	This element represents the management Capabilities of the File Server Configuration Service. If the two arrays of extrinsic methods (SynchronousMethodsSupported and AsynchronousMethodsSupported) are empty, then the implementation is readonly.
6.6.14 CIM_FileServerConfigurationService	Mandatory	The File Server Configuration Service provides the methods to manipulate File Servers.
6.6.15 CIM_FileServerSettings	Conditional	Conditional requirement: The FileServer ComputerSystem has a FileServerSettings associated with it. This class contains the settings for the File Server.

**Table 60 - CIM Elements for File Server Manipulation**

Element Name	Requirement	Description
6.6.16 CIM_HostedDependency	Optional	Associates a Virtual File Server to the Computer System hosting it. This association will not exist for non-Virtual File Servers.
6.6.17 CIM_HostedService (Hosting Computer System to FileServerConfigurationService)	Mandatory	Associates the FileServerConfigurationService with the hosting computer system.
6.6.18 CIM_IPInterfaceSettingData	Optional	This class contains the settings for single IP interface.
6.6.19 CIM_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.)	Conditional	Conditional requirement: The NetworkVLAN is supported by the provider. Associates an IPProtocolEndpoint to NetworkVLAN.
6.6.20 CIM_NetworkVLAN	Conditional	Conditional requirement: The NetworkVLAN is supported by the provider. This element represents the virtual LAN (VLAN) tag settings for an IP interface. In the context of a file server, it represents the VLAN information.
6.6.21 CIM_NFSSettingData	Conditional	Conditional requirement: NFS Exports are supported by the provider. This class contains the NFS settings for the File Server.
6.6.22 CIM_NISSettingData	Conditional	Conditional requirement: NIS (Network Information System) is supported by the provider. This class contains the NIS settings for the File Server.
6.6.23 CIM_SettingsDefineCapabilities (CIFSettingData)	Conditional	Conditional requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported. Associates CIFSSettingData with FileServerCapabilities.
6.6.24 CIM_SettingsDefineCapabilities (DNSSettingData)	Conditional	Conditional requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported. Associates DNSSettingData with FileServerCapabilities.
6.6.25 CIM_SettingsDefineCapabilities (FileServerSettings)	Conditional	Conditional requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported. Associates FileServerSettings with FileServerCapabilities.
6.6.26 CIM_SettingsDefineCapabilities (IPInterfaceSettingData)	Conditional	Conditional requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported. Associates IPInterfaceSettingData with FileServerCapabilities.
6.6.27 CIM_SettingsDefineCapabilities (NFSSettingData)	Conditional	Conditional requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported. Associates NFSSettingData with FileServerCapabilities.
6.6.28 CIM_SettingsDefineCapabilities (NISSettingData)	Conditional	Conditional requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported. Associates NISSettingData with FileServerCapabilities.
6.6.29 CIM_SettingsDefineState (ComputerSystem FileServer to FileServerSettings)	Conditional	Conditional requirement: The FileServer ComputerSystem has a FileServerSettings associated with it. The FileServer's state represented by its FileServerSettings.

**Table 60 - CIM Elements for File Server Manipulation**

Element Name	Requirement	Description
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_Computer_System AND ANY SourceInstance.CIM_Computer_System::Dedicated[*] = 16	Optional	CQL -Creation of a File Server element.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_Computer_System AND ANY SourceInstance.CIM_Computer_System::Dedicated[*] = 16	Optional	CQL -Deletion of a File Server element.

**6.6.1 CIM\_CIFSSettingData**

Created By: Extrinsic: CreateFileServer

Modified By: Extrinsic: ModifyCIFS

Deleted By: Extrinsic: DeleteFileServer

Requirement: CIFS shares are supported by the provider.

Table 61 describes class CIM\_CIFSSettingData.

**Table 61 - SMI Referenced Properties/Methods for CIM\_CIFSSettingData**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the CIFSSettingData.
Enabled		Mandatory	This boolean indicates if CIFS is enabled on the File Server.
Charset		Optional	Specifies the character set to be used by the File Server when servicing CIFS Shares. The values are 0 1 2 ('Standard-ASCII' 'IBM-437','IBM-850'). If absent, then "Standard-ASCII" is assumed.
UseTCPOnly		Optional	This boolean if set to 'true' allows only TCP transport connections. If 'false', then both TCP and Netbios transport connections are allowed. The default value is 'false'.
NETBIOSName		Optional	The NetBIOS name of the FileServer.
WINSIP		Optional	An array of IP Addresses of Windows Internet Name Servers.
AuthenticationDomain		Mandatory	Name of CIFS domain to which the File Server is joined. Represents either the NTLM domain or the ActiveDirectory domain.
AuthenticationMode		Mandatory	Specifies if authentication is to be performed against either NTLM or ActiveDirectory domains. Valid values are 'NTLM' or 'ActiveDirectory'.
UseKerberos		Optional	Determines how ActiveDirectory authentication is performed. If 'true', limit ActiveDirectory authentication to use Kerberos. Otherwise do not limit to Kerberos only.
UseOpportunisticLocking		Optional	This boolean determines if opportunistic locking should be used by CIFS FileServer. If 'true', enable opportunistic locking.
SMBSigningOnly		Optional	This boolean determines if CIFS clients are allowed to connect if they use SMB signing for security. If 'true', then require clients to use SMB signing. Otherwise, do not require.
ClientsConnectAnonymously		Optional	This boolean dictates if the FileServer joins the CIFS Domain Controller anonymously or if a user and password are required. If 'true', then join anonymously. Otherwise, use DomainControllerUser and DomainControllerPassword to join.



**Table 61 - SMI Referenced Properties/Methods for CIM\_CIFSSettingData**

Properties	Flags	Requirement	Description & Notes
JoinDomainAnonymously		Optional	This boolean dictates if the FileServer joins the CIFS Domain Controller anonymously or if a user and password are required. If 'true', then join anonymously. Otherwise, use DomainControllerUser and DomainControllerPassword to join.
DomainControllerUser		Optional	User name to use when the Fileserver joins the CIFS Domain Controller.
DomainControllerPassword		Optional	Password to use when joining the CIFS Domain Controller.
CIFSDomainController		Optional	Name of the CIFS Domain Controller.
CASupported		Optional	This property applies to CIFS/SMB shares only. If it is true, it means that "Continuous Availability" is supported for CIFS shares. Continuous Availability (CA) - Client/Server mediated recovery from network and server failure with application transparency. Like Multi-Channel IO, this feature is somewhat analogous to capabilities available in NFSv4.
MultiChannelSupported		Optional	This property applies to CIFS/SMB protocol only. If it is true, it means that "Multi-Channel" feature is supported for CIFS/SMB. Multi-Channel (MPIO) - Provides the ability to access multiple Ethernet links as a logical pool supporting multiple SMB sessions and providing native bandwidth aggregation, link failover, MPIO intelligence. This feature enables the use of multiple physical network interfaces in an SMB 2.2 client and server. This enhancement in SMB 2.2 provides capabilities analogous to those currently available in NFSv4.
ProtocolVersions		Optional	An array of strings listing the versions of the CIFS file sharing protocol supported by the File Server.

### 6.6.2 CIM\_ConcreteComponent (FileServerSettings to CIFSSettingData)

Created By: External

Modified By: Static

Deleted By: External

Requirement: CIFS shares are supported by the provider.

Table 62 describes class CIM\_ConcreteComponent (FileServerSettings to CIFSSettingData).

**Table 62 - SMI Referenced Properties/Methods for CIM\_ConcreteComponent (FileServerSettings to CIFS-SettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerSettings.
PartComponent		Mandatory	The CIFSSettingData.

### 6.6.3 CIM\_ConcreteComponent (FileServerSettings to DNSSettingData)

Created By: External

Modified By: Static

Deleted By: External

Requirement: The DNSSettingData is supported by the provider.

Table 63 describes class CIM\_ConcreteComponent (FileServerSettings to DNSSettingData).

**Table 63 - SMI Referenced Properties/Methods for CIM\_ConcreteComponent (FileServerSettings to DNSSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerSettings.
PartComponent		Mandatory	The DNSSettingData.

#### 6.6.4 CIM\_ConcreteComponent (FileServerSettings to IPInterfaceSettingData)

Created By: External

Modified By: Static

Deleted By: External

Requirement: There is an instance of IPInterfaceSettingData.

Table 64 describes class CIM\_ConcreteComponent (FileServerSettings to IPInterfaceSettingData).

**Table 64 - SMI Referenced Properties/Methods for CIM\_ConcreteComponent (FileServerSettings to IPInterfaceSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerSettings.
PartComponent		Mandatory	The IPInterfaceSettingData.

#### 6.6.5 CIM\_ConcreteComponent (FileServerSettings to NFSSettingData)

Created By: External

Modified By: Static

Deleted By: External

Requirement: NFS Exports are supported by the provider.

Table 65 describes class CIM\_ConcreteComponent (FileServerSettings to NFSSettingData).

**Table 65 - SMI Referenced Properties/Methods for CIM\_ConcreteComponent (FileServerSettings to NFSSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerSettings.
PartComponent		Mandatory	The NFSSettingData.

#### 6.6.6 CIM\_ConcreteComponent (FileServerSettings to NISSettingData)

Created By: External

Modified By: Static

Deleted By: External

Requirement: NIS (Network Information System) is supported by the provider.

Table 66 describes class CIM\_ConcreteComponent (FileServerSettings to NISSettingData).

**Table 66 - SMI Referenced Properties/Methods for CIM\_ConcreteComponent (FileServerSettings to NISSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerSettings.
PartComponent		Mandatory	The NISSettingData.

#### 6.6.7 CIM\_DNSSettingData

Created By: Extrinsic

Modified By: Extrinsic

Deleted By: Extrinsic

Requirement: The DNSSettingData is supported by the provider.

Table 67 describes class CIM\_DNSSettingData.

**Table 67 - SMI Referenced Properties/Methods for CIM\_DNSSettingData**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the DNSSettingData.
DomainName		Mandatory	The DNS domain to use for looking up addresses.
DNSServerAddresses		Mandatory	The addresses of DNS servers to contact. The array specifies the order in which the DNS servers will be contacted.

#### 6.6.8 CIM\_ElementCapabilities (FileServerConfigurationService to FileServerCapabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 68 describes class CIM\_ElementCapabilities (FileServerConfigurationService to FileServerCapabilities).

**Table 68 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (FileServerConfigurationService to FileServerCapabilities)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The File Server Configuration Service.
Capabilities		Mandatory	The File Server Capabilities element.

#### 6.6.9 CIM\_ElementCapabilities (FileServerConfigurationService to FileServerConfigurationCapabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 69 describes class CIM\_ElementCapabilities (FileServerConfigurationService to FileServerConfigurationCapabilities).

**Table 69 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (FileServerConfigurationService to FileServerConfigurationCapabilities)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The File Server Configuration Service.
Capabilities		Mandatory	The File Server Configuration Capabilities element.

#### 6.6.10 CIM\_ElementSettingData (ComputerSystem FileServer to FileServerSettings)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Optional

Table 70 describes class CIM\_ElementSettingData (ComputerSystem FileServer to FileServerSettings).

**Table 70 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (ComputerSystem FileServer to FileServerSettings)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The File Server ComputerSystem.
SettingData		Mandatory	The FileServerSettings.

#### 6.6.11 CIM\_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Optional

Table 71 describes class CIM\_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint).

**Table 71 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The IPProtocolEndpoint.
SettingData		Mandatory	The IPInterfaceSettingData.

#### 6.6.12 CIM\_FileServerCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 72 describes class CIM\_FileServerCapabilities.

**Table 72 - SMI Referenced Properties/Methods for CIM\_FileServerCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the FileServerCapabilities element of a File Server Configuration Service.
ElementName		Mandatory	A user-friendly name for this Capabilities element.
FileServerSettingsSupported		Mandatory	Indicates if FileServerSettings is supported for the FileServer. FileServerSettings will be supported if the value is "true".
CIFSSupported		Mandatory	Indicates if CIFS Shares are supported by the FileServer. CIFS Shares will be supported if the value is "true".
NFSSupported		Mandatory	Indicates if NFS Exports are supported by the FileServer. NFS Exports will be supported if the value is "true".
NISSupported		Mandatory	Indicates if NIS (Network Information System) is supported by the FileServer. NIS will be supported if the value is "true".
DNSSupported		Mandatory	Indicates if DNS is supported by the FileServer. DNS will be supported if the value is "true".
NetworkVLANSupported		Mandatory	Indicates if network VLAN Tagging is supported by the FileServer. VLAN tagging will be supported if the value is "true".
ScaleOutSupported		Mandatory	Indicates if ScaleOut is supported by the FileServer. ScaleOut will be supported if the value is "true".
CreateGoalSettings()		Mandatory	This extrinsic method supports the creation of a set of Settings that are a supported variant of the Settings passed as embedded instances via IN parameters. The method returns the supported Settings in OUT parameters, each containing an array of embedded instances. Many of the IN parameters are optional, and if left NULL result in NULL being returned in the corresponding OUT parameters.

### 6.6.13 CIM\_FileServerConfigurationCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 73 describes class CIM\_FileServerConfigurationCapabilities.

**Table 73 - SMI Referenced Properties/Methods for CIM\_FileServerConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for this element representing the capabilities of a File Server Configuration Service.
ElementName		Mandatory	A user-friendly name for this Capabilities element.

**Table 73 - SMI Referenced Properties/Methods for CIM\_FileServerConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
SynchronousMethodsSupported	N	Mandatory	The Service supports a number of extrinsic methods -- this property identifies the ones that can be called synchronously. Note: A supported method shall be listed in this property or in the AsynchronousMethodsSupported property or both.
AsynchronousMethodsSupported	N	Mandatory	The Service supports a number of extrinsic methods -- this property identifies the ones that can be called asynchronously. Note: A supported method shall be listed in this property or in the SynchronousMethodsSupported property or both.
CanConfigureCIFS		Mandatory	Indicates if the CIFS Settings can be configured. The settings can be configured if the value is "true".
CanConfigureNFS		Mandatory	Indicates if the NFS Settings can be configured. The settings can be configured if the value is "true".
CanConfigureNIS		Mandatory	Indicates if the NIS (Network Information Service) Settings can be configured. The settings can be configured if the value is "true".
CanConfigureDNS		Mandatory	Indicates if the DNS Settings can be configured. The settings can be configured if the value is "true".
CanConfigureNetworkVLSN		Mandatory	Indicates if the network VLAN Tagging Settings can be configured. The settings can be configured if the value is "true".

**6.6.14 CIM\_FileServerConfigurationService**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 74 describes class CIM\_FileServerConfigurationService.

**Table 74 - SMI Referenced Properties/Methods for CIM\_FileServerConfigurationService**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A user-friendly name for this Service.
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
CreationClassName		Mandatory	Key.
Name		Mandatory	Key.
CreateFileServer()		Mandatory	Create a new instance of File Server.
ModifyFileServer()		Mandatory	Modify an existing File Server. This is used to modify FileServerSettings, CIFSSettingData, NFSSettingData, DNSSettingData, or NISSettingData.
DeleteFileServer()		Mandatory	Delete an existing File Server.
AddIPInterface()		Optional	Add a new IPInterface to an existing File Server.

**Table 74 - SMI Referenced Properties/Methods for CIM\_FileServerConfigurationService**

Properties	Flags	Requirement	Description & Notes
ModifyIPInterface()		Optional	Modify an IPInterface associated with an existing File Server.
DeletelPInterface()		Optional	Delete an IPInterface associated with an existing File Server.

**6.6.15 CIM\_FileServerSettings**

Created By: Extrinsic: CreateFileServer

Modified By: Extrinsic: ModifyFileServer

Deleted By: Extrinsic: DeleteFileServer

Requirement: The FileServer ComputerSystem has a FileServerSettings associated with it.

Table 75 describes class CIM\_FileServerSettings.

**Table 75 - SMI Referenced Properties/Methods for CIM\_FileServerSettings**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the FileServerSettings.
HostLookupOrder		Optional	Specifies the services and order to use them for host lookup. An array of elements with these values: 'DNS','NIS', 'None', or 'UploadedFile'. 'UploadedFile' refers to the uploaded file of host names.
UserLoginLookupOrder		Optional	Specifies the services and order to use them for user lookup. An array of elements with these values: 'DNS','NIS', 'None', or 'UploadedFile'. 'file' 'UploadedFile' refers to the uploaded file of user passwords.
NFSCIFSAccountMapping		Optional	Controls the mapping of accounts between NFS and CIFS. Valid values are 'None', 'All', or 'Domain'. If 'None', then no account mapping is performed. If 'All', then mapping is done for all CIFS domains. If 'Domain', then mapping is done for the users in the CIFS domain specified in AccountMappingDomain.
AccountMappingDomain		Optional	If NFSCIFSAccountMapping = 'Domain', then this property will contain the name of the domain to use for NFS to CIFS account mapping.

**6.6.16 CIM\_HostedDependency**

Created By: Extrinsic: CreateFileServer

Modified By: Static

Deleted By: Extrinsic: DeleteFileServer

Requirement: Optional

Table 76 describes class CIM\_HostedDependency.

**Table 76 - SMI Referenced Properties/Methods for CIM\_HostedDependency**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The Virtual File Server ComputerSystem. A Virtual File Server ComputerSystem is a File Server and shall have Dedicated=16 (File Server).
Antecedent		Mandatory	The hosting ComputerSystem. The hosting ComputerSystem may be the top level NAS ComputerSystem or an Multiple Computer System (non-top level) system.

#### 6.6.17 CIM\_HostedService (Hosting Computer System to FileServerConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 77 describes class CIM\_HostedService (Hosting Computer System to FileServerConfigurationService).

**Table 77 - SMI Referenced Properties/Methods for CIM\_HostedService (Hosting Computer System to File-ServerConfigurationService)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The File Server Configuration Service.
Antecedent		Mandatory	The hosting ComputerSystem.

#### 6.6.18 CIM\_IPInterfaceSettingData

Created By: Extrinsic: CreateFileServer | AddIPInterface

Modified By: Extrinsic: ModifyIPInterface

Deleted By: Extrinsic: DeleteFileServer | DeleteIPInterface

Requirement: Optional

Table 78 describes class CIM\_IPInterfaceSettingData.

**Table 78 - SMI Referenced Properties/Methods for CIM\_IPInterfaceSettingData**

Properties	Flags	Requirement	Description & Notes
IPAddress		Mandatory	The IPAddress that will be used by the File Server. This can be either an IPv4 or IPv6 address.
AddressType		Mandatory	The IPAddress format. This can be either "IPv4" or "IPv6".
SubnetMask		Mandatory	The subnet mask that will be used by the File Server.
IPv6PrefixLength		Conditional	Conditional requirement: Required if the array property CIM_IPInterfaceSettingData.AddressType contains the string \IPv6\.'If AddressType specifies IPv6, then this specifies the prefix length for the IPv6 address in IPAddress.



**Table 78 - SMI Referenced Properties/Methods for CIM\_IPInterfaceSettingData**

Properties	Flags	Requirement	Description & Notes
VLANId		Optional	If present contains the ID of the VLAN that this IP setting will be associated with.
MTU		Optional	If present contains the maximum transmission unit to be used for this IP setting. If not present, then the default of 1500 will be used.
RSSCapable		Optional	This property is used to indicate whether this IPInterface is Receive-side Scaling (RSS) capable or not. Receive-side Scaling (RSS)- Receive-Side Scaling resolves the single-processor bottleneck by allowing the receive side network load from a network adapter to be shared across multiple processors. RSS enables packet receive-processing to scale with the number of available processors.
RDMACapable		Optional	This property is used to indicate whether this IPInterface is Remote Direct Memory Access (RDMA) capable or not. Remote Direct Memory Access Protocol (RDMA) - Accelerated I/O delivery model which works by allowing application software to bypass most layers of software and communicate directly with the hardware.
LinkSpeed		Optional	Speed of this IPInterface in bits per second.

**6.6.19 CIM\_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: The NetworkVLAN is supported by the provider.

Table 79 describes class CIM\_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.).

**Table 79 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.)**

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	The IPProtocolEndpoint.
Collection		Mandatory	The NetworkVLAN.

**6.6.20 CIM\_NetworkVLAN**

Created By: Extrinsic

Modified By: Extrinsic

Deleted By: Extrinsic

Requirement: The NetworkVLAN is supported by the provider.

Table 80 describes class CIM\_NetworkVLAN.

**Table 80 - SMI Referenced Properties/Methods for CIM\_NetworkVLAN**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the NetworkVLAN.
VLANid		Mandatory	The VLAN id that is to be associated with an IP interface. The id shall be included in all IP packets being sent through an IP interface.
TransmissionSize		Mandatory	The maximum transmission unit size that is associated with an IP Interface.

#### 6.6.21 CIM\_NFSSettingData

Created By: Extrinsic: CreateFileServer

Modified By: Extrinsic: ModifyNFS

Deleted By: Extrinsic: DeleteFileServer

Requirement: NFS Exports are supported by the provider.

Table 81 describes class CIM\_NFSSettingData.

**Table 81 - SMI Referenced Properties/Methods for CIM\_NFSSettingData**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the NFSSettingData.
Enabled		Mandatory	This boolean indicates if NFS is enabled on the File Server.
Charset		Optional	Specifies the character set to be used by the File Server when servicing CIFS Shares. The values are 0 1 2 ('Standard-ASCII' 'UTF8' 'ISO-8859-1'). If absent, then 'ISO-8859-1' is assumed.
MaximumTCPConnections		Optional	This specifies the number of concurrent TCP connections that are allowed for the NFS protocol. If set to 0, then TCP will be disabled for NFS.
Port		Optional	The port the File Server listens for mount requests. If absent, default to 2049.
NonNFSuid		Optional	User ID to use for requests from non-NFS access. If absent, default to -1.
NonNFSgid		Optional	Group ID to use for requests from non-NFS access. If absent, default to -1.
UseReservedPorts		Optional	This boolean specifies that the File Server will only allow NFS mount requests from client machine TCP/IP ports less than 1024. If 'true', only allow mount requests from ports less than 1024. Otherwise, allow mount requests from any client port.
OnlyRootChown		Optional	This boolean specifies if the root user is allowed to issue chown (change ownership) requests. If 'true', then only let root user issue chown request. Otherwise, allow any user to issue chown requests.

#### 6.6.22 CIM\_NISSettingData

Created By: Extrinsic: CreateFileServer

Modified By: Extrinsic: ModifyFileServer

Deleted By: Extrinsic: DeleteFileServer

Requirement: NIS (Network Information System) is supported by the provider.

Table 82 describes class CIM\_NISSettingData.

**Table 82 - SMI Referenced Properties/Methods for CIM\_NISSettingData**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the NISSettingData.
DomainName		Mandatory	NIS Domain Name.
ServerIP		Mandatory	An array of IP Addresses IP Addresses of NIS Servers.

#### 6.6.23 CIM\_SettingsDefineCapabilities (CIFSettingData)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported.

Table 83 describes class CIM\_SettingsDefineCapabilities (CIFSettingData).

**Table 83 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (CIFSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerCapabilities reference.
PartComponent		Mandatory	The CIFSSettingData reference.

#### 6.6.24 CIM\_SettingsDefineCapabilities (DNSSettingData)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported.

Table 84 describes class CIM\_SettingsDefineCapabilities (DNSSettingData).

**Table 84 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (DNSSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerCapabilities reference.
PartComponent		Mandatory	The CIFSSettingData reference.

#### 6.6.25 CIM\_SettingsDefineCapabilities (FileServerSettings)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported.

Table 85 describes class CIM\_SettingsDefineCapabilities (FileServerSettings).

**Table 85 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (FileServerSettings)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerCapabilities reference.
PartComponent		Mandatory	The FileServerSetting reference.

#### 6.6.26 CIM\_SettingsDefineCapabilities (IPInterfaceSettingData)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported.

Table 86 describes class CIM\_SettingsDefineCapabilities (IPInterfaceSettingData).

**Table 86 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (IPInterfaceSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerCapabilities reference.
PartComponent		Mandatory	The IPInterfaceSettingData reference.

#### 6.6.27 CIM\_SettingsDefineCapabilities (NFSSettingData)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported.

Table 87 describes class CIM\_SettingsDefineCapabilities (NFSSettingData).

**Table 87 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (NFSSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerCapabilities reference.
PartComponent		Mandatory	The NFSSettingData reference.

#### 6.6.28 CIM\_SettingsDefineCapabilities (NISSettingData)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Synchronous creation of a FileServer is supported or Asynchronous creation of a FileServer is supported.

Table 88 describes class CIM\_SettingsDefineCapabilities (NISSettingData).

**Table 88 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (NISSettingData)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The FileServerCapabilities reference.
PartComponent		Mandatory	The NISSettingData reference.

#### 6.6.29 CIM\_SettingsDefineState (ComputerSystem FileServer to FileServerSettings)

Created By: External

Modified By: Static

Deleted By: External

Requirement: The FileServer ComputerSystem has a FileServerSettings associated with it.

Table 89 describes class CIM\_SettingsDefineState (ComputerSystem FileServer to FileServerSettings).

**Table 89 - SMI Referenced Properties/Methods for CIM\_SettingsDefineState (ComputerSystem FileServer to FileServerSettings)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The File Server ComputerSystem.
SettingData		Mandatory	The FileServerSettings.

**STABLE**

---



---



---



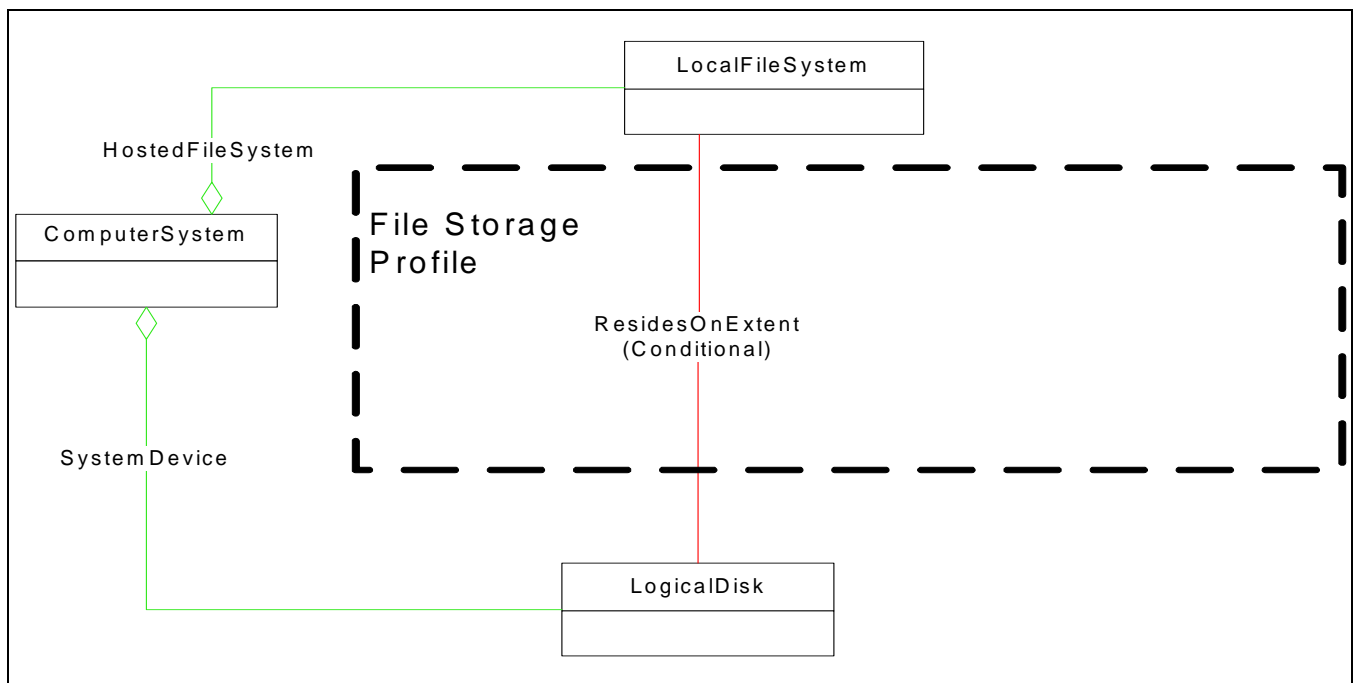
---

**STABLE**
**7 File Storage Profile****7.1 Description****7.1.1 Synopsis****Profile Name:** File Storage (Component Profile)**Version:** 1.4.0**Organization:** SNIA**Central Class:** N/A**Scoping Class:** ComputerSystem**Related Profiles:** Not defined in this standard.**7.1.2 Overview**

The File Storage Profile is a component profile for autonomous profiles that support filesystems. Specifically, in this release of SMI-S, this includes the NAS Head and Self-Contained NAS Profiles.

**7.1.3 Implementation**

Figure 11: "File Storage Instance" illustrates the mandatory and optional classes for the modeling of file storage for the profiles that support filesystems. This profile is supported by the Self-contained NAS and the NAS Head Profiles.

**Figure 11 - File Storage Instance**

The File Storage Profile models the mapping of filesystems to LogicalDisks. For the NAS Head and Self-contained NAS Profiles each filesystem shall be established on one LogicalDisk. The relationship

between the LocalFileSystem and the LogicalDisk is represented by the ResidesOnExtent association. This association is listed as conditional on the parent profile being either the NAS Head or the Self-contained NAS Profile. The LogicalDisk may be a LogicalDisk as defined in the Block Services Package or part of the parent profile.

The FileStorage Profile is a “read-only” profile. That is, the methods for creating, modifying or deleting a LocalFileSystem are external to the File Storage Profile. The SMI-S prescribed way of performing these functions are covered by the Filesystem Manipulation Profile.

## **7.2 Health and Fault Management Consideration**

Not defined in this standard.

---

---

## **DEPRECATED**

## **7.3 Cascading Considerations**

In some cases, the parent profile does not implement Block Services Package. In this case, the parent profile would implement a LogicalDisk that is “imported” from another profile (e.g., a Volume Management Profile). This section discusses those cascading considerations.

### **7.3.1 Cascaded Resources**

A File Storage profile may get its storage from the operating system (HDR Profile), a volume manager, an Array or Storage Virtualizer. As such, there is a cascading relationship between the File Storage Profile and the profiles (e.g., Volume Management Profiles) that provide the storage for the File Storage Profile. Figure 12 illustrates the constructs to be used to model this cascading relationship.



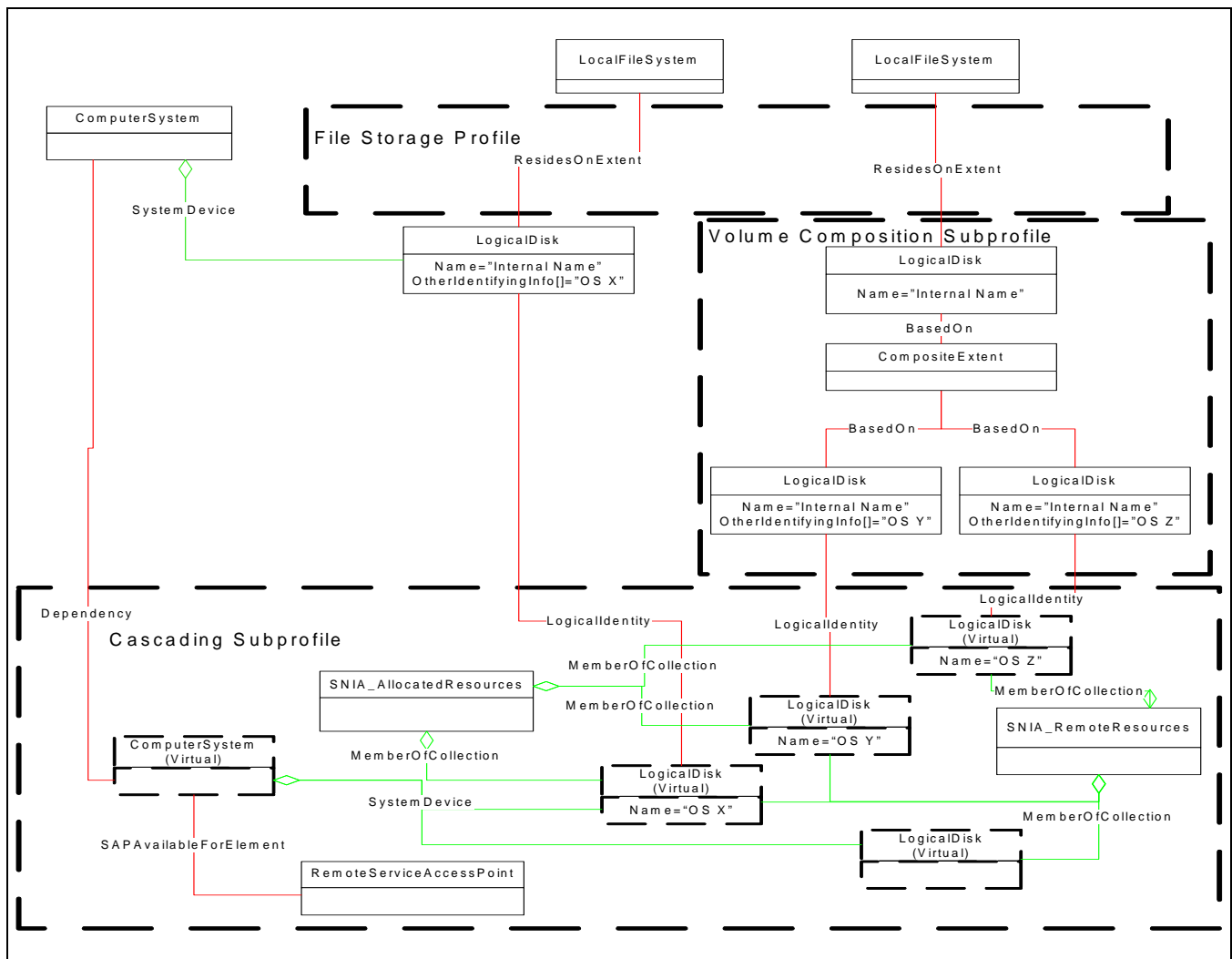


Figure 12 - Cascading File Storage

Figure 12: "Cascading File Storage" shows two filesystems (**LocalFileSystem**). Both reside on one **LogicalDisk**. But the **LogicalDisk** on the right is a composite of lower level **LogicalDisks**. The storage that is imported from the remote profile are **LogicalDisks** at the lowest level of the Filesystem Profile. So, in the first (left side) case, the **LogicalIdentity** is between the **LogicalDisk** on which the filesystem resides to the imported **LogicalDisk** (or **StorageVolume**). In the second case (the right side) the **LogicalIdentity** is between the "lowest level" **LogicalDisks** in **Volume Composition** and the imported **LogicalDisks** (or **StorageVolumes**).

NOTE **LogicalIdentity** is an abstract class and would be subclassed by an implementation.

The **ComputerSystem** in the Filesystem Profile would be the computer system that hosts the filesystem. The "Virtual" **ComputerSystem** is the top level **ComputerSystem** of the HDR, Volume Manager, Array or Storage Virtualizer. There shall be a **Dependency** association between these computer systems. **LogicalDisks** (or **StorageVolumes**) that are in use by the Filesystem Profile would have a **MemberOfCollection** association to the **SNIA\_AllocatedResources** collection. All the **LogicalDisks** (or **StorageVolumes**) that the Filesystem Profile can see (including the ones that are allocated) would have a **MemberOfCollection** association to the **SNIA\_RemoteResources** instance.

The RemoteServiceAccessPoint associated to the virtual computer system via SAPAvailableForFileShare would be information on the management interface for the HDR, Volume Manager, Array or Storage Virtualizer.

Table 90 provides the specific cascading information for cascading file storage.

**Table 90 - Cascaded Storage**

File Storage Resource	Leaf Profile	Leaf Resource	Association	Notes
LogicalDisk	Volume Management or HDR	LogicalDisk	LogicalIdentity	
LogicalDisk	Array or Storage Virtualizer	StorageVolume	LogicalIdentity	

### 7.3.2 Ownership Privileges

In support of the cascading File Storage, an implementation may assert ownership over the LogicalDisks (or StorageVolumes) that they import. If the Volume Management implementation supports Ownership, the File Storage implementation may assert ownership using the following Privileges:

- Activity - Execute
- ActivityQualifier - CreateOrModifyFromStoragePool and ReturnToStoragePool
- FormatQualifier - Method

NOTE HDR does not support Block Storage Resource Ownership, so this cannot be supported if the underlying profile is HDR.

## DEPRECATED

---

## 7.4 Methods of the Profile

### 7.4.1 Extrinsic Methods of the Profile

Not defined in this standard

NOTE The methods for defining the various mappings would be handled by the Filesystem Manipulation Profile.

### 7.4.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

## 7.5 Client Considerations and Recipes

Not defined in this standard.

## 7.6 CIM Elements

Table 91 describes the CIM elements for File Storage.

**Table 91 - CIM Elements for File Storage**

Element Name	Requirement	Description
7.6.1 CIM_ResidesOnExtent	Conditional	Conditional requirement: NAS Profiles require that LocalFileSystems reside on one LogicalDisk. or NAS Profiles require that LocalFileSystems reside on one LogicalDisk. Represents the association between a local FileSystem and the underlying LogicalDisk that it is built on.

### 7.6.1 CIM\_ResidesOnExtent

Created By: External

Modified By: Static

Deleted By: External

Requirement: NAS Profiles require that LocalFileSystems reside on one LogicalDisk. or NAS Profiles require that LocalFileSystems reside on one LogicalDisk.

Table 92 describes class CIM\_ResidesOnExtent.

**Table 92 - SMI Referenced Properties/Methods for CIM\_ResidesOnExtent**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The LocalFileSystem that is built on top of a LogicalDisk.
Antecedent		Mandatory	The LogicalDisk that underlies a LocalFileSystem.

---



---

**STABLE**

---



---



---



---

## STABLE

### 8 Filesystem Profile

#### 8.1 Description

##### 8.1.1 Synopsis

**Profile Name:** Filesystem (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** LocalFileSystem

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 93 describes the related profiles for Filesystem.

**Table 93 - Related Profiles for Filesystem**

Profile Name	Organization	Version	Requirement	Description
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

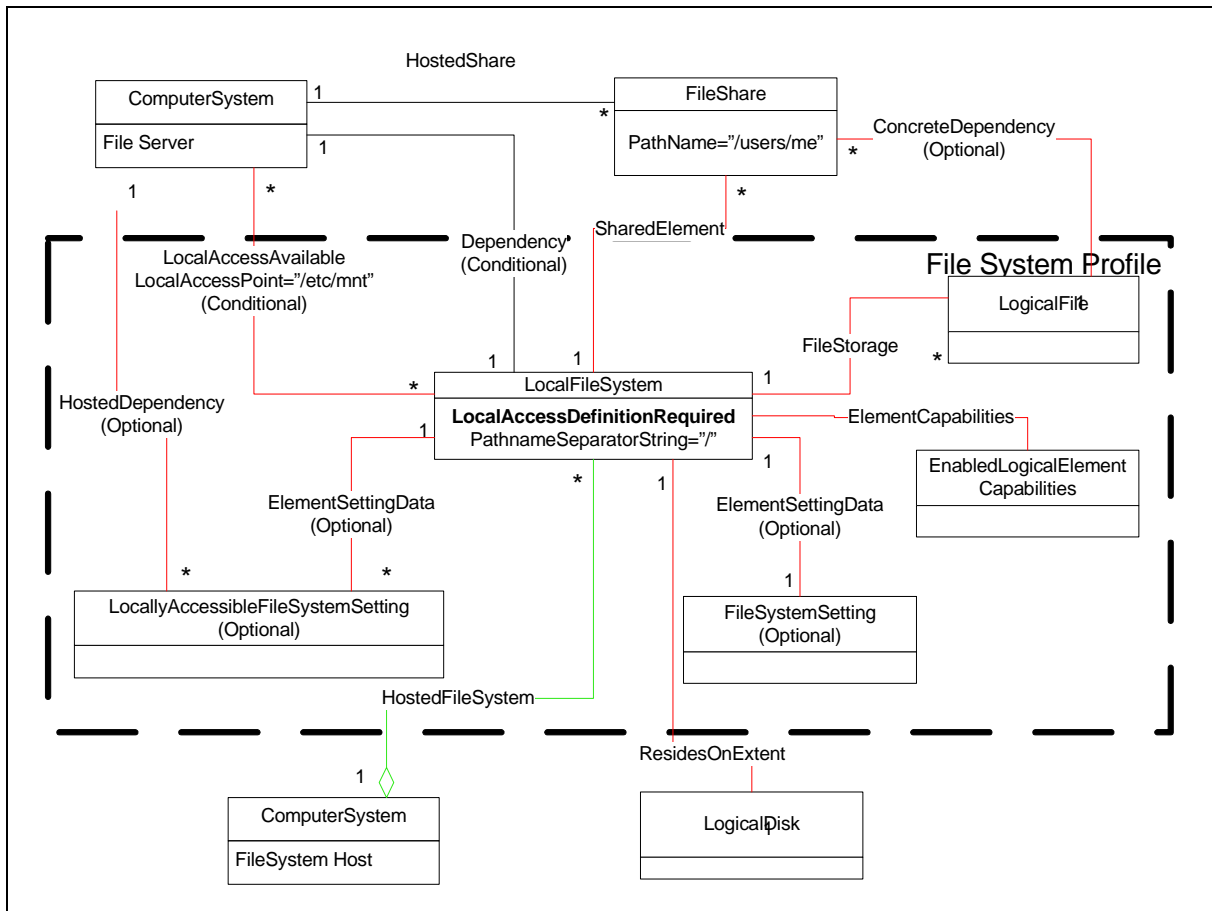
The Filesystem Profile is a component profile for autonomous profiles that support filesystems. Specifically, in this release of SMI-S, this includes the NAS Head and the Self-Contained NAS Profiles. A number of other profiles and component profiles make use of elements of the Filesystem Profile and will be referred to in this specification as "filesystem-related profiles" -- these include but are not limited to the Filesystem Manipulation Profile, File Export Profile, File Export Manipulation Profile, NAS Head Profile, and so on.

The state transitions involved when an appropriate storage element is transformed into a filesystem are described in Annex B (Informative) State Transitions from Storage to File Shares.

##### 8.1.2 Instance Diagrams

Figure 13: "Filesystem Instance" illustrates the mandatory, optional, and conditional classes for the modeling of filesystems for the profiles that support filesystems. This profile is supported by the Self-contained NAS and the NAS Head Profiles. The dashed box contains the elements that this profile supports -- the elements outside the dashed box depend on other profiles for their maintenance (creation, deletion, and modification). There are two ComputerSystems shown outside the box that represent different dedicated roles that could be performed by different actual computers (or could be performed by a single computer).

## Filesystem Profile



**Figure 13 - Filesystem Instance**

A filesystem shall be represented in the model as an instance of **LocalFileSystem**. A **LocalFileSystem** instance shall have a **ResidesOnExtent** association to a **LogicalDisk** (shown, but outside this profile). A client would determine the size (in bytes) of a filesystem by inspecting the size of the **LogicalDisk** on which the **LocalFileSystem** resides.

**NOTE** The filesystem-related profiles build **LocalFileSystems** on a **LogicalDisk(s)**. In the cases supported in this release of SMI-S, one **LocalFileSystem** may be established on one **LogicalDisk**. In a future release, more elaborate mappings may exist between **FileSystems** and one or more **LogicalDisks**.

The **LocalFileSystem** shall have a **HostedFileSystem** association to a **ComputerSystem**. Normally this will be the top level **ComputerSystem** of the parent profile (typically one of the filesystem-related profiles such as the NAS Head or the Self-Contained NAS Profile). However, if the Multiple Computer System Profile is implemented, the **HostedFileSystem** may be associated to a component **ComputerSystem**. See 25 Multiple Computer System Profile in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*.

The **LocalFileSystem** element may also have an **ElementSettingData** association to the **FileSystemSetting** for that filesystem. However, the **FileSystemSetting** and **ElementSettingData** are optional in this profile.

The **LocalFileSystem** may also have an **ElementCapabilities** association to an **EnabledLogicalUnitCapabilities** to identify naming and requested state change capabilities.

There may be zero or more FileShare elements associated to the LocalFileSystem element via the SharedElement association. An implementation would be required to populate only those FileShare elements representing files (or directories) that are exported using a supported file sharing protocol (such as CIFS or NFS). The path to the file or directory from the root of the LocalFileSystem is specified by the FileShare.Name property.

NOTE In order to support backward compatibility with the NAS Head and Self-contained NAS Profiles in previous SMI-S releases, the class LogicalFile (shown outside the dashed box in the figure) and two associations (ConcreteDependency outside the dashed box and FileStorage shown inside the dashed box) must be supported. These duplicate the functionality provided by specifying FileShare.Name, at the cost of requiring that certain LogicalFiles, but not all, be surfaced.

---

## EXPERIMENTAL

### 8.1.2.1 Local Access Requirement

In addition, if the property LocalFileSystem.LocalAccessDefinitionRequired is set to true, the filesystem must be made exportable via a file server. In that case, there shall be a LocalAccessAvailableToFS association from the LocalFileSystem element to the file server ComputerSystem and, optionally, a LocallyAccessibleFilesystemSettings element associated via an ElementSettingData association to the LocalFileSystem. The LocallyAccessibleFilesystemSettings element is an instance of ScopedSettingData and is associated to the file server ComputerSystem via a ScopedSetting association. The ScopedSetting association indicates that this setting is constrained by the associated file server. The LocalAccessAvailableToFS association is required but conditional on LocalAccessDefinitionRequired being true, while the LocallyAccessibleFilesystemSettings element and the ScopedSetting association are not required (i.e., optional).

NOTE They are still conditional on LocalAccessDefinitionRequired being true, but in this version of SMI-S, that is not represented in the XML file.

Since LocalAccessAvailableToFS is an association, there can only be ONE instance per LocalFileSystem for each FileServer. This is a common restriction. For each LocalAccessAvailableToFS association, there should only be zero (if optionally not implemented) or one (if optionally implemented) instances of LocallyAccessibleFilesystemSettings.

### 8.1.2.2 Directory Service Use

A filesystem needs to be supported by a directory service that resolves user and group identifiers (referred to as UID, GID, or SID) to specific operational users and groups. The enumerated property LocalFileSystem.DirectoryServiceUsage indicates the kind of support a filesystem requires from a directory service -- the options include "Not Used", "Optional", and "Required". If "Required", the filesystem will be associated to a computer system that provides infrastructure support for such identity resolution.

The directory service may be hosted by any ComputerSystem, but it must be accessible in real-time to the ComputerSystem that makes the filesystem available to operational users -- this is either a file server ComputerSystem (one may already be required if LocalFileSystem.LocalAccessDefinitionRequired is true, but it is optional otherwise) or the ComputerSystem hosting the filesystem. The directory service may be "natively" hosted on that ComputerSystem (file server or filesystem host) or may be identified by that ComputerSystem in some way.

The support relationship between a LocalFileSystem element and the ComputerSystem that identifies and uses the directory service shall be represented by a Dependency association with the ComputerSystem element as the Antecedent and the LocalFileSystem element as the Dependent.

In the instance diagram above, the Dependency association has been shown between the LocalFileSystem and a file server ComputerSystem (with Dedicated[]="16"). A LocalFileSystem element shall only identify one ComputerSystem for directory service access. In addition, the consistency of filesystem security implementation requires that all the file server ComputerSystems that make a

filesystem locally available must use the same directory service or use mutually consistent directory services.

## EXPERIMENTAL

---

### 8.1.2.3 Element Naming

The name of a FileSystem may be changed. The existence of the EnabledLogicalElementCapabilities instance associated to the FileSystem indicates that the FileSystem can be named. If ElementNameEditSupported is set to TRUE, then the ElementName of the associated FileSystem may be modified. The ElementNameMask property provides the regular expression that expresses the limits of the name; see 8.7.5 for the class definition for EnabledLogicalElementCapabilities for details for this property.

## 8.2 Health and Fault Management Consideration

The Filesystem Profile supports state information (e.g., OperationalStatus) on the following elements of the model:

- Local filesystems (See Table 104 - SMI Referenced Properties/Methods for CIM\_HostedFileSystem (LocalFileSystem))

### 8.2.1 OperationalStatus for Filesystems

Table 94 describes each filesystem OperationalStatus.

**Table 94 - Filesystem OperationalStatus**

Primary OperationalStatus	Description
2 "OK"	The filesystem has good status
3 "Degraded"	The filesystem is operating in a degraded mode. This could be due to the health state of the underlying storage being degraded or in error.
4 "Stressed"	The filesystem resources are stressed
5 "Predictive Failure"	The filesystem might fail because some resource or component is predicted to fail
6 "Error"	An error has occurred causing the filesystem to become unavailable. Operator intervention through SMI-S (managing the LocalFileSystem) to restore the filesystem may be possible.
6 "Error"	An error has occurred causing the filesystem to become unavailable. Automated recovery may be in progress.
7 "Non-recoverable Error"	The filesystem is not functioning. Operator intervention through SMI-S will not fix the problem.
8 "Starting"	The filesystem is in process of initialization and is not yet available operationally.
9 "Stopping"	The filesystem is in process of stopping, and is not available operationally.
10 "Stopped"	The filesystem cannot be accessed operationally because it is stopped -- if this did not happened because of operator intervention or happened in real-time, the OperationalStatus would have been "Lost Communication" rather than "Stopped".
11 "In Service"	The filesystem is offline in maintenance mode, and is not available operationally.



**Table 94 - Filesystem OperationalStatus**

Primary OperationalStatus	Description
13 "Lost Communications"	The filesystem cannot be accessed operationally -- if this happened because of operator intervention it would have been "Stopped" rather than "Lost Communication".
14 "Aborted"	The filesystem is stopped but in a manner that may have left it in an inconsistent state.
15 "Dormant"	The filesystem is offline; and the reason for not being accessible is unknown.
16 "Supporting Entity in Error"	The filesystem is in an error state, or may be OK but not accessible, because a supporting entity is not accessible.

### 8.3 Methods of the Profile

#### 8.3.1 Extrinsic Methods of the Profile

Not defined in this standard.

#### 8.3.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

### 8.4 Use Cases

Not defined in this standard.

### 8.5 CIM Elements

Table 95 describes the CIM elements for Filesystem.

**Table 95 - CIM Elements for Filesystem**

Element Name	Requirement	Description
8.5.1 CIM_Dependency (Uses Directory Services From)	Conditional	Conditional requirement: Required if LocalFileSystem.DirectoryServiceUsage is either "Required" or "Optional". Associates a ComputerSystem that indicates a directory service that supports the dependent LocalFileSystem.
8.5.2 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to LocalFileSystem)	Optional	Experimental. Expressed the ability for the file system to be named or have its state changed.
8.5.3 CIM_ElementSettingData (FileSystem)	Optional	Associates a LocalFileSystem to its FileSystemSetting element.

**Table 95 - CIM Elements for Filesystem**

Element Name	Requirement	Description
8.5.4 CIM_ElementSettingData (Local Access Required)	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. Associates a LocalFileSystem to LocallyAccessibleFileSystemSetting elements, one for each file server that has local access.
8.5.5 CIM_EnabledLogicalElementCapabilities (LocalFileSystem)	Optional	Experimental. This class is used to express the naming and possible requested state change possibilities for file systems.
8.5.6 CIM_FileStorage	Mandatory	Associates a LogicalFile (or Directory) to the LocalFileSystem that contains it. This is provided for backward compatibility with previous versions of SMI-S.
8.5.7 CIM_FileSystemSetting	Optional	This element represents the configuration settings of a filesystem represented by a LocalFileSystem.
8.5.8 CIM_HostedDependency (Local Access Required)	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. Associates a file server ComputerSystem to the LocallyAccessibleFileSystemSetting elements that get scoping information from that file server.
8.5.9 CIM_HostedFileSystem (LocalFileSystem)	Mandatory	Associates a LocalFileSystem to the ComputerSystem that hosts it.
8.5.10 CIM_LocalAccessAvailableToFS	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. Associates a LocalFileSystem to a file server ComputerSystem that can export files or directories as shares.
8.5.11 CIM_LocallyAccessibleFileSystemSetting	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. This element represents the configuration settings of a LocalFileSystem that can be made locally accessible (i.e., can have a file or directory made accessible to operational users) from a file server ComputerSystem. This Setting provides further details on the functionality supported and the parameters of that functionality when locally accessible.
8.5.12 CIM_LocalFileSystem	Mandatory	Represents a filesystem in a Filesystem-related profile.
8.5.13 CIM_LogicalFile	Mandatory	In an earlier release of SMI-S, the Filesystem-related profiles made a limited set of LogicalFiles (or Directory subclass) instances visible (these were any file or directory that was exported as a share. This element is required by the profiles to maintain backward compatibility for clients conforming to earlier versions of SMI-S.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LocalFileSystem AND SourceInstance.CIM_LocalFileSystem::OperationalStatus <> PreviousInstance.CIM_LocalFileSystem::OperationalStatus	Mandatory	CQL -Change of Status of a filesystem. PreviousInstance is optional, but may be supplied by an implementation of the Profile.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LocalFileSystem	Mandatory	CQL -Creation of a LocalFileSystem element.

**Table 95 - CIM Elements for Filesystem**

Element Name	Requirement	Description
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LocalFileSystem	Mandatory	CQL -Deletion of a LocalFileSystem element.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM13'	Optional	Filesystem available space has changed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM14'	Mandatory	Filesystem is inaccessible.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM15'	Mandatory	Filesystem has returned to normal access.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM28'	Mandatory	Filesystem capacity is at warning threshold
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM29'	Mandatory	Filesystem capacity is critical.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM30'	Mandatory	Filsystem capacity issues resolved.

**8.5.1 CIM\_Dependency (Uses Directory Services From)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if LocalFileSystem.DirectoryServiceUsage is either "Required" or "Optional".

Table 96 describes class CIM\_Dependency (Uses Directory Services From).

**Table 96 - SMI Referenced Properties/Methods for CIM\_Dependency (Uses Directory Services From)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The ComputerSystem that indicates the directory service(s) that support user, group and other security principal identities for a filesystem.
Dependent		Mandatory	The LocalFileSystem whose use of user, group, and other security principal identities is supported by the antecedent ComputerSystem.

**8.5.2 CIM\_ElementCapabilities (EnabledLogicalElementCapabilities to LocalFileSystem)**

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 97 describes class CIM\_ElementCapabilities (EnabledLogicalElementCapabilities to LocalFileSystem).

**Table 97 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (EnabledLogicalElementCapabilities to LocalFileSystem)**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The capabilities object associated with the file system.
ManagedElement		Mandatory	The LocalFileSystem.

### 8.5.3 CIM\_ElementSettingData (FileSystem)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Optional

Table 98 describes class CIM\_ElementSettingData (FileSystem).

**Table 98 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (FileSystem)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The LocalFileSystem.
SettingData		Mandatory	The settings established on the LocalFileSystem when first created or as modified.

### 8.5.4 CIM\_ElementSettingData (Local Access Required)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true.

Table 99 describes class CIM\_ElementSettingData (Local Access Required).

**Table 99 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (Local Access Required)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The LocalFileSystem that is being made locally accessible.
SettingData		Mandatory	The local access settings of the LocalFileSystem, specified when first created or established later.

### 8.5.5 CIM\_EnabledLogicalElementCapabilities (LocalFileSystem)

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 100 describes class CIM\_EnabledLogicalElementCapabilities (LocalFileSystem).

**Table 100 - SMI Referenced Properties/Methods for CIM\_EnabledLogicalElementCapabilities (Local-FileSystem)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	The moniker for the instance.
ElementNameEditSupported		Mandatory	Denotes whether a file system can be named.
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See MOF for details.
RequestedStatesSupported		Optional	Expresses the states to which this file system may be changed using the RequestStateChange method. If this property, it may be assumed that the state may not be changed.
GetElementNameCapabilities()		Conditional	Conditional requirement: Required if Filesystem Manipulation is implemented.

### 8.5.6 CIM\_FileStorage

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 101 describes class CIM\_FileStorage.

**Table 101 - SMI Referenced Properties/Methods for CIM\_FileStorage**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The LocalFileSystem that contains the LogicalFile.
PartComponent		Mandatory	The LogicalFile contained in the LocalFileSystem.

### 8.5.7 CIM\_FileSystemSetting

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 102 describes class CIM\_FileSystemSetting.

**Table 102 - SMI Referenced Properties/Methods for CIM\_FileSystemSetting**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for this FileSystemSetting element.
ElementName		Mandatory	A user-friendly name for this FileSystemSetting element.
ActualFileSystemType		Mandatory	This identifies the type of filesystem that this FileSystemSetting represents.
FilenameCaseAttributes		Mandatory	This specifies the support provided for using upper and lower case characters in a filename.
ObjectTypes		Mandatory	This is an array that specifies the different types of objects that this filesystem may be used to provide and provides further details in corresponding entries in other attributes.
NumberOfObjectsMin		Mandatory	This is an array that specifies the minimum number of objects of the type specified by the corresponding entry in ObjectTypes[].
NumberOfObjectsMax		Mandatory	This is an array that specifies the maximum number of objects of the type specified by the corresponding entry in ObjectTypes[].
NumberOfObjects		Mandatory	This is an array that specifies the expected number of objects of the type specified by the corresponding entry in ObjectTypes[].
ObjectSize		Mandatory	This is an array that specifies the expected size of a typical object of the type specified by the corresponding entry in ObjectTypes[].
ObjectSizeMin		Mandatory	This is an array that specifies the minimum size of an object of the type specified by the corresponding entry in ObjectTypes[].
ObjectSizeMax		Mandatory	This is an array that specifies the minimum size of an object of the type specified by the corresponding entry in ObjectTypes[].
FilenameReservedCharacterSet		Optional	This string or character array specifies the characters reserved (i.e., not allowed) for use in filenames of a filesystem with this setting.
DataExtentsSharing		Optional	This specifies whether a filesystem with this setting supports the creation of data blocks (or storage extents) that are shared between files.
CopyTarget		Optional	This specifies that, if possible, support should be provided for using a filesystem created with this setting as a target of a Copy operation.
FilenameStreamFormats		Optional	This is an array that specifies the stream formats (e.g., UTF-8) supported for filenames by a filesystem with this setting.
FilenameFormats		Optional	This is an array that specifies the formats (e.g. DOS 8.3 names) supported for filenames by a filesystem with this setting.
FilenameLengthMax		Optional	This specifies the maximum length of a filename supported by a filesystem with this setting.
SupportedLockingSemantics		Optional	This array specifies the set of file access/locking semantics supported by a filesystem with this setting.
SupportedAuthorizationProtocols		Optional	This array specifies the kind of file authorization protocols supported by a filesystem with this setting.
SupportedAuthenticationProtocols		Optional	This array specifies the kind of file authentication protocols supported by a filesystem with this setting.

### 8.5.8 CIM\_HostedDependency (Local Access Required)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true.

Table 103 describes class CIM\_HostedDependency (Local Access Required).

**Table 103 - SMI Referenced Properties/Methods for CIM\_HostedDependency (Local Access Required)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The ComputerSystem that provides the scope for the LocallyAccessibleFileSystemSetting.
Dependent		Mandatory	The local access settings of the LocalFileSystem, established when first created or as modified later, that is dependent on some information provided by the file server that is the scoping ComputerSystem.

### 8.5.9 CIM\_HostedFileSystem (LocalFileSystem)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Mandatory

Table 104 describes class CIM\_HostedFileSystem (LocalFileSystem).

**Table 104 - SMI Referenced Properties/Methods for CIM\_HostedFileSystem (LocalFileSystem)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that hosts a LocalFileSystem.
PartComponent		Mandatory	The LocalFileSystem that represents the hosted filesystem.

### 8.5.10 CIM\_LocalAccessAvailableToFS

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=2 (Required) or 3 (Required - vendor default).

Table 105 describes class CIM\_LocalAccessAvailableToFS.

**Table 105 - SMI Referenced Properties/Methods for CIM\_LocalAccessAvailableToFS**

Properties	Flags	Requirement	Description & Notes
LocalAccessPoint		Optional	The name used by the file server ComputerSystem to identify the filesystem. Sometimes referred to as a mount-point.  For many UNIX-based systems, this will be a qualified full pathname.  For Windows systems this could also be the drive letter used for the LogicalDisk that the filesystem is resident on.
FileSystem		Mandatory	The LocalFileSystem that is being made available to the file server ComputerSystem.
FileServer		Mandatory	The ComputerSystem that will be able to export shares from this LocalFileSystem.

### 8.5.11 CIM\_LocallyAccessibleFileSystemSetting

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true.

Table 106 describes class CIM\_LocallyAccessibleFileSystemSetting.

**Table 106 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for a LocallyAccessibleFileSystemSetting.
ElementName		Mandatory	A user-friendly name for this LocallyAccessibleFileSystemSetting element.
InitialEnabledState		Optional	InitialEnabledState is an integer enumeration that indicates the enabled/disabled states initially set for a locally accessible filesystem (LAFS). The element functions by passing commands onto the underlying filesystem, and so cannot indicate transitions between requested states because those states cannot be requested. The following text briefly summarizes the various enabled/disabled initial states:  Enabled (2) indicates that the element will execute commands, will process any queued commands, and will queue new requests.  Disabled (3) indicates that the element will not execute commands and will drop any new requests.  In Test (7) indicates that the element will be in a test state.  Deferred (8) indicates that the element will not process any commands but will queue new requests.  Quiesce (9) indicates that the element is enabled but in a restricted mode. The element's behavior is similar to the Enabled state, but it only processes a restricted set of commands. All other requests are queued.
OtherEnabledState		Optional	A string describing the element's initial enabled/disabled state when the InitialEnabledState property is set to 1 ("Other"). This property MUST be set to NULL when InitialEnabledState is any value other than 1.



**Table 106 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
FailurePolicy		Optional	An enumerated value that specifies if the operation to make a filesystem locally accessible to a scoping ComputerSystem should be attempted one or more times in the foreground or tried repeatedly in the background until it succeeds. The number of attempts would be limited by the corresponding RetriesMax property of the setting.
RetriesMax		Optional	An integer specifying the maximum number of attempts that should be made by the scoping ComputerSystem to make a LocalFileSystem locally accessible. A value of '0' specifies an implementation-specific default.
RequestRetryPolicy		Optional	An enumerated value representing the policy that is supported by the operational file server on a request to the operational filesystem that either failed or left the file server hanging. If the request is being performed in the foreground, the options are to try once and fail if a timeout happens, or, to try repeatedly. If the request can be performed in the background, the request will be tried repeatedly until stopped.
TransmissionRetriesMax		Optional	An integer specifying the maximum number of retransmission attempts to be made from the operational file server to the operational filesystem when the transmission of a request fails or makes the file server hang. A value of '0' specifies an implementation-specific default. This is only relevant if there is a transmission channel between the file server and the underlying filesystem.
RetransmissionTimeoutMin		Optional	An integer specifying the minimum number of milliseconds that the operational file server must wait before assuming that a request to the operational filesystem has failed. '0' indicates an implementation-specific default. This is only relevant if there is a transmission channel between the operational file server and the operational filesystem.
CachingOptions		Optional	An enumerated value that specifies if a local cache is supported by the operational file server when accessing the underlying operational filesystem.
BuffersSupport		Optional	An array or enumerated values that specifies the buffering mechanisms supported by the operational file server for accessing the underlying operational filesystem." If supported, other properties will establish the level of support. If the property is NULL or the empty array, buffering is not supported.
ReadBufferSizeMin		Optional	An integer specifying the minimum number of bytes that must be allocated to each buffer used for reading. A value of '0' specifies an implementation-specific default.
ReadBufferSizeMax		Optional	An integer specifying the maximum number of bytes that may be allocated to each buffer used for reading. A value of '0' specifies an implementation-specific default.
WriteBufferSizeMin		Optional	An integer specifying the minimum number of bytes that must be allocated to each buffer used for writing. A value of '0' specifies an implementation-specific default.
WriteBufferSizeMax		Optional	An integer specifying the maximum number of bytes that may be allocated to each buffer used for writing. A value of '0' specifies an implementation-specific default.

**Table 106 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
AttributeCaching		Optional	<p>An array of enumerated values that specify whether attribute caching is (or is not) supported by the operational file server when accessing specific types of objects from the underlying operational filesystem. The object type and the support parameters are specified in the corresponding AttributeCachingObjects, AttributeCachingTimeMin, and AttributeCachingTimeMax array properties.</p> <p>Object types contained by a filesystem that can be accessed locally are represented by an entry in these arrays. The entry in the AttributeCaching array can be 'On', 'Off', or 'Unknown'. Implementation of this feature requires support from other system components, so it is quite possible that specifying 'On' may still not result in caching behavior. 'Unknown' indicates that the access operation will try to work with whatever options the operational file server and filesystem can support. In all cases, AttributeCachingTimeMin and AttributeCachingTimeMax provide the minimum and maximum time for which the attributes can be cached. When this Setting is used as a Goal, the client may specify 'Unknown', but the Setting in the created object should contain the supported setting, whether 'On' or 'Off'.</p>
AttributeCachingObjects		Optional	<p>An array of enumerated values that specify the attribute caching support provided to various object types by the operational file server when accessing the underlying operational filesystem. These", types represent the types of objects stored in a filesystem - files and directories as well as others that may be defined in the future. The corresponding properties, AttributeCaching, AttributeCachingTimeMin, and AttributeCachingTimeMax provide the supported features for the type of object. 'None' and 'All' cannot both be specified; if either one is specified, it must be the first entry in the array and the entry is interpreted as the default setting for all objects. If neither 'None' or 'All' are specified, the caching settings for other objects are defaulted by the implementation. If 'Rest' is specified, the entry applies to all known object types other than the named ones. If 'Unknown' is specified it applies to object types not known to this application (this can happen when foreign filesystems are made locally accessible).</p>
AttributeCachingTimeMin		Optional	<p>An array of integers specifying, in milliseconds, the minimum time for which an object of the type specified by the corresponding AttributeCaching property must be retained in the attribute cache. When used as a Goal, a value of '0' indicates an implementation-specific default.</p>
AttributeCachingTimeMax		Optional	<p>An array of integers specifying, in milliseconds, the maximum time for which an object of the type specified by the corresponding AttributeCaching property must be retained in the attribute cache. When used as a Goal, a value of '0' indicates an implementation-specific default.</p>
ReadWritePolicy		Optional	<p>An enumerated value that specifies the Read-Write policy set on the operational filesystem and supported by the operational file server when accessing it. 'Read Only' specifies that the access to the operational filesystem by the operational file server is set up solely for reading. 'Read/Write' specifies that the access to the operational filesystem by the operational file server is set up for both reading and writing. 'Force Read/Write' specifies that 'Read-Only' has been overridden by a client with write access to the operational filesystem. This option is intended for use when the associated filesystem has been made 'Read Only' by default, as might happen if it were created to be the target of a Synchronization or Mirror operation.</p>

**Table 106 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
LockPolicy		Optional	An enumerated value that specifies the Locking that will be enforced on the operational filesystem by the operational file server when accessing it. 'Enforce None' does not enforce locks. 'Enforce Write' does not allow writes to locked files. 'Enforce Read/Write' does not allow reads or writes to locked files.
EnableOnSystemStart		Optional	An enumerated value that specifies if local access from the operational file server to the operational filesystem should be enabled when the file server is started.
ReadWritePref		Optional	An instance of a CIM_Privilege, encoded as a string, that expresses the client's expectations about access to elements contained in the operational filesystem. The provider is expected to surface this access using the CIM privilege model.
ExecutePref		Optional	An enumerated value that specifies if support should be provided on the operational file server for executing elements contained in the operational filesystem accessed through this local access point. This may require setting up specialized paging or execution buffers either on the operational file server or on the operational filesystem side (as appropriate for the implementation). Note that this does not provide any rights to actually execute any element but only specifies support for such execution, if permitted.
RootAccessPref		Optional	An instance of a CIM_Privilege, encoded as a string, that expresses the client's expectations about privileged access by appropriately privileged System Administrative users on the operational file server ('root' or 'superuser') to the operational filesystem and its elements. The provider is expected to surface this access using the CIM privilege model.  Support for the privileged access might require setup at both the operational file server as well as the operational filesystem, so there is no guarantee that the request can be satisfied.

**8.5.12 CIM\_LocalFileSystem**

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 107 describes class CIM\_LocalFileSystem.

**Table 107 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
CSCreationClassName		Mandatory	The CIM class of the hosting ComputerSystem element.
CSName		Mandatory	The Name property of the hosting ComputerSystem element.
CreationClassName		Mandatory	The CIM class of this LocalFileSystem element.
Name		Mandatory	A unique name for this LocalFileSystem element in the context of the hosting ComputerSystem.
OperationalStatus		Mandatory	The current operational status of the filesystem represented by this LocalFileSystem element.

**Table 107 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
Root		Optional	A path that specifies the "mount point" of the filesystem in an unitary computer system that is both the host of the filesystem and is the file server that makes it available.
BlockSize		Optional	The size of a block in bytes for certain filesystem types that require a fixed block size when creating a filesystem.
FileSystemSize		Optional	The total current size of the filesystem in blocks.
AvailableSpace		Optional	The space available currently in the filesystem in blocks. NOTE: This value is an approximation as it can vary continuously when the filesystem is in use.
ReadOnly		Optional	Indicates that this is a read-only filesystem that does not allow modifications to contained files and directories.
EncryptionMethod		Optional	Indicates if files are encrypted by the filesystem implementation and the method of encryption.
CompressionMethod		Optional	Indicates if files are compressed by the filesystem implementation before being stored, and the methods of compression.
CaseSensitive		Mandatory	Whether this filesystem is sensitive to the case of characters in filenames.
CasePreserved		Mandatory	Whether this filesystem implementation preserves the case of characters in filenames when saving and restoring.
CodeSet		Optional	The codeset used in filenames by the filesystem implementation.
MaxFileNameLength		Mandatory	The length of the longest filename supported by the filesystem implementation.
FileSystemType		Mandatory	This is a string that matches FileSystemSetting.ActualFileSystemType property used to create the filesystem.
NumberOfFiles		Optional	The actual current number of files in the filesystem. NOTE: This value is an approximation as it can vary continuously when the filesystem is in use.
LocalAccessDefinitionRequired		Mandatory	This boolean property indicates whether or not this LocalFileSystem must be made locally accessible ("mounted") from a file server ComputerSystem before it can be shared or otherwise made available to operational clients.

**Table 107 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
PathNameSeparatorString		Mandatory	This indicates the string of characters used to separate directory components of a canonically formatted path to a file from the root of the filesystem. This string is expected to be specific to the ActualFileSystemType and so is vendor/implementation dependent. However, by surfacing it we make it possible for a client to parse a pathname into the hierarchical sequence of directories that compose it.
DirectoryServiceUsage		Optional	<p>This enumeration indicates whether the filesystem supports security principal information and therefore requires support from a file server that uses one or more directory services. If the filesystem requires such support, there must be a concrete subclass of Dependency between the LocalFileSystem element and the specified file server ComputerSystem. The values supported by this property are:</p> <p>"Not Used" indicates that the filesystem will not support security principal information and so will not require support from a directory service.</p> <p>"Optional" indicates that the filesystem may support security principal information. If it does, it will require support from a directory service and the Dependency association described above must exist.</p> <p>"Required" indicates that the filesystem supports security principal information and will require support from a directory service. The Dependency association described above must exist.</p>

**8.5.13 CIM\_LogicalFile**

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 108 describes class CIM\_LogicalFile.

**Table 108 - SMI Referenced Properties/Methods for CIM\_LogicalFile**

Properties	Flags	Requirement	Description & Notes
CSCreationClassName		Mandatory	Class Name of the ComputerSystem that hosts the filesystem containing this file.
CSName		Mandatory	The Name property of the ComputerSystem that hosts the filesystem containing this file.
FSCreationClassName		Mandatory	Class Name of the LocalFileSystem that represents the filesystem containing this file.
FSName		Mandatory	The Name property of the LocalFileSystem that represents the filesystem containing this file.
CreationClassName		Mandatory	Class Name of this instance of LogicalFile that represents the file.
Name		Mandatory	The Name property of the LogicalFile that represents the file.
ElementName		Mandatory	The pathname from the root of the containing LocalFileSystem to this LogicalFile. The root of the LocalFileSystem is indicated if this is NULL or the empty string. The format of the pathname is specific to the LocalFileSystem's FileSystemType. If it is a sequence of directories from the root, the separator string is specified by the CIM_LocalFileSystem.PathNameSeparatorString property.

**STABLE**

---

---

---

## STABLE

### 9 Filesystem Manipulation Profile

#### 9.1 Description

##### 9.1.1 Synopsis

**Profile Name:** Filesystem Manipulation (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** FileSystemConfigurationService

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 109 describes the related profiles for Filesystem Manipulation.

**Table 109 - Related Profiles for Filesystem Manipulation**

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	Experimental.
Filesystem	SNIA	1.7.0	Mandatory	
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2
Volume Composition	SNIA	1.5.0	Optional	

##### 9.1.2 Overview

The Filesystem Manipulation Profile is a component profile that provides support for configuring and manipulating filesystems in the context of filesystem profiles (currently consisting of the NAS Head and the Self-Contained NAS Profiles). A number of other profiles make use of elements of the Filesystem profiles and will be referred to in this specification as “filesystem-related profiles” -- these include, but are not limited to, the Filesystem Profile, File Export Profile, File Export Manipulation Profile, and NAS Head Profile.

The state transitions involved when an appropriate storage element is transformed into a filesystem are described in Annex B (Informative) State Transitions from Storage to File Shares.

##### 9.1.3 Instance Diagrams

###### 9.1.3.1 Filesystem Creation classes and associations

Figure 14 illustrate the constructs involved with creating a LocalFileSystem for a Filesystem Profile. This summarizes the mandatory classes and associations for this profile. Specific areas are discussed in later sections.

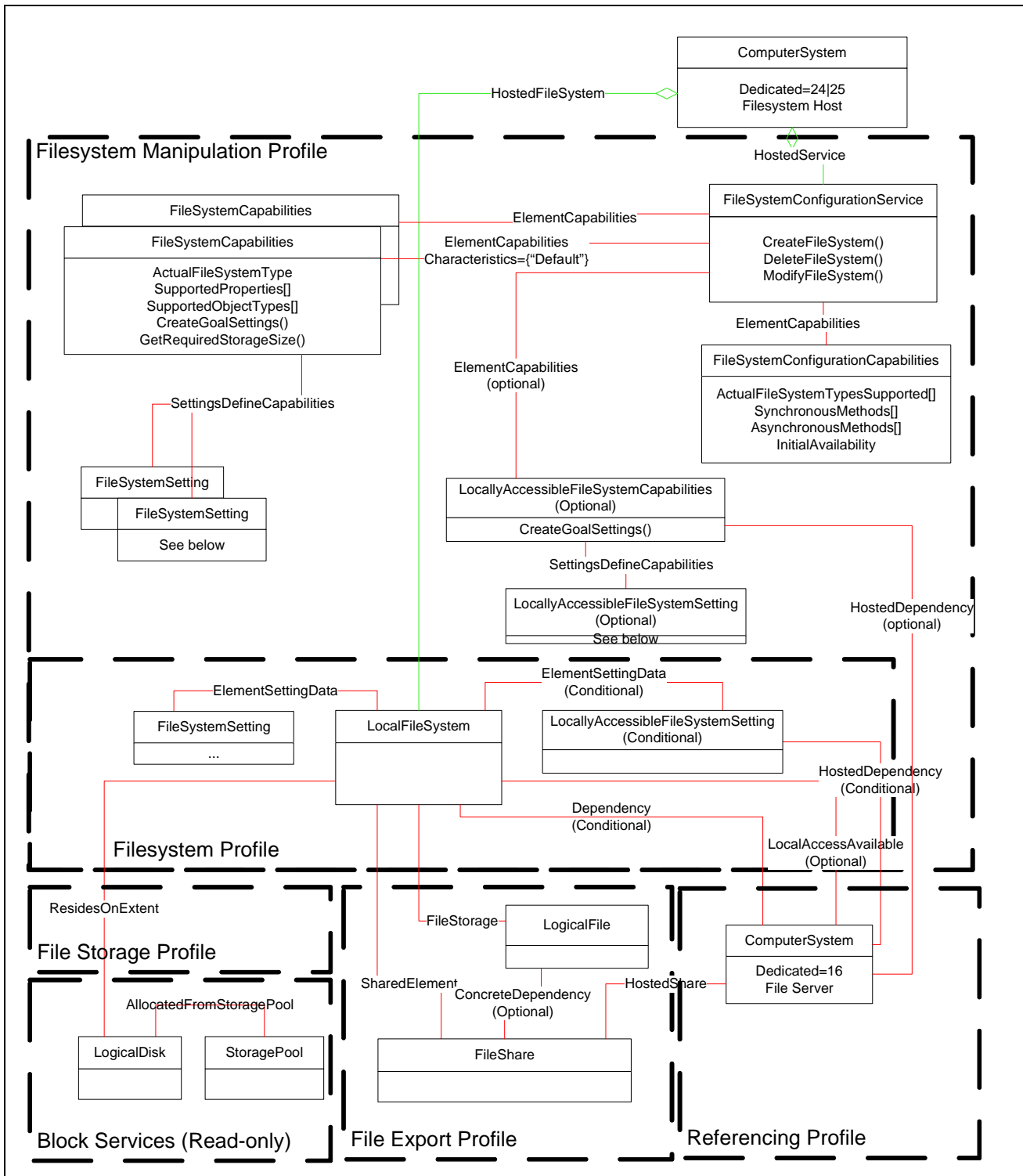


Figure 14 - LocalFileSystem Creation Instance Diagram

If a filesystem-related profile supports the Filesystem Manipulation Profile, it shall have at least one instance of the FileSystemConfigurationService. This service shall be hosted on the top level



ComputerSystem of the filesystem-related profile. The methods offered are CreateFileSystem, ModifyFileSystem, and DeleteFileSystem.

Associated to the FileSystemConfigurationService (via ElementCapabilities) shall be one instance of FileSystemConfigurationCapabilities that describes the capabilities of the service. It identifies the methods supported, whether the methods support Job Control or not, the types of filesystems that are supported, and whether or not the filesystem shall be made operationally available after creation.

For each type of filesystem that can be created, there shall be one FileSystemCapabilities element that defines the range of capabilities supported for that particular filesystem type. An ElementCapabilities association links each FileSystemCapabilities to the FileSystemConfigurationService. One of these FileSystemCapabilities may also be identified as a default capability (by setting "Default" as one of the entries in the array property Characteristics of its ElementCapabilities association). This default FileSystemCapabilities element is used when the client does not specify a goal element when requesting the CreateFileSystem method. The default FileSystemCapabilities element implicitly indicates the default filesystem type to be used for creation.

For the convenience of clients a Filesystem Manipulation Profile shall populate a set of "pre-defined" FileSystemSettings for each of the FileSystemCapabilities. These shall be associated to the FileSystemCapabilities via the SettingsDefineCapabilities association (the association must have its SettingsDefineCapabilities.PropertyPolicy property set to "Correlated" and the SettingsDefineCapabilities.ValueRole property must include "Supported" as an entry) and shall be for the same filesystem type as the associated capabilities element (same value for the ActualFileSystemType property in both classes).

NOTE That they are pre-defined and exist at all times does not imply that these FileSystemSettings must be made persistent by the implementation -- rather it should be possible for the implementation to regenerate them if requested, though a simple regenerating implementation may not necessarily scale.

The FileSystemCapabilities element supports three methods: CreateGoalSettings, GetRequiredStorageSize and GetElementNameCapabilities. These methods are described in detail in 9.3.1, "Extrinsic Methods of the Profile". The basic function of the first two is to establish at least one client-approved FileSystemSettings element (as a goal) and to determine the size of the LogicalDisk required to support the desired filesystem.

CreateGoalSettings takes an array of embedded-instance SettingData elements as the input TemplateGoalSettings and SupportedGoalSettings parameters and can generate an array of embedded-instance SettingData elements as the output SupportedGoalSettings parameter. However, in this profile, SMI-S only uses a single embedded-instance FileSystemSetting element in the input parameters (both TemplateGoalSettings and SupportedGoalSettings) and generate a single valid embedded-instance FileSystemSetting element as output (SupportedGoalSettings). If a client supplies a NULL (or the empty string) FileSystemSetting as input to this method, the returned FileSystemSetting embedded-instance shall be a default setting for the ActualFileSystemType of the FileSystemCapabilities. If the input (the embedded-instance FileSystemSetting element) is not NULL, the method may return a "best fit" to the requested setting. The client may iterate on this method until it acquires a setting that suits its needs. This embedded-instance settings structure may be used when the CreateFileSystem or ModifyFileSystem methods are invoked. The details of how iterative negotiation can work are discussed in 9.3.1.1, "FileSystemCapabilities.CreateGoalSettings". Note that the FileSystemType remains unchanged in all of these interactions. It is an error if the client or server changes the FileSystemType unilaterally.

NOTE It is not possible to guarantee that negotiation will terminate with an agreed upon setting and a fall-back mechanism is needed. This profile does not require negotiation -- an implementation may support only a set of pre-defined correlated point settings that a client can preload and use without modification. The implementation could also support only settings whose properties are selectable from an arithmetic progression or from a fixed enumeration, so that the client may construct a goal setting that is guaranteed to be supported without negotiation.

NOTE That a client has obtained a goal setting supported by the implementation does not guarantee that a create or modify request will succeed. Such a setting only specifies a supported static configuration not that the current dynamic environment has the resources to implement a specific request.

After obtaining a goal `FileSystemSetting`, the next step is to determine the `LogicalDisk` size required to support the `FileSystemSetting`. This is done by invoking the `FileSystemCapabilities.GetRequiredStorageSize` method of this profile. The inputs are the embedded-instance `FileSystemSetting` structure and an embedded-instance `StorageSetting` structure that describes the quality of service the client wants for the storage (e.g., data redundancy, package redundancy, etc.). The method returns three numbers corresponding to the `StorageSetting`: the expected size, the minimum size, and a maximum usable size. The client would use these numbers in specifying or evaluating the appropriate `LogicalDisk(s)` on which to create the `FileSystem`. The method also returns as output the actual `StorageSetting` used as an `EmbeddedInstance` structure (assuming that these can be substituted for the input `StorageSetting`).

NOTE This profile recommends that `GetRequiredStorageSize()` be used only if a `LocalFileSystem` is to be created on a single `LogicalDisk`. If the intent is to use more than one `LogicalDisk` for the `LocalFileSystem`, this profile recommends using the `CreateFileSystem` method to make the implementation create or select the `LogicalDisks` to use.

- Armed with the `FileSystem` goal (the embedded-instance `FileSystemSetting` structure) and one or more `LogicalDisks`, the client can now use the `CreateFileSystem` method to create the filesystem. The `CreateFileSystem` method creates a `LocalFileSystem` instance, and a new `FileSystemSetting` instance as well as several necessary associations. These associations are:
- `HostedFileSystem` association between the `LocalFileSystem` and the `ComputerSystem` that hosts it
- `ResidesOnExtent` association between the filesystem and one of the `LogicalDisk(s)` for the filesystem data

NOTE Even when multiple `LogicalDisks` are created or used for the `LocalFileSystem`, only one `LogicalDisk` will have the `ResidesOnExtent` association.

- `ElementSettingData` to associate the filesystem to the `FileSystemSetting` defined for it

`CreateFileSystem` allows `LogicalDisks` to be obtained from a number of `StoragePools` using an array of embedded-instance `StorageSettings`. The `CreateFileSystem` implementation must use the capabilities of the `StoragePools` (and the associated `StorageConfigurationService`) to create the necessary `LogicalDisks`. The `LogicalDisks` used for this purpose are returned as output values for the `InExtents` parameter.

To determine if the implementation supports supplying the `ElementName` during the creation of a file system and to determine the supported methods to modify the `ElementName` of the existing file system, invoke the method `FileSystemCapabilities.GetElementNameCapabilities`.

If the property `FileSystemConfigurationCapabilities.LocalAccessibilitySupport` specifies that `CreateFileSystem` method provides the optional parameters for establishing local access ("mounting") from file server `ComputerSystems`, the `LocalFileSystem.LocalAccessDefinitionRequired` will be set to true and the `LocalFileSystem` will need to be made locally accessible from the specified file server `ComputerSystems`. The following elements are created:

- A `LocalAccessAvailableToFS` association representing local access by a file server `ComputerSystem` to the `LocalFileSystem` from within its namespace is created using the provided (or defaulted) `LocallyAccessibleFileSystemSetting` parameter (as an `EmbeddedInstance` parameter). This association goes between the File Server `ComputerSystem` and the `LocalFileSystem`; its `LocalAccessPoint` property specifies the local access point (or "mount-point") in the file server `ComputerSystem`.
- An instance of `LocallyAccessibleFileSystemSetting` is optionally created and associated to:
  - The `LocalFileSystem` via an optional `ElementSettingData` association.
  - The file server `ComputerSystem` via an optional `HostedDependency` (`ScopedSetting`) association (this represents that a `LocalFileSystem` could be mounted on many file server `ComputerSystems` with different "mount" parameters, so these parameters are specific to the pair of `LocalFileSystem` and file server `ComputerSystem`).

- For backward compatibility with previous releases of SMI-S:
  - The root directory of the LocalFileSystem is represented as a LogicalFile
  - A FileStorage association is created between the LogicalFile and the LocalFileSystem

The DeleteFileSystem method is straightforward -- it deletes the LocalFileSystem and the FileSystemSetting, and the associations to those instances (HostedFileSystem, both ElementSettingData elements, ResidesOnExtent, LocalAccessAvailableToFS, and LocallyAccessibleFileSystemSetting). Any created LogicalFiles associated to the LocalFileSystem via FileStorage will also be deleted as a side-effect of deleting the LocalFileSystem (so there is no separate requirement necessary for backward compatibility). The implementation may delete the LogicalDisk(s), however, this is not required by this profile. If the LogicalDisk(s) are not deleted, they become available for use in another CreateFileSystem operation.

The ModifyFileSystem method modifies an existing LocalFileSystem -- this requires a new FileSystemSetting structure to be used as a goal. But not any FileSystemSetting structure will do -- the client must use one created with the same FileSystemCapabilities.CreateGoalSettings method that would have been used to create the filesystem, or an appropriate compatible FileSystemCapabilities instance. The CreateGoalSettings method is used to establish a new FileSystemSetting goal (as with the original filesystem creation, it may be necessary to iterate on the CreateGoalSettings method). Since only properties controlled by Settings can be changed by ModifyFileSystem (i.e., the LogicalDisk(s) already created cannot be changed, though new ones can be created and/or added), the effect of ModifyFileSystem is to change some properties of the LocalFileSystem or of the associated FileSystemSetting.

NOTE Depending on what property is being modified, it may also be necessary to invoke the GetRequiredStorageSize method to verify that the current LogicalDisk still supports the new goals.

#### 9.1.3.1.1 Dependency on support for Locally Accessible Filesystem Capabilities

Both CreateFileSystem and ModifyFileSystem need a LocallyAccessibleFileSystemSetting element for each file server ComputerSystem. The client first obtains a LocallyAccessibleFileSystemCapabilities element by following ElementCapabilities association from the FileSystemConfigurationService to a LocallyAccessibleFileSystemCapabilities that is associated via ScopedCapabilities (HostedDependency) to the File Server ComputerSystem.

NOTE It is expected that there will only be one LocallyAccessibleFileSystemCapabilities element per file server ComputerSystem. All the variability can be found by following SettingsDefineCapabilities to LocallyAccessibleFileSystemSetting elements. It is a requirement that the LocallyAccessibleFileSystemSettings that define the LocallyAccessibleFileSystemCapabilities be associated via HostedDependency (ScopedSetting) to the same file server ComputerSystem as the one indicated by the HostedDependency (ScopedCapabilities).

The client calls LocallyAccessibleFileSystemCapabilities.CreateGoalSettings() with the appropriate parameters to obtain an appropriate LocallyAccessibleFileSystemSetting element -- CreateGoalSettings can be used to negotiate if necessary.

#### 9.1.3.1.2 Dependency on support for Directory Services

A filesystem may support security principal identifiers associated with filesystem objects for access (typically, read/write/execute) as well as for tracking usage (as would be needed for supporting user and/or group quotas). If the filesystem supports such identifiers, it would require support from a directory service for validating these identifiers (relating them to accounts and other user-related information). Operationally, computer systems (and not filesystems) are associated to directory services or configured for directory services. Directory service configurations of computer systems are much more complex than needed or appropriate for filesystems. This makes it easier to make the filesystem depend on a computer system, usually a file server, for providing access to directory services for resolving security principal identifiers.

A filesystem that requires support from a directory service will have the property `DirectoryServicesUsage` of its `LocalFileSystem` element set to "Required". In that case, there shall be a `Dependency` association between the `LocalFileSystem` element and a file server `ComputerSystem` element (with `Dedicated="16"`). The associated file server must be configured for access to directory services that it provides for the filesystem.

NOTE If `DirectoryServicesUsage` is "Optional", a client can look for the `Dependency` association to determine if the filesystem supports security principal identifiers. This is not supported in this release of the profile.

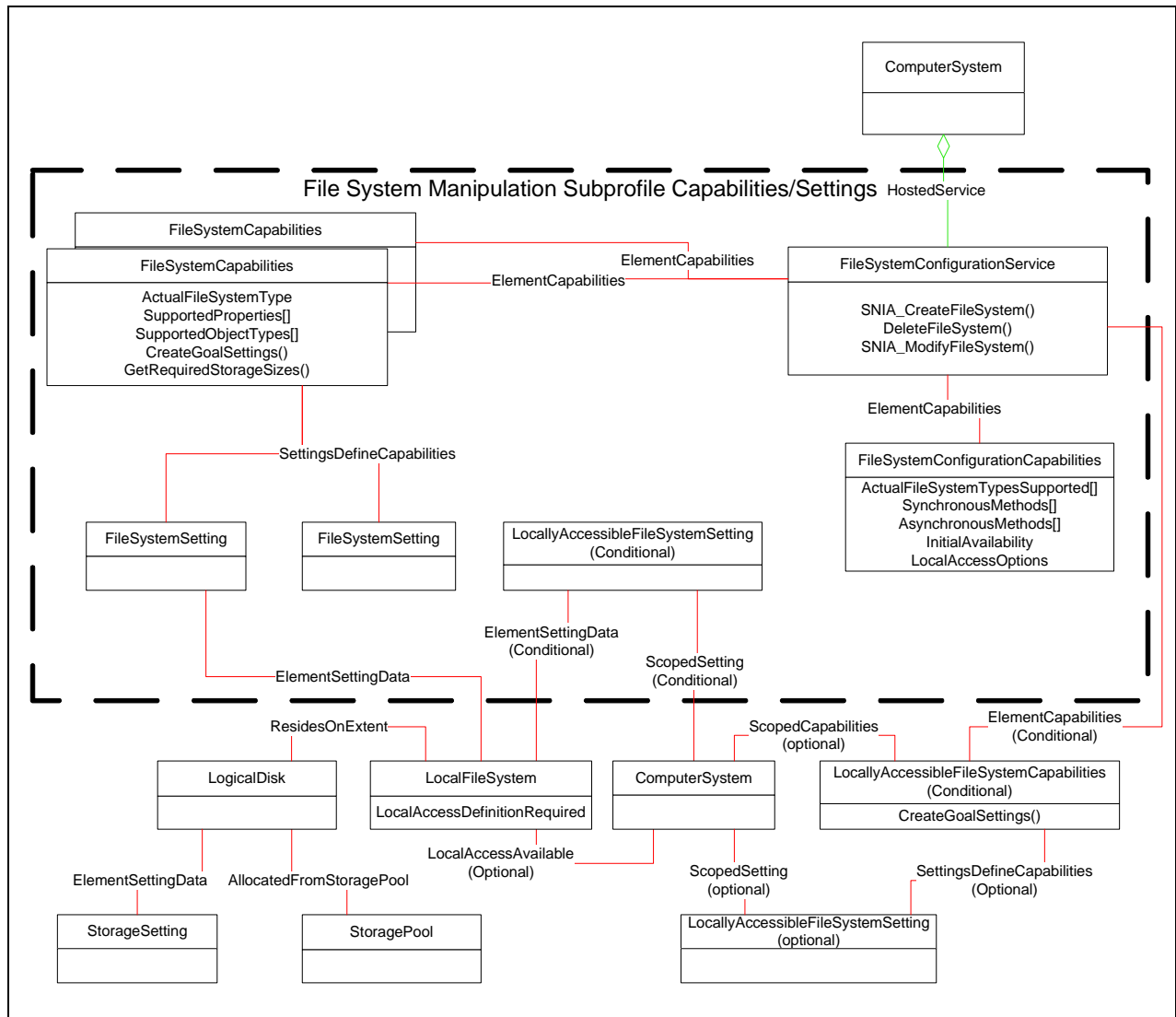
#### 9.1.3.1.3 Summary of this profile

This profile consists of the following classes, associations, and methods:

- 1) One `LocallyAccessibleFileSystemCapabilities` scoped to the file server `ComputerSystem`
- 2) `ElementCapabilities` association to the `FileSystemConfigurationService`
- 3) `SettingsDefineCapabilities` association to `LocallyAccessibleFileSystemSetting`
- 4) `LocallyAccessibleFileSystemSetting` for defining `LocallyAccessibleFileSystemCapabilities`
- 5) `HostedDependency` (`ScopedSetting`) association from the File Server `ComputerSystem` to `LocallyAccessibleFileSystemSetting`
- 6) A `HostedDependency` association from the same file server `ComputerSystem` to the defined `LocallyAccessibleFileSystemCapabilities`
- 7) `LocallyAccessibleFileSystemCapabilities.CreateGoalSettings` extrinsic method to create `LocallyAccessibleFileSystemSetting` elements scoped to the file server `ComputerSystem` to use as Goals. Note that this method is different from the method described as part of the `FileSystemCapabilities` element.
- 8) A `Dependency` association from the `LocalFileSystem` element to a file server `ComputerSystem`.

### 9.1.3.2 Finding FileSystem Configuration Services, Capabilities and Pre-defined Settings

When creating a filesystem the first step is to determine what can be created. Figure 15 illustrates an instance diagram showing the instances that shall exist for supporting filesystem creation.



### Figure 15 - Capabilities and Settings for Filesystem Creation

At least one `FileSystemConfigurationService` shall exist if the Filesystem Profile has implemented the Filesystem Manipulation Profile. The instance(s) of this service can be found by following the `HostedService` association filtering on the target class of `FileSystemConfigurationService`.

NOTE If no service is found from the Top Level ComputerSystem, the client should look for component computer systems that may be hosting the service. This is not recommended, but permitted.

An instance of the `FileSystemConfigurationCapabilities` shall be associated to the `FileSystemConfigurationService` via the `ElementCapabilities` association. A client should follow this association (filtering on the result value of "FileSystemConfigurationCapabilities") to inspect the configuration capabilities that are supported. The client would choose between the filesystem types specified in the array property `SupportedActualFileSystemTypes`.

For each entry in the SupportedActualFileSystemTypes array, there shall be one instance of FileSystemCapabilities with the same value for the ActualFileSystemType property that shall be

associated to the FileSystemConfigurationService using the ElementCapabilities association (filtering on the result value of FileSystemCapabilities). This FileSystemCapabilities element shall specify the supported capabilities for that ActualFileSystemType using a collection of FileSystemSettings. These FileSystemSettings shall be associated to the FileSystemCapabilities via SettingsDefineCapabilities.

An implementation may support a set of pre-defined FileSystemSettings that clients could use directly if desired. The SettingsDefineCapabilities association from the FileSystemCapabilities to the pre-defined FileSystemSettings shall have the PropertyPolicy property be "Correlated", the ValueRole property be "Supported" and the ValueRange property be "Point". Other pre-defined combinations of property values may be specified by FileSystemSettings whose SettingsDefineCapabilities association has the PropertyPolicy be "Independent", ValueRole property be "Supported" and the ValueRange array property contain "Minimums", "Maximums", or "Increment" (see 9.3.1.1.1 for further details on the interpretation of the ValueRange property). These settings can be used by the client to compose FileSystemSettings that are more likely to be directly usable.

## 9.2 Health and Fault Management Considerations

The key element of this profile is the FileSystemConfigurationService and the hosting ComputerSystem. The operational status of the hosting ComputerSystem should possibly be part of the referring autonomous profile (NAS Head or SC NAS), the Filesystem Profile or in the Multiple Computer System Profile.

### 9.2.1 OperationalStatus for FileSystemConfigurationService

Not defined in this standard.

### 9.2.2 OperationalStatus for LocalFileSystem

Table 110 describes the Operational status for LocalFileSystem.

**Table 110 - LocalFileSystem OperationalStatus**

Primary OperationalStatus	Secondary OperationalStatus	Description
2 "OK"		The filesystem has good status
2 "OK"	4 "Stressed"	The filesystem resources are stressed
2 "OK"	5 "Predictive Failure"	The filesystem might fail because some resource or component is predicted to fail
2 "OK"	16 "Supporting Entity in Error"	The filesystem may be OK, but is not accessible because a supporting entity is not accessible.
3 "Degraded"		The filesystem is operating in a degraded mode. This could be due to the health state of the underlying storage being degraded or in error.
6 "Error"		An error has occurred causing the filesystem to become unavailable. Operator intervention through SMI-S (managing the LocalFileSystem) to restore the filesystem may be possible.
6 "Error"		An error has occurred causing the filesystem to become unavailable. Automated recovery may be in progress.

**Table 110 - LocalFileSystem OperationalStatus**

Primary OperationalStatus	Secondary OperationalStatus	Description
6 "Error"	7 "Non-recoverable Error"	The filesystem is not functioning. Operator intervention through SMI-S will not fix the problem.
6 "Error"	16 "Supporting Entity in Error"	The filesystem is in an error state because a supporting entity is not accessible.
8 "Starting"		The filesystem is in process of initialization and is not yet available operationally.
9 "Stopping"		The filesystem is in process of stopping, and is not available operationally.
10 "Stopped"		The filesystem cannot be accessed operationally because it is stopped -- if this did not happen because of operator intervention or happened in real-time, the OperationalStatus would have been "Lost Communication" rather than "Stopped".
11 "In Service"		The filesystem is offline in maintenance mode, and is not available operationally.
13 "Lost Communications"		The filesystem cannot be accessed operationally -- if this happened because of operator intervention it would have been "Stopped" rather than "Lost Communication".
14 "Aborted"		The filesystem is stopped but in a manner that may have left it in an inconsistent state.
15 "Dormant"		The filesystem is offline; and the reason for not being accessible is unknown.

### 9.3 Methods of the Profile

#### 9.3.1 Extrinsic Methods of the Profile

Table 111 details the filesystem manipulation methods that cause Instance Creation, Deletion or Modification.

**Table 111 - Filesystem Manipulation Methods that cause Instance Creation, Deletion or Modification**

Method	Created Instances	Deleted Instances	Modified Instances
FileSystemConfigurationService. CreateFileSystem	LocalFileSystem FileSystemSetting ElementSettingData ResidesOnExtent HostedFileSystem LogicalDisk(s) StorageSetting(s) LocalAccessAvailableTo FS(s) LocallyAccessibleFileSy stemSetting(s) ElementSettingData(s) HostedDependency Dependency	N/A	StoragePool(s) LogicalDisk(s)
FileSystemConfigurationS ervice.DeleteFileSystem		LocalFileSystem FileSystemSetting ElementSettingData ResidesOnExtent HostedFileSystem LocalAccessAvailableTo FS(s) LocallyAccessibleFileSy stemSetting(s) ElementSettingData(s) HostedDependency Dependency	N/A
FileSystemConfigurationS ervice.ModifyFileSystem	(IF REQUESTED) LogicalDisk(s) StorageSetting(s) LocalAccessAvailableTo FS LocallyAccessibleFileSy stemSetting ElementSettingData(s) HostedDependency	(if Local Access is modified) LocalAccessAvailableTo FS LocallyAccessibleFileSy stemSetting ElementSettingData(s) HostedDependency	FileSystemSetting (if changed) ResidesOnExtent (if added)
FileSystemCapabilities.Cr eateGoalSettings	N/A	N/A	N/A
LocallyAccessibleFileSys temCapabilities.CreateGo alSettings	N/A	N/A	N/A
FileSystemCapabilities.Ge tRequiredStorageSize	N/A	N/A	N/A
GetElementNameCapabili ties	N/A	N/A	N/A



### 9.3.1.1 FileSystemCapabilities.CreateGoalSettings

This extrinsic method of the `FileSystemCapabilities` class validates support for a caller-proposed `FileSystemSetting` passed as the `TemplateGoalSettings` parameter. This profile restricts the usage of this method to a single entry array for both `TemplateGoalSettings` and `SupportedGoalSettings` parameters.

If the input `TemplateGoalSettings` is `NULL` or the empty string, this method returns a single default `FileSystemSetting` in the `SupportedGoalSettings` array. Both `TemplateGoalSettings` and `SupportedGoalSettings` are string arrays containing embedded instances of type `FileSystemSetting`. As such, these settings do not exist in the implementation but are the responsibility of the client.

If the `TemplateGoalSettings` specifies values that cannot be supported, this method shall return an appropriate error and should return a best match for a `SupportedGoalSettings`.

The client and the implementation can engage in a negotiation process that may require iterative calls to this method. To assist the implementation in tracking the progress of the negotiation, the client may pass previously returned values of `SupportedGoalSettings` as the new input value of `SupportedGoalSettings`. The implementation may determine that a step has not resulted in progress if the input and output values of `SupportedGoalSettings` are the same. A client may infer from the same result that the `TemplateGoalSettings` must be modified.

#### 9.3.1.1.1 Client Considerations

It is important to understand that the client is acting as an agent for a human user -- either a "system" administrator, or other entity with administrative privileges with respect to the filesystem or the filesystem host. During negotiation, the client will show the current state to the user, the `SupportedGoalSettings` received to date (either the latest or some subset), and the `TemplateGoalSettings` proposed (the most recent, but possibly more). But the administrator needs a representation of what is available, possibly the range or sets of values that the different setting properties can take. Some decisions are assumed to have been made already, such as the type of filesystem to be created and the number of `LogicalDisks` to use and their `StorageSettings`. It is possible that the `LogicalDisks` for use by this filesystem have already been designated by the user; if not, the `StoragePool(s)` from which they will be created is already designated or will be selected by an independent process.

The `SettingsDefineCapabilities` association from the selected `Capabilities` element provides the information for the client to lay out these options. "Point" settings can be identified using `FileSystemSettings` -- these points can be further qualified to indicate whether these are supported (or not), and even whether they represent some ideal point in the space -- a "minimum", or a "maximum", or an "optimal" point. Other settings can provide ranges for properties, by specifying a minimum, a maximum, an increment, and an arithmetic progression of values can be specified (a continuous range can be specified with a zero increment). Specifying a set of supported values for a property that do not follow some pattern is possible, if a bit tedious.

The set of settings associated via `SettingsDefineCapabilities` are expected to be quite stable -- real systems do not continually vary the functionality they can support. Such variations do occur -- for instance, if a new PCMCIA card is added to a running system -- and the best way for a client to be able to add these to the set of choices presented to a user is to subscribe to indications on new `Capabilities` elements and new instances of `SettingsDefineCapabilities`.

There is no guarantee that a negotiation will terminate successfully with the client and the implementation achieving agreement. The implementation may support some simpler mechanisms, short of fully-fledged negotiation, that would be used by a client to obtain an acceptable `TemplateGoalSettings`. The following two use cases are easily covered:

- 1) Client considers only the canned settings that are specified exactly. The canned settings are specified by the FileSystemSettings that are associated to the FileSystemCapabilities via SettingDefinesCapabilities association with the following property values:
  - SettingDefinesCapabilities.PropertyPolicy = "Correlated"
  - SettingDefinesCapabilities.ValueRole = "Supported"
  - SettingDefinesCapabilities.ValueRange = "Point"
- 2) Client considers canned settings that range over values specified using minimum/increment/maximum for the Setting properties. For example, these could be specified by the FileSystemSettings that are associated to the FileSystemCapabilities via SettingDefinesCapabilities association with the following property values:
  - SettingDefinesCapabilities.ValueRange = "Minimums" or "Maximums" or "Increments"
  - The PropertyPolicy and ValueRole properties of SettingDefinesCapabilities will be appropriately specified

#### 9.3.1.1.2 Signature and Parameters of FileSystemCapabilities.CreateGoalSettings

Table 112 describes the parameters for Extrinsic Method FileSystemCapabilities.CreateGoalSettings.

**Table 112 - Parameters for Extrinsic Method FileSystemCapabilities.CreateGoalSettings**

Parameter Name	Qualifier	Type	Description & Notes
TemplateGoalSettings[]	IN	string	EmbeddedInstance ("CIM_FileSystemSetting")  TemplateGoalSettings is a string array containing embedded instances of class FileSystemSetting, or a derived class. This parameter specifies the client's requirements and is used to locate matching settings that the implementation can support.
SupportedGoalSettings[]	INOUT	string	EmbeddedInstance("CIM_FileSystemSetting")  SupportedGoalSettings is a string array containing embedded instances of class FileSystemSetting, or a derived class. On input, it specifies a previously returned set of Settings that the implementation could support. On output, it specifies a new set of Settings that the implementation can support. If the output set is identical to the input set, both client and implementation may conclude that this is the best match for the TemplateGoalSettings that is available. If the output does not match the input and the non-NULL output does not match the non-NULL TemplateGoalSettings, then the method must return "Alternative Proposed". If the output is NULL, the method must return an "Failed".
Normal Return			
Status		uint32	ValueMap{}, Values{}  "Success", "Failed", "Timeout", "Alternative Proposed"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

### 9.3.1.2 GetRequiredStorageSize

This extrinsic method returns the minimum, expected, and maximum size for a LogicalDisk that would support a filesystem specified by the input FileSystemGoal parameter. The caller may provide relevant settings of the LogicalDisk via the ExtentSetting parameter. The minimum, maximum, and expected sizes are returned as output parameters.

If the input FileSystemGoal is NULL, the implementation may return an error or use a default FileSystemSetting associated with this FileSystemCapabilities element. The actual FileSystemSetting used is returned as an OUT parameter.

If the input ExtentSetting parameter is NULL or is an empty array, the implementation may use a default StorageSetting associated with the StorageConfigurationService hosted on the same ComputerSystem as the FileSystemConfigurationService associated with this FileSystemCapabilities element. The actual StorageSetting used is returned as an OUT parameter.

NOTE The actual FileSystemSetting and StorageSetting used are being returned as OUT parameters. This is a non-backward-compatible change from SMI-S 1.1.

#### 9.3.1.2.1 Signature and Parameters of GetRequiredStorageSize

Table 113 describes the parameters for Extrinsic Method FileSystemCapabilities.GetRequiredStorageSize.

**Table 113 - Parameters for Extrinsic Method FileSystemCapabilities.GetRequiredStorageSize**

Parameter Name	Qualifier	Type	Description & Notes
FileSystemGoal	INOUT, EI	string	EmbeddedInstance ("CIM_FileSystemSetting")  FileSystemGoal is an Embedded Instance element of class CIM_FileSystemSetting, or a derived class, that specifies the settings for the FileSystem to be created. If NULL on input, a default for this FileSystemCapabilities is used. On output, this returns the actual FileSystemSetting that was used.
ExtentSetting	INOUT, EI	string	EmbeddedInstance("CIM_StorageSetting")  ExtentSetting is an Embedded Instance element of class CIM_StorageSetting, or a derived class, that specifies the settings for the LogicalDisk to be used for building this FileSystem. If NULL on input, a default StorageSetting will be obtained from a StorageConfigurationService hosted on the same ComputerSystem as this FileSystemConfigurationService. On output, this returns the actual StorageSetting that was used. If the output is NULL, the method must return an "Failed".
ExpectedSize	OUT	uint64	An integer that indicates the size of the storage extent that this FileSystem is expected to need. An entry value of 0 indicates that there is no expected size.
MinimumSizeAcceptable	OUT	uint64	An integer that indicates the size of the smallest storage extent that would support the specified FileSystem. A value of 0 indicates that there is no minimum size.
MaximumSizeUsable	OUT	uint64	An integer that indicates the size of the largest storage extent that would be usable for the specified FileSystem. A value of 0 indicates that there is no maximum size.
Normal Return			

**Table 113 - Parameters for Extrinsic Method FileSystemCapabilities.GetRequiredStorageSize (Continued)**

Parameter Name	Qualifier	Type	Description & Notes
Status		uint32	ValueMap{}, Values{} "Success", "Failed", "Timeout"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

**9.3.1.3 FileSystemCapabilities.GetElementNameCapabilities**

This method indicates if ElementName can be specified as a part of invoking an appropriate method of FileSystemConfigurationService to create a new file system. Additionally, the returned data includes the methods that can be used to modify the ElementName of existing file systems.

```
uint32 GetElementNameCapabilities(
    [OUT,
    ValueMap { "2", "3", "4", "..", "32768..65535" },
    Values { "ElementName can be supplied during creation",
    "ElementName can be modified with InvokeMethod",
    "ElementName can be modified with intrinsic method",
    "DMTF Reserved", "Vendor Specific" }]
    uint32 SupportedFeatures[],
    [OUT] string ElementNameMask,
    [OUT] uint16 MaxElementNameLen);
```

The parameters are:

- SupportedFeatures: This OUT parameter is an array indicating what methods can accept the element name for creation or modification of a FileSystem. For example, the value of "ElementName can be supplied during creation" indicates the method such as CreateFileSystem( ) accepts the ElementName when creating a new FileSystem. An empty array indicates ElementNaming for ElementType is not supported.
- MaxElementNameLen: This OUT parameter specifies the maximum supported ElementName length.
- ElementNameMask: This OUT parameter expresses the restrictions on ElementName. The mask is expressed as a regular expression. See DMTF standard ABNF with the Management Profile Specification Usage Guide, Annex C for the regular expression syntax permitted. Since the ElementNameMask can describe the maximum length of the ElementName, any length defined in the regexp is in addition to the restriction defined in MaxElementNameLen (causing the smaller value to be the maximum length). If NULL, it indicates no restrictions on the ElementName.

**9.3.1.4 LocallyAccessibleFileSystemCapabilities.CreateGoalSettings**

This extrinsic method of the LocallyAccessibleFileSystemCapabilities class validates support for a caller-proposed LocallyAccessibleFileSystemSetting passed as the TemplateGoalSettings parameter. This profile restricts the usage of this method to a single entry array for both TemplateGoalSettings and SupportedGoalSettings parameters.

If the input `TemplateGoalSettings` is `NULL` or the empty string, this method returns a single default `LocallyAccessibleFileSystemSetting` in the `SupportedGoalSettings` array. Both `TemplateGoalSettings` and `SupportedGoalSettings` are string arrays containing embedded instances of type `LocallyAccessibleFileSystemSetting`. As such, these settings do not exist in the implementation but are the responsibility of the client.

If the `TemplateGoalSettings` specifies values that cannot be supported, this method shall return an appropriate error and should return a best match for a `SupportedGoalSettings`.

The client and the implementation engage in a negotiation process that may require iterative calls to this method. To assist the implementation in tracking the progress of the negotiation, the client may pass previously returned values of `SupportedGoalSettings` as the new input value of `SupportedGoalSettings`. The implementation may determine that a step has not resulted in progress if the input and output values of `SupportedGoalSettings` are the same. A client may infer from the same result that the `TemplateGoalSettings` must be modified.

#### 9.3.1.4.1 Client Considerations

It is important to understand that the client is acting as an agent for a human user -- either a "system" administrator, or other entity with administrative privileges to the filesystem. During negotiation, the client will show the current state to the user -- the `SupportedGoalSettings` received to date (either the latest or some subset), the `TemplateGoalSettings` proposed (the most recent, but possibly more). But the administrator needs a representation of what is available, possibly the range or sets of values that the different setting properties can take. Some decisions are assumed to have been made already, such as whether the local access is read-only or the file server that is going to access the filesystem.

The `SettingsDefineCapabilities` association from the selected `Capabilities` element provides the information for the client to lay out these options. "Point" settings can be identified supported points in the space of properties -- these points can be further qualified to indicate whether these are supported or not, or whether they represent some ideal point in the space -- a "minimum", or a "maximum", or an "optimal" point. Other settings can provide ranges for properties -- by specifying a minimum, a maximum, and an increment an arithmetic progression of values can be specified (a continuous range can be specified with a zero increment). Specifying a set of supported values for a property that do not follow some pattern is possible, if a bit tedious.

The set of settings associated via `SettingsDefineCapabilities` are expected to be quite stable -- real systems do not continually vary the functionality they can support. Such variations do occur -- for instance, if a new PCMCIA card is added to a running system -- and the best way for a client to be able to add these to the set of choices presented to a user is to subscribe to indications on new `Capabilities` elements and new instances of `SettingsDefineCapabilities`.

There is no guarantee that a negotiation will terminate successfully with the client and the implementation achieving agreement. The implementation may support some simpler mechanisms, short of fully-fledged negotiation, that would be used by a client to obtain an acceptable `TemplateGoalSettings`. The following two use cases are easily covered:

- 1) Client only considers only the canned settings specified exactly. The canned settings are specified by the `LocallyAccessibleFileSystemSetting` elements that are associated to the `LocallyAccessibleFileSystemCapabilities` via `SettingDefinesCapabilities` association with the following property values:
  - `SettingDefinesCapabilities.PropertyPolicy` = "Correlated"
  - `SettingDefinesCapabilities.ValueRole` = "Supported"
  - `SettingDefinesCapabilities.ValueRange` = "Point"
- 2) Client considers canned settings that range over values specified using minimum/increment/maximum for the `Setting` properties. For example, these could be specified by the `LocallyAccessible-`

FileSystemSetting elements that are associated to the LocallyAccessibleFileSystemCapabilities via SettingDefinesCapabilities association with the following property values:

- SettingDefinesCapabilities.ValueRange = "Minimums" or "Maximums" or "Increments"
- The PropertyPolicy and ValueRole properties of SettingDefinesCapabilities will be appropriately specified

Comparing the two CreateGoalSettings extrinsic methods of this profile, the FileSystemCapabilities.CreateGoalSettings() can be significantly more complex than LocallyAccessibleFileSystemCapabilities.CreateGoalSettings(). The client can choose to implement a simpler negotiation protocol for one -- this specification does not mandate the extent to which the client must use this protocol.

#### 9.3.1.4.2 Signature and Parameters of CreateGoalSettings

Table 114 describes the parameters for Extrinsic Method LocallyAccessibleFileSystemCapabilities.CreateGoalSettings.

**Table 114 - Parameters for Extrinsic Method LocallyAccessibleFileSystemCapabilities.CreateGoalSettings**

Parameter Name	Qualifier	Type	Description & Notes
TemplateGoalSettings[]	IN	string	EmbeddedInstance ("CIM_LocallyAccessibleFileSystemSetting")  TemplateGoalSettings is a string array containing embedded instances of class LocallyAccessibleFileSystemSetting, or a derived class. This parameter specifies the client's requirements that is used to locate matching settings that the implementation can support.
SupportedGoalSettings[]	INOUT	string	EmbeddedInstance("CIM_LocallyAccessibleFileSystemSetting" )  SupportedGoalSettings is a string array containing embedded instances of class LocallyAccessibleFileSystemSetting, or a derived class. On input, it specifies a previously returned set of Settings that the implementation could support. On output, it specifies a new set of Settings that the implementation can support. If the output set is identical to the input set, both client and implementation may conclude that this is the best match for the TemplateGoalSettings that is available. If the output does not match the input and the non-NULL output does not match the non-NULL TemplateGoalSettings, then the method must return "\"Alternative Proposed\"". If the output is NULL, the method must return an "Failed".
Normal Return			
Status		uint32	ValueMap{}, Values{}  "Success", "Failed", "Timeout", "Alternative Proposed"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

### 9.3.1.5 FileSystemConfigurationService.CreateFileSystem

This extrinsic method creates a LocalFileSystem and returns it as the value of the parameter TheElement. The desired settings for the LocalFileSystem are specified by the Goal parameter (a string-valued EmbeddedInstance object of class FileSystemSetting).

filesystem vendors differ in their models for creating a filesystem. Some vendors require that the storage element already exist; others create the storage element at the same time as the filesystem. Some vendors require a local access point ("mount-point") that supports defining a name or pathname that allows a file server to access the filesystem; others do not require any such object (though it could be argued that they provide a default local access mechanism). This extrinsic method supports variant mechanisms for specifying, at create time, storage element creation as well as local access by a file server. The FileSystemConfigurationCapabilities associated with the FileSystemConfigurationServices contains the property BlockStorageCreationSupport that specifies support for create-time storage element creation; the property LocalAccessibilitySupport that specifies support for local access by a file server at creation; the property DirectoryServerParameterSupported that specifies support for specifying a file server that provides access to a Directory Service (if enabled separately).

To support backward compatibility with previous releases of SMI-S, an instance of Directory (a derived class of LogicalFile) is created representing the root directory of the newly created filesystem. This Directory element is associated to the LocalFileSystem via FileStorage.

A FileSystemSetting element that represents the settings of the LocalFileSystem (either identical to the Goal or equivalent) shall be associated via ElementSettingData to the LocalFileSystem. The implementation shall create a new FileSystemSetting for this purpose.

The output parameter TheElement shall contain a reference to the newly created LocalFileSystem. Even if this operation does not complete but creates a job, an implementation may return a valid reference in TheElement. If the job fails subsequently, it is possible for this reference to become invalid.

#### 9.3.1.5.1 Specifying Storage Underlying the Filesystem

BlockStorageCreationSupport is an array of enumerated values that specifies if:

- An enumerated value that specifies whether the extrinsic methods may use an already existing LogicalDisk - this is either required, optional, or not allowed. If "Not Allowed", the Pools and ExtentSettings parameters must be used to create LogicalDisk(s) for this LocalFileSystem and the InExtents parameter must be NULL. If "Optional", either the Pools and ExtentSettings parameters or the InExtents parameter should be specified, but not both. If "Required", on the InExtents parameter may be specified and the Pools and ExtentSettings parameters must be NULL.
- (optional) An integer that specifies an upper limit to the number of storage elements that can be specified, either as InExtents parameters or as Pools and ExtentSettings.
- (optional) An integer that specifies the number of distinct storage pools that the Pools parameters can specify -- zero, if Pools is not supported or if there is no limit, and a specific number if there is a limit. In practice it is expected that the value will be either zero or one.
- (optional) A truth value represented as '0' for false and '1' for true that indicates whether an entry in the ExtentSettings array parameter can be NULL (indicating that a default setting is to be used).

The storage used for creating the LocalFileSystem is specified by the InExtents parameter that must be an array of LogicalDisks. If InExtents is NULL and BlockStorageCreationSupport indicates that Pools are optional or required, the parameter Pools must specify an array of StoragePools from which storage may be allocated -- the requirements for the LogicalDisks allocated from this Pool is specified in the ExtentSettings array parameter. The Pools may use an associated StorageConfigurationService. The LocalFileSystem is associated to one of the LogicalDisk(s) via the ResidesOnExtent association. The other LogicalDisks extend the distinguished LogicalDisk (as modeled by the Volume Composition Profile).

### 9.3.1.5.2 Specifying Local Access to the Filesystem

LocalAccessibilitySupport is an enumeration that specifies whether the implementation requires a local access specification, or makes it optional (thus using a vendor default), or does not require one ("local access" does not have a meaning for the vendor).

The LocalFileSystem shall be hosted on the same ComputerSystem as the FileSystemConfigurationService.

NOTE The requirement that the LocalFileSystem have the same host as the Service is too restrictive but this method can be extended in the future with a FileSystemHost parameter (implicitly NULL in 1.2).

If LocalAccess is supported, whether optional or required, this method supports specifying one file server ComputerSystem (the reference parameter FileServer) that is given local access to this filesystem. If LocalAccess is optional, the FileServer parameter may be NULL. The local access name on the FileServer is specified in the LocalAccessPoint string parameter -- if the implementation uses pathnames, this will be formatted as a pathname (directory names separated by the PathNameSeparatorString). The implementation could also use a differently formatted local access name (for instance, a simple name). The settings to be used for this are specified in the LocalAccessSetting, an EmbeddedInstance element of class LocallyAccessibleFileSystemSetting.

NOTE If a second file server ComputerSystem is to be given local access, the ModifyFileSystem method would be used.

Corresponding to the LocalAccessPoint parameter, a LocalAccessAvailableToFS association instance and a LocallyAccessibleFileSystemSettings element are created with the following properties and associations:

- The LocalAccessAvailableToFS association goes between the FileServer parameter and the created LocalFileSystem (TheElement parameter).
- The LocalAccessAvailableToFS.LocalAccessPoints property is set to the value of the LocalAccessPoint string.
- The LocallyAccessibleFileSystemSetting element has an ElementSettingData association to the LocalFileSystem (TheElement).
- The LocallyAccessibleFileSystemSetting element has a HostedDependency (ScopedSetting) association to the FileServer parameter.

If LocalAccess is supported, the FileServer parameter should not be NULL.

NOTE If it is NULL, the implementation may leave the filesystem operationally inaccessible -- however, this can be corrected by calling the ModifyFileSystem method. This is not an Error.

If LocalAccess is supported and the FileServer parameter is not NULL, the LocalAccessPoint string may be NULL or the empty string. In this case, the LocalAccessSetting parameter should indicate the implementation-specific default format. The default value that is used is returned as the OUT value of the LocalAccessPoint parameter. It is an Error if the LocalAccessSetting parameter does not provide an appropriate default mechanism for constructing a local access name.

The LocalAccessSetting parameter will return an EmbeddedInstance of the LocallyAccessibleFileSystemSetting actually used on output.

### 9.3.1.5.3 Specifying access to Directory Services

DirectoryServerParameterSupported is a property that specifies whether the filesystem must have access to a file server that provides access to directory services so that security principal information may be supported. If the newly created filesystem must be able to resolve such information, the DirectoryServer parameter must be specified to the CreateFileSystem method.

The DirectoryServer parameter specifies a file server ComputerSystem that is configured to access a directory service that resolves security principal identifies (UIDs, GIDs, or SIDs) used by the filesystem.



This profile does not specify the configuration of any directory service (if there is one), any directory server, or the file server that is specified by the DirectoryServer parameter. For operational efficiency reasons, this must be a file server since security principal information such as usage and detection of threshold violations must happen in real-time.

If the DirectoryServer is required and is specified, an association, a concrete subclass of Dependency, shall be surfaced between the newly created LocalFileSystem element (as Dependent) and the specified file server (as Antecedent). The CreateFileSystem method will return a reference to this file server as the return value of the parameter. In this case, the LocalFileSystem.DirectoryServiceUsage property shall be set to "Supported".

Any file server that is configured with write access to a filesystem must use the same or a compatible directory service (effectively the same) as the file server indicated by the Dependency association.

### 9.3.2 Signature and Parameters of CreateFileSystem

Table 115 describes the parameters for Extrinsic Method FileSystemConfigurationService.CreateFileSystem.

**Table 115 - Parameters for Extrinsic Method FileSystemConfigurationService.CreateFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
ElementName	IN	string	An end user relevant name for the FileSystem being created. The value shall be stored in the 'ElementName' property for the created element. This parameter shall not be NULL or the empty string.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
Goal	IN, OUT, EI	string	EmbeddedInstance ("CIM_FileSystemSetting")  The FileSystemSetting requirements for the FileSystem. If NULL or the empty string, a default FileSystemSetting shall be specified by the FileSystemCapabilities element associated to the FileSystemConfigurationService by the DefaultElementCapabilities association. The actual FileSystemSetting used is returned on output.
TheElement	OUT, REF	CIM_LocalFileSystem	The newly created FileSystem.
InExtents[]	IN, OUT, REF, NULL allowed,	CIM_LogicalDisk	The LogicalDisk(s) on which the created FileSystem shall reside. If this is NULL, the Pools and ExtentSettings parameters cannot be NULL and are used to create LogicalDisk(s). The LogicalDisk(s) actually used will be returned on output.
Pools[]	IN, REF, NULL allowed	CIM_StoragePool	An array of concrete StoragePool elements corresponding to the ExtentSettings parameter from which to create LogicalDisks in case the InExtents parameter is NULL. If InExtents is not NULL, this must be NULL.
ExtentSettings[]	IN, EI, NULL Allowed	string	EmbeddedInstance ("CIM_StorageSetting")  An array of embedded StorageSetting structures that specify the settings to use for creating LogicalDisks if the InExtents parameter is NULL and Pools is specified. Each LogicalDisk will be created from the corresponding entry in Pools, so each StorageSetting entry must be supported by the capabilities of the corresponding Pools entry.

**Table 115 - Parameters for Extrinsic Method FileSystemConfigurationService.CreateFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
Sizes[]	IN, OUT, NULL Allowed	uint64	An array of numbers that specifies the size in bytes of the LogicalDisks to be created corresponding to the Pools and ExtentSettings parameters. The sum of Sizes should be at least as much as (or greater than) the FileSystem size needed.
FileServer	IN, OUT, REF, NULL Allowed	ComputerSystem	<p>A reference to a ComputerSystem element that will access the created LocalFileSystem and is capable of exporting the filesystem as a file share. The local access point with respect to the file server is specified by the LocalAccessPoint parameter. If FileSystemConfigurationCapabilities.LocalAccessibilitySupport specifies that local access points are supported but implementation-defaulted, the corresponding entry in the LocalAccessPoint parameter should be NULL or the empty string as the LocalAccessPoint name is constructed as per the vendor default algorithm. A LocalAccessAvailableToFS association is created between the FileServer and the LocalFileSystem. The parameters for local access are specified by the LocalAccessSetting parameter.</p> <p>Since this filesystem has just been created, the LocalAccessSetting can support Write privileges. If the LocalAccessSetting entry is NULL or the empty string, the implementation uses a default associated with the LocallyAccessibleFileSystemCapabilities associated to the FileServer.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilitySupport specifies that a local access point is required and FileServer is NULL, no LocalAccessAvailableToFS associations are created and the filesystem may not be accessible. This shall not cause an Error.</p> <p>On output, this parameter contains a reference to the actual FileServer that has access to the created LocalFileSystem.</p>

**Table 115 - Parameters for Extrinsic Method FileSystemConfigurationService.CreateFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
LocalAccessPoint	IN, OUT, REF, NULL Allowed	string	<p>A string to use as a pathname in the name space of the file server ComputerSystem. The format of the string is vendor-dependent and it should be considered opaque from the client's standpoint. It could be interpreted as a hierarchical fully-qualified name for the local access point (say in a Unix-based operating environment), or it could be a drive letter (as in a Windows operating environment). A LocalAccessAvailableToFS association is created going between the new LocalFileSystem and the FileServer parameter. The LocalAccessAvailableToFS.LocalAccessPoint property will be set to this parameter.</p> <p>The parameters for local access are specified by the LocalAccessSetting parameter.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilityS support specifies that local access points are required, then LocalAccessPoint shall not be NULL or an empty string.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilityS support specifies that local access points can be vendor-defaulted, then LocalAccessPoint can be NULL or an empty string and the implementation shall create a name using a vendor-specific algorithm.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilityS support specifies that local access points cannot be vendor-defaulted, then LocalAccessPoint shall not be NULL and the implementation shall not create a default pathname. This is an Error.</p> <p>On output, this parameter contains the actual LocalAccessPoint used (including any name created by vendor-default).</p>
LocalAccessSetting	IN, EI, OUT, NULL Allowed	string	<p>EmbeddedInstance ("CIM_LocallyAccessibleFileSystemSetting")</p> <p>An embedded LocallyAccessibleFileSystemSetting element that specifies the settings to use to establish a local access point. This element will be used to create a LocalAccessAvailableToFS association and will be cloned to create a LocallyAccessibleFileSystemSetting element that is scoped via HostedDependency (ScopedSetting) to the FileServer and associated via ElementSettingData to the LocalFileSystem.</p> <p>If a LocallyAccessibleFileSystemSetting entry is NULL or the empty string, the implementation shall use the default provided by the LocallyAccessibleFileSystemCapabilities element of the FileSystemConfigurationService that is associated to the FileServer via CIM_Dependency. The LocalAccessSetting may specify a Write Privilege.</p> <p>The LocalAccessSetting actually used is returned as the OUT EmbeddedInstance parameter.</p>

**Table 115 - Parameters for Extrinsic Method FileSystemConfigurationService.CreateFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
DirectoryServer	IN, OUT, NULL Allowed	ComputerSystem	<p>A reference to a ComputerSystem element that has access to directory services. The newly created filesystem can use it to support security principal information associated with filesystem objects, such as quotas for users and groups. This is represented by providing a Dependency association between the LocalFileSystem element and the ComputerSystem indicated by this parameter. The requirements for this parameter are further specified by FileSystemConfigurationCapabilities.DirectoryServerParameterSupported.</p> <p>If DirectoryServerParameterSupported specifies 'Not Used', or 'Supported, Defaulted to FileServer', or 'Supported, Defaulted to FileSystem host', it is an Error if DirectoryServer is not NULL.</p> <p>Otherwise, (i.e., if DirectoryServerParameterSupported specifies 'Supported'), and if the DirectoryServer is not NULL, the new filesystem will use the directory services made available by the specified DirectoryServer. If DirectoryServer is NULL, it will be defaulted to the FileServer parameter. If the FileServer parameter is also NULL, the DirectoryServer will be defaulted to the host of the newly created filesystem.</p> <p>On output, this parameter contains a reference to the actual DirectoryServer if one was established.</p>
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

### 9.3.2.1 FileSystemConfigurationService.ModifyFileSystem

This extrinsic method modifies a LocalFileSystem specified by the parameter TheElement. The desired settings for the LocalFileSystem are specified by the Goal parameter (a string-valued EmbeddedInstance object of class FileSystemSetting).

As with CreateFileSystem, the FileSystemConfigurationCapabilities associated with the FileSystemConfigurationService specifies support for some of the optional behaviors of this method. BlockStorageCreationSupport indicates whether this method only uses previously existing storage elements or if it can create them at the same time as modifying or creating the filesystem. In addition this can specify if additional LogicalDisks can be added to the existing set of LogicalDisks and whether the implementation limits the number of LogicalDisks underlying a filesystem. LocalAccessibilitySupport indicates whether the implementation requires support for local access points (or if they are optional or not required at all).

This element shall have a `FileSystemSetting` use `ActualFileSystemType` property is supported by the `FileSystemConfigurationService` (as specified by the `SupportedActualFileSystemTypes` property of the associated `FileSystemConfigurationCapabilities`). The existing `LogicalDisks` used by the `LocalFileSystem` cannot be released by this method, but this method may add `LogicalDisks`. These `LogicalDisks` may be specified by the `InExtents` parameter (if that is either required or optional) or, if `InExtents` is `NULL` (if `Pools` are optional or required), the set of `LogicalDisks` is not changed. New `LogicalDisks` may also be added by specifying an array of `StoragePools` in the `Pools` parameter and an array of `StorageSettings` that can be used to create them.

If the operation would result in a change in the size of a filesystem, the `ResidesOnExtent` association shall be used to determine how to implement the change. If the existing or additional `LogicalDisk(s)` specified, or any additional `LogicalDisks` created, cannot support the goal size, an appropriate error value shall be returned, and no action shall be taken. If the operation succeeds, the `ResidesOnExtent` association shall reference the same `LogicalDisk` as before (however, the `LogicalDisk` will be built upon a larger number of underlying `LogicalDisks`, as modeled by the `Volume Composition Profile`).

If the new `Goal` is different from the old `FileSystemSetting` element associated to the `LocalFileSystem` element, then the implementation must change the setting properties of the `LocalFileSystem`. This may be accomplished by modifying the old `FileSystemSetting` element directly, or by deleting it and then re-creating a new `FileSystemSetting` element with the same `InstanceId`. Just like the old element, the new `FileSystemSetting` element shall be associated to the `LocalFileSystem` element via an `ElementSettingData` association.

If local access is supported, whether optional or required, any change is specified by providing the `FileServer` parameter (which shall be a reference to a `ComputerSystem`). If a `FileServer` is not being added to the set or modified or removed from the set, the `FileServer` parameter may be `NULL`.

If the specified `FileServer` does not duplicate a `ComputerSystem` that has already been specified as having local access, this method adds it to the set. The pathname is specified by the `LocalAccessPoint` string array parameter. The settings to be used for these are specified in the `LocalAccessSetting`, an `EmbeddedInstance` element of class `LocallyAccessibleFileSystemSetting`.

If the specified `FileServer` duplicates a `ComputerSystem` that has already been specified as having local access, this method either modifies the local access or removes it from the set. If the `LocalAccessPoint` parameter is `NULL` or consists of an empty string, this call removes the `FileServer` from the set. If the `LocalAccessPoint` parameter is not `NULL` but specifies the current path, then this call modifies the settings of the local access -- the new settings are specified by the `LocalAccessSetting` parameter. If the `LocalAccessPoint` parameter is not `NULL` but specifies a path other than the current path, then this call modifies the pathname as well as the settings. If this filesystem is in operational use when such a request is made, the request may have to be suspended until the filesystem can be put into an appropriate state for making the change.

The settings to be used for adding or modifying are specified by the `LocalAccessSetting` parameter, an `EmbeddedInstance` element of class `LocallyAccessibleFileSystemSetting`.

To add a local access point, a `LocalAccessAvailableToFS` association and a `LocallyAccessibleFileSystemSettings` element are created with the following properties and associations:

- A `LocalAccessAvailableToFS` association goes between the `FileServer` parameter and the `LocalFileSystem` (`TheElement` parameter).
- A `LocalAccessAvailableToFS.LocalAccessPoint` property is set to the `LocalAccessPoint` string.
- A `LocallyAccessibleFileSystemSetting` element with a `ElementSettingData` association to the `LocalFileSystem` (`TheElement` parameter).
- The `LocallyAccessibleFileSystemSetting` element has a `HostedDependency` (`ScopedSetting`) association to the `FileServer` parameter.

NOTE The WaitTime and InUseOptions parameters are supported if the FileSystemCapabilities.SupportedProperties includes the "RequireInUseOptions" option.

NOTE A client can identify all local access specifications for a filesystem by looking for the LocalAccessAvailableToFS association from the LocalFileSystem to a file server ComputerSystem and the LocallyAccessibleFileSystemSetting associated to the LocalFileSystem via ElementSettingData and the same file server ComputerSystem via HostedDependency (ScopedSetting).

### 9.3.3 Signature and Parameters of ModifyFileSystem

Table 116 describes the parameters for Extrinsic Method FileSystemConfigurationService.ModifyFileSystem.

**Table 116 - Parameters for Extrinsic Method FileSystemConfigurationService.ModifyFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
ElementName	IN, OUT	string	An end user relevant name for the filesystem being modified. If NULL, the existing TheElement.ElementName property is not changed. If not NULL, this parameter will supply a new name for the Element parameter. The actual ElementName is returned as the output value.
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
Goal	IN, OUT, EI	string	EmbeddedInstance ("CIM_FileSystemSetting")  The FileSystemSetting requirements for the filesystem specified by the TheElement parameter. If NULL or the empty string, the settings for TheElement will not be changed. If not NULL, this parameter will supply new settings that replace or are merged with the current settings of TheElement.
TheElement	IN, REF	CIM_LocalFileSystem	The LocalFileSystem element to modify.
InExtents[]	IN, OUT, REF, NULL allowed,	CIM_LogicalDisk	The LogicalDisk(s) used to extend the current set of LogicalDisks used for the TheElement filesystem. If this is not NULL, the Pool and ExtentSettings must be NULL. If both this and Pool are NULL, the current set will not be changed. The current set of LogicalDisk(s) will be returned as the output.
Pools[]	IN, REF, NULL allowed	CIM_StoragePool	An array of concrete storage pools corresponding to the ExtentSettings array parameter. These storage pools are used to create additional LogicalDisks to extend the TheElement filesystem. The InExtents parameter must be NULL and the ExtentSettings parameter must not be NULL. Otherwise, the current set of LogicalDisks is not changed.
ExtentSettings[]	IN, EI, NULL Allowed	string	EmbeddedInstance ("CIM_StorageSetting")  An array of embedded StorageSetting structures that specify the settings to use for creating additional LogicalDisks for the TheElement filesystem. The InExtents parameter must be NULL and Pools must be specified. Each LogicalDisk will be created from the corresponding Pool, so each StorageSetting entry must be supported by the capabilities of the corresponding Pool entry.
Sizes[]	IN, NULL Allowed	uint64	An array of numbers that specifies the size in bytes of the LogicalDisks to be created corresponding to the ExtentSettings array parameter.

**Table 116 - Parameters for Extrinsic Method FileSystemConfigurationService.ModifyFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
FileServer	IN, OUT, REF, NULL Allowed	REF Computer System	<p>A reference to a ComputerSystem element representing a file server.</p> <p>If this parameter is NULL, no change is made to the local access configuration. If it is not NULL, the change to the configuration consists of the following cases:</p> <p>1.) If the FileServer does not already support local access to the TheElement, it will be added and made capable of exporting the filesystem as file shares. The local access point is specified by the LocalAccessPoint parameter.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilitySupport specifies that local access points are vendor-defaulted, the corresponding entry in the LocalAccessPoints parameter should be NULL or the empty string as the LocalAccessPoint name is constructed by a vendor-default algorithm.</p> <p>A LocalAccessAvailableToFS association is created between the FileServer and the TheElement. The parameters for local access are specified by the LocalAccessSetting parameter, an EmbeddedInstance element of class LocallyAccessibleFileSystemSetting.</p> <p>If the LocalAccessSetting parameter is NULL or the empty string, the implementation uses a default associated with the LocallyAccessibleFileSystemCapabilities of the FileSystemConfigurationService associated to the FileServer by HostedDependency (ScopedSetting). At most one of the LocalAccessSettings entries for all the file server ComputerSystems that provide local access to this filesystem shall specify an element with Write Privileges.</p> <p>2) If FileServer already supports local access to the TheElement, and the LocalAccessPoint parameter is NULL or a set of empty strings, this will remove the FileServer from the configured set. If there are existing operational users of the TheElement filesystem, they will need to be informed and the implementation might have to wait to reach a consistent state before the request can be completed.</p> <p>3) If FileServer already supports local access to the TheElement, and the LocalAccessPoint parameter is the same as the current configuration, then this is a request to change the settings but not the local access point. The LocalAccessSetting parameter will specify the new setting. Depending on the precise change, the filesystem may need to suspend operations. If there are existing operational users of the filesystem, they will need to be informed and the implementation might have to wait to reach a consistent state before the request can be completed.</p> <p>4) If FileServer already supports local access to the TheElement, and the LocalAccessPoint parameter is different from the current configuration, then this is equivalent to removing local access and then restoring it with different settings. If there are existing operational users of the filesystem, they will need to be informed and the implementation might have to wait to reach a consistent state before the request can be completed. Note that existing operational users will not be able to reconnect as the share name may have changed.</p>

**Table 116 - Parameters for Extrinsic Method FileSystemConfigurationService.ModifyFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
LocalAccessPoint	IN, OUT, REF, NULL Allowed	string	<p>A string to use as a pathname in the name space of the file server ComputerSystem specified by the FileServer parameter. The format of the string is vendor-dependent and it should be considered opaque to the client. It could be interpreted as a hierarchical fully-qualified name for the local access point (say in a Unix-based operating environment), or it could be a drive letter (as in a Windows operating environment). A LocalAccessAvailableToFS association is created going between the TheElement and the FileServer. The LocalAccessAvailableToFS.LocalAccessPoint property will be set to the value of this parameter.</p> <p>The parameters for local access are specified by the LocalAccessSetting parameter.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilitySupport specifies that local access points are required, then LocalAccessPoint shall not be NULL or an empty string if this is a new FileServer that does not have local access to TheElement. This is an Error.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilitySupport specifies that local access points can be vendor-defaulted, then LocalAccessPoint can be NULL or an empty string and the implementation shall create a name using a vendor-specific algorithm.</p> <p>If FileSystemConfigurationCapabilities.LocalAccessibilitySupport specifies that local access points cannot be vendor-defaulted, and this is a new FileServer that does not have local access to TheElement, then LocalAccessPoint shall not be NULL and the implementation shall not create a default pathname. This is an Error.</p> <p>On output, this parameter contains the actual LocalAccessPoint used (including any name created by vendor-default).</p>
LocalAccessSetting	IN, EI, OUT, NULL Allowed	string	<p>EmbeddedInstance ("CIM_LocallyAccessibleFileSystemSetting")</p> <p>An embedded LocallyAccessibleFileSystemSetting element that specifies the settings to use for establishing a local access point. Each entry will be used to create or modify a LocalAccessAvailableToFS association and will be cloned to create a LocallyAccessibleFileSystemSetting element that is scoped via ScopedSetting (or HostedDependency) to the file server ComputerSystem specified by the FileServer parameter. The clone will be associated via ElementSettingData to the LocalFileSystem.</p> <p>If this parameter is NULL or the empty string, and a new value is needed, the implementation shall use the default provided by the LocallyAccessibleFileSystemCapabilities associated to the FileServer parameter. The LocalAccessSetting actually used is returned as the OUT parameter.</p>
InUseOptions	IN	uint16	An enumerated integer that specifies the action to take if the filesystem is still in operational use when this request is made. This option is only relevant if the FileSystem needs to be made unavailable while the request is being executed.
WaitTime	IN	uint32	An integer that indicates the time in seconds to wait before performing the request on this filesystem. The combination of InUseOptions = '4' and WaitTime = '0' (the default) is interpreted as 'Wait (forever) until Quiescence, then Execute Request.



**Table 116 - Parameters for Extrinsic Method FileSystemConfigurationService.ModifyFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

**9.3.3.1 FileSystemConfigurationService.DeleteFileSystem**

This is an extrinsic method that shall delete a LocalFileSystem specified by the parameter TheElement and delete any associated elements and associations that are no longer needed. The deleted elements include the LogicalFile/Directory and FileStorage, if they were surfaced; the LocalAccessAvailableToFS association, the LocallyAccessibleFileSystemSetting element and its associations, ElementSettingData, HostedDependency (ScopedSetting); HostedFileSystem, ResidesOnExtent, and any associations that might be orphaned by the deletion of TheElement. An implementation is not required to delete or re-allocate the LogicalDisk(s) that TheElement used.

NOTE The WaitTime and InUseOptions parameters are supported if the FileSystemCapabilities.SupportedProperties includes the "RequireInUseOptions" option.

**9.3.4 Signature and Parameters of DeleteFileSystem**

Table 117 describes the parameters for Extrinsic Method FileSystemConfigurationService.DeleteFileSystem.

**Table 117 - Parameters for Extrinsic Method FileSystemConfigurationService.DeleteFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
Job	OUT, REF	CIM_ConcreteJob	Reference to the job (may be null if job completed).
TheElement	IN, REF	CIM_LocalFileSystem	The filesystem element to delete.
InUseOptions	IN	uint16	An enumerated integer that specifies the action to take if TheElement is still in use when this request is made. This option is only relevant if the filesystem needs to be made unavailable while the request is being executed.
WaitTime	IN	uint32	An integer that indicates the time in seconds to wait before performing the request on TheElement filesystem. The combination of InUseOptions = '4' and WaitTime = '0' (the default) is interpreted as 'Wait (forever) until Quiescence, then Execute Request.
Normal Return			
Status		uint32	"Job Completed with No Error", "Failed", "Method Parameters Checked - Job Started"

**Table 117 - Parameters for Extrinsic Method FileSystemConfigurationService.DeleteFileSystem**

Parameter Name	Qualifier	Type	Description & Notes
Error Returns			
Invalid Property Value	OUT, Indication	CIM_Error	A single named property of an instance parameter (either reference or embedded) has an invalid value
Invalid Combination of Values	OUT, Indication	CIM_Error	An invalid combination of named properties of an instance parameter (either reference or embedded) has been requested.

### 9.3.5 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

## 9.4 Use Cases

### 9.4.1 Filesystem Manipulation Supported Capabilities Patterns

Table 118, “Filesystem Manipulation Supported Capabilities Patterns” lists the patterns that are formally recognized by this version of the specification for determining capabilities of various NAS implementations:

**Table 118 - Filesystem Manipulation Supported Capabilities Patterns**

Supported ActualFileSystem Types	Supported Synchronous Methods	Supported Asynchronous Methods	Initial Availability
Any	none	none	none
Any	CreateFileSystem, DeleteFileSystem, ModifyFileSystem, CreateGoalSettings, GetRequiredStorageSizes	none	Any
Any	CreateGoalSettings, GetRequiredStorageSizes	CreateFileSystem, DeleteFileSystem, ModifyFileSystem	Any

## 9.5 CIM Elements

Table 119 describes the CIM elements for Filesystem Manipulation.

**Table 119 - CIM Elements for Filesystem Manipulation**

Element Name	Requirement	Description
9.5.1 CIM_Dependency (Uses Directory Services From)	Conditional	Conditional requirement: Required if LocalFileSystem.DirectoryServiceUsage is either 'Required' or 'Optional'. Associates a ComputerSystem that indicates a directory service that supports the dependent LocalFileSystem.
9.5.2 CIM_ElementCapabilities (FS Configuration Capabilities)	Mandatory	In this profile, associates the Filesystem Configuration Service to the Capabilities element that represents the capabilities that it supports.
9.5.3 CIM_ElementCapabilities (Local Access Configuration Capabilities)	Conditional	<p>Conditional requirement: Required if FileSystemConfigurationCapabilities.LocalAccessSupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'.</p> <p>In this profile, associates the Filesystem Configuration Service to the Capabilities instance that represents the capabilities for Local Access that it supports.</p>
9.5.4 CIM_ElementCapabilities (Default)	Optional	This entry represents the single default FileSystemCapabilities element for the Filesystem Configuration Service.
9.5.5 CIM_ElementCapabilities (Non-Default)	Optional	In this profile, associates the Filesystem Configuration Service to the FileSystemCapabilities elements that represent all the types of filesystems that are not the default type of file system and can be configured.
9.5.6 CIM_ElementSettingData (Attached to Filesystem)	Optional	<p>Associates a FileSystemSetting element to a LocalFileSystem. One of these association elements is created by CreateFileSystem when the LocalFileSystem is first created.</p> <p>The profile does not specify how other instances of this association may be surfaced by the implementation.</p>
9.5.7 CIM_ElementSettingData (Local Access Required)	Conditional	Conditional requirement: Required if FileSystemConfigurationCapabilities.LocalAccessSupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'. Associates a LocalFileSystem and the LocallyAccessibleFileSystemSetting elements.

**Table 119 - CIM Elements for Filesystem Manipulation**

Element Name	Requirement	Description
9.5.8 CIM_FileSystemCapabilities	Mandatory	This element represents the Capabilities of the Filesystem Configuration Service for managing Filesystems. The Service can be associated with multiple FileSystemCapabilities elements, one per ActualFileSystemType property value. For each value that is in the array property FileSystemConfigurationCapabilities.SupportedActualFileSystemTypes, there will be exactly one corresponding FileSystemCapabilities element with the matching ActualFileSystemType property.
9.5.9 CIM_FileSystemConfigurationCapabilities	Mandatory	This element represents the management Capabilities of the Filesystem Configuration Service.
9.5.10 CIM_FileSystemConfigurationService	Mandatory	The Filesystem Configuration Service provides the methods to manipulate file systems.
9.5.11 CIM_FileSystemSetting (Attached to FileSystem)	Optional	<p>This element represents the configuration settings of a LocalFileSystem. One instance of this class is created by the CreateFileSystem extrinsic method when the LocalFileSystem was created.</p> <p>This profile does not specify how other instances of this class might be created.</p>
9.5.12 CIM_FileSystemSetting (Predefined FS Settings)	Optional	This element represents sample configuration settings usable for creating or modifying a LocalFileSystem. It represents "predefined" settings supported by the FileSystemConfigurationService and is associated with a FileSystemCapabilities element by a SettingsDefineCapabilities association. The FileSystemSetting.ActualFileSystemType property must specify the same value as the associated FileSystemCapabilities.ActualFileSystemType property.
9.5.13 CIM_HostedDependency (Attached to File System)	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. Associates a Local Access configuration setting to the file server ComputerSystem that provides the operational scope for its functionality.

**Table 119 - CIM Elements for Filesystem Manipulation**

Element Name	Requirement	Description
9.5.14 CIM_HostedDependency (Predefined Capabilities)	Conditional	Conditional requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'. Associates a Local Access Capabilities to the File Server that provides the operational scope for its functionality. All of the Settings associated to the referenced Capabilities element must be scoped by the same File Server ComputerSystem. This scoping allows the CreateGoalSetting method of the Capabilities element to know which File Server provides the scope for any Goal element that it creates.
9.5.15 CIM_HostedDependency (Predefined Setting)	Optional	Associates a predefined LocallyAccessibleFileSystemService to the file server ComputerSystem that provides the operational scope for its functionality.
9.5.16 CIM_HostedFileSystem	Mandatory	Associates a LocalFileSystem to the ComputerSystem that hosts it.
9.5.17 CIM_HostedService	Mandatory	In this profile, associates the Filesystem Configuration Service to the hosting ComputerSystem. This is expected to be the top-level ComputerSystem of the parent Filesystem Profile.
9.5.18 CIM_LocalAccessAvailableToFS	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. Associates a LocalFileSystem to a File Server Computer System that can export files or directories as shares.
9.5.19 CIM_LocalFileSystem	Mandatory	Represents a LocalFileSystem hosted by and made available through a ComputerSystem (usually the top-level ComputerSystem of a Filesystem Profile).
9.5.20 CIM_LocallyAccessibleFileSystemCapabilities	Conditional	Conditional requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'. The element represents the Local Access configuration Capabilities of the File System Configuration Service. This class provides a CreateGoalSettings method that will return a LocallyAccessibleFileSystemService element as an EmbeddedInstance that may be used for making a filesystem locally accessible to a file server ComputerSystem (by the methods CreateFileSystem and ModifyFileSystem). Since the returned EmbeddedInstance setting element is an instance of a ScopedSetting class, it must be associated with a ComputerSystem via ScopedSettingData when it is instantiated.

**Table 119 - CIM Elements for Filesystem Manipulation**

Element Name	Requirement	Description
9.5.21 CIM_LocallyAccessibleFileSystemSetting	Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. This element represents the configuration settings of a LocalFileSystem that has a contained file or directory that has been made locally accessible from a file server ComputerSystem. This Setting provides further details on the functionality supported and the parameters of that functionality when locally accessible.
9.5.22 CIM_SettingsDefineCapabilities (Predefined FS Settings)	Optional	These Setting elements provide detailed information about the FileSystemSettings supported by the associated FileSystemCapabilities element.
9.5.23 CIM_SettingsDefineCapabilities (Predefined Local Access Settings)	Conditional	Conditional requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'. The Setting elements that are associated to this Capabilities element are scoped to the File Server ComputerSystem that provides the operational context for local access.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LocalFileSystem	Mandatory	Creation of a LocalFileSystem element.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LocalFileSystem	Mandatory	Deletion of a LocalFileSystem element
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LocalFileSystem	Mandatory	Modification of a LocalFileSystem element.

**9.5.1 CIM\_Dependency (Uses Directory Services From)**

Created By: Extrinsic: CreateFileSystem

Modified By: Extrinsic: ModifyFileSystem

Deleted By: Extrinsic: DeleteFileSystem

Requirement: Required if LocalFileSystem.DirectoryServiceUsage is either 'Required' or 'Optional'.

Table 120 describes class CIM\_Dependency (Uses Directory Services From).

**Table 120 - SMI Referenced Properties/Methods for CIM\_Dependency (Uses Directory Services From)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The ComputerSystem that indicates the directory service(s) that support user, group and other security principal identities for a filesystem.
Dependent		Mandatory	The LocalFileSystem whose use of user, group, and other security principal identities is supported by the antecedent ComputerSystem.

**9.5.2 CIM\_ElementCapabilities (FS Configuration Capabilities)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 121 describes class CIM\_ElementCapabilities (FS Configuration Capabilities).

**Table 121 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (FS Configuration Capabilities)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The Filesystem Configuration Service.
Capabilities		Mandatory	The Filesystem Configuration Capabilities element.

### 9.5.3 CIM\_ElementCapabilities (Local Access Configuration Capabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'.

Table 122 describes class CIM\_ElementCapabilities (Local Access Configuration Capabilities).

**Table 122 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (Local Access Configuration Capabilities)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The Filesystem Configuration Service.
Capabilities		Mandatory	The Filesystem Configuration Capabilities element.

### 9.5.4 CIM\_ElementCapabilities (Default)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 123 describes class CIM\_ElementCapabilities (Default).

**Table 123 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (Default)**

Properties	Flags	Requirement	Description & Notes
Characteristics		Optional	
Capabilities		Mandatory	
ManagedElement		Mandatory	

### 9.5.5 CIM\_ElementCapabilities (Non-Default)

Created By: Static

Modified By: Static

Deleted By: Static  
Requirement: Optional

Table 124 describes class CIM\_ElementCapabilities (Non-Default).

**Table 124 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (Non-Default)**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

#### 9.5.6 CIM\_ElementSettingData (Attached to Filesystem)

Created By: Extrinsic: CreateFileSystem  
Modified By: Extrinsic: ModifyFileSystem  
Deleted By: Extrinsic: DeleteFileSystem  
Requirement: Optional

Table 125 describes class CIM\_ElementSettingData (Attached to Filesystem).

**Table 125 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (Attached to Filesystem)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The LocalFileSystem element representing a filesystem.
SettingData		Mandatory	The configuration of the LocalFileSystem.

#### 9.5.7 CIM\_ElementSettingData (Local Access Required)

Created By: Extrinsic: CreateFileSystem  
Modified By: Extrinsic: ModifyFileSystem  
Deleted By: Extrinsic: DeleteFileSystem  
Requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'.

Table 126 describes class CIM\_ElementSettingData (Local Access Required).

**Table 126 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (Local Access Required)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The LocalFileSystem that is being made locally accessible.
SettingData		Mandatory	The local access settings of the LocalFileSystem, specified on creation or modification.

#### 9.5.8 CIM\_FileSystemCapabilities

Created By: Static  
Modified By: Static  
Deleted By: Static  
Requirement: Mandatory



Table 127 describes class CIM\_FileSystemCapabilities.

**Table 127 - SMI Referenced Properties/Methods for CIM\_FileSystemCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the FileSystemCapabilities element of a Filesystem Configuration Service.
ElementName		Mandatory	A user-friendly name for this Capabilities element.
ActualFileSystemType		Mandatory	This identifies the type of filesystem that this FileSystemCapabilities represents.
SupportedProperties		Mandatory	This is the list of configuration properties (of FileSystemSetting) that are supported for specification at creation time by this FileSystemCapabilities element.
CreateGoalSettings()		Mandatory	This extrinsic method supports the creation of a set of FileSystemSettings that is a supported variant of an array of FileSystemSettings passed in as an embedded IN parameter. The method returns the supported FileSystemSetting in an array of embedded OUT parameters. This profile only supports arrays with a single entry.
GetRequiredStorageSize()		Optional	This extrinsic method supports determining the storage space requirements for a filesystem specified by the combination of a FileSystemSetting and a StorageSetting. The StorageSetting specifies the required redundancy, multiple Logical Disk usage, and other storage mapping considerations, while the FileSystemSetting transforms client quality-of-service specifications to storage resource requirements.

### 9.5.9 CIM\_FileSystemConfigurationCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 128 describes class CIM\_FileSystemConfigurationCapabilities.

**Table 128 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for this element representing the capabilities of a Filesystem Configuration Service.
ElementName		Mandatory	A user-friendly name for this Capabilities element.
SupportedActualFileSystemTypes		Mandatory	The Service can be associated with multiple Capabilities elements, one per ActualFileSystemType property value. This property lists all of the supported ActualFileSystemTypes. Each entry in this array must have exactly one corresponding FileSystemCapabilities element with that entry as the value of the ActualFileSystemType property.
SupportedSynchronousMethods	N	Mandatory	The Service supports a number of extrinsic methods -- this property identifies the ones that can be called synchronously. A supported method shall be listed in this property or in the SupportedAsynchronousMethods property or both.

**Table 128 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousMethods	N	Mandatory	The Service supports a number of extrinsic methods -- this property identifies the ones that can be called asynchronously. A supported method shall be listed in this property or in the SupportedSynchronousMethods property or both.
InitialAvailability		Mandatory	This property represents the state of availability of a LocalFileSystem on initial creation using the FileSystemConfigurationService associated with this Capabilities element.
LocalAccessibilitySupport		Optional	This specifies whether a LocalFileSystem created or modified by this FileSystemConfigurationService needs to be made locally accessible at a local access point before a file server ComputerSystem can make it available to operational clients or for export as a share. This is typical of some NAS and filesystem implementations. If not specified, the default is "Local Access Not Required".

**Table 128 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
BlockStorageCreationSupport		Optional	<p>BlockStorageCreationSupport is an ordered array of enumerated values that place a number of restrictions on the use of parameters for CreateFileSystem and ModifyFileSystem.</p> <ol style="list-style-type: none"> <li>1. The first entry is an enumerated value that specifies if an already existing LogicalDisk may be used -- this is either required, optional, or not allowed. "Not Allowed" indicates that the Pools and ExtentSettings parameters must be used to create LogicalDisk(s) for this filesystem and the InExtents parameter must be NULL. "Optional" indicates that either the Pools and ExtentSettings parameters or the InExtents parameter should be specified, but not both. "Required" indicates that the InExtents parameter may be specified and the Pools and ExtentSettings parameters must be NULL.</li> <li>2. (optional) An integer that specifies an upper limit to the number of StorageElements that can be specified, either as InExtents parameters or as Pools and ExtentSettings.</li> <li>3. (optional) An integer that specifies the number of distinct pools that the Pools parameters can specify -- zero, if Pools is not supported or if there is no limit, and a specific number if there is a limit. In practice we expect that the value will be either zero or one.</li> <li>4. (optional) A boolean value, represented by '0' for false and '1' for true, that indicates whether an entry in the ExtentSettings array parameter can be NULL (indicating that a default setting is to be used).</li> </ol>

**Table 128 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
DirectoryServerParameterSupported		Optional	<p>This enumeration indicates support for the DirectoryServer parameter to the extrinsic method FileSystemConfigurationService.CreateFileSystem(). The options are:</p> <p>'Not Used' indicates that the filesystem does not support security principal information associated with filesystem objects. The LocalFileSystem will not be associated to a DirectoryServer.</p> <p>'Supported' indicates that the filesystem supports security principal information associated with filesystem objects. The LocalFileSystem will be associated to a directory server ComputerSystem. And the DirectoryServer parameter of CreateFileSystem is required. If it is not specified, it will be defaulted to the FileServer parameter in the same call. If the FileServer parameter is also not specified, the DirectoryServer parameter will be defaulted to the host of the FileSystemConfigurationService.</p> <p>'Supported, Defaulted to FileServer' indicates that the filesystem supports security principal information associated with filesystem objects. The LocalFileSystem will be associated to a directory server ComputerSystem. The DirectoryServer parameter of CreateFileSystem is NOT supported, but is automatically defaulted to the FileServer parameter of the same call. If the FileServer parameter is not specified, the DirectoryServer parameter will be defaulted to the host of the FileSystemConfigurationService.</p> <p>'Supported, Defaulted to FileSystem host' indicates that the filesystem supports security principal information associated with filesystem objects. The LocalFileSystem will be associated to a directory server ComputerSystem. The DirectoryServer parameter of CreateFileSystem is NOT supported, but is automatically defaulted to the host of the FileSystem created by CreateFileSystem().</p>

**9.5.10 CIM\_FileSystemConfigurationService**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 129 describes class CIM\_FileSystemConfigurationService.

**Table 129 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationService**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A user-friendly name for this Service.
SystemCreationClassName		Mandatory	The CIM Class name of the ComputerSystem hosting the Service.
SystemName		Mandatory	The Name property of the ComputerSystem hosting the Service.
CreationClassName		Mandatory	The CIM Class name of the Service.
Name		Mandatory	The unique name of the Service.

**Table 129 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationService**

Properties	Flags	Requirement	Description & Notes
CreateFileSystem()		Mandatory	Creates a LocalFileSystem as specified by parameters and Capabilities of the service and returns a reference to it. If appropriate and supported, a Job may be created and a reference to the Job will be returned.
ModifyFileSystem()		Optional	Modifies a LocalFileSystem indicated by a reference and as specified by referenceparameters and Capabilities of the service. If appropriate and supported, a Job may be created and a reference to the Job will be returned.
DeleteFileSystem()		Mandatory	Deletes a LocalFileSystem indicated by reference. If appropriate and supported, a Job may be created and a reference to the Job will be returned.

**9.5.11 CIM\_FileSystemSetting (Attached to FileSystem)**

Created By: Extrinsic: CreateFileSystem

Modified By: Extrinsic: ModifyFileSystem

Deleted By: Extrinsic: DeleteFileSystem

Requirement: Optional

Table 130 describes class CIM\_FileSystemSetting (Attached to FileSystem).

**Table 130 - SMI Referenced Properties/Methods for CIM\_FileSystemSetting (Attached to FileSystem)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for a FileSystemSetting element.
ElementName		Mandatory	A client defined user-friendly name for this FileSystemSetting element.
ActualFileSystemType		Mandatory	This identifies the type of filesystem that this FileSystemSetting represents.
DataExtentsSharing		Optional	This allows the creation of data blocks (or storage extents) that are shared between files.
CopyTarget		Optional	This specifies if support should be provided for using the created filesystem as a target of a Copy operation.
FilenameCaseAttributes		Mandatory	This specifies the support provided for using upper and lower case characters in a filename.
ObjectTypes		Mandatory	This is an array that specifies the different types of objects that this filesystem may be used to provide and provides further details in corresponding entries in other attributes.
NumberOfObjectsMin		Optional	This is an array that specifies the minimum number of objects of the type specified by the corresponding entry in ObjectTypes[] that will be supportable by the LocalFileSystem configured by this FileSystemSetting element.
NumberOfObjectsMax		Optional	This is an array that specifies the maximum number of objects of the type specified by the corresponding entry in ObjectTypes[] that can be supported by the LocalFileSystem configured by this FileSystemSetting element.
NumberOfObjects		Optional	This is an array that specifies the expected number of objects of the type specified by the corresponding entry in ObjectTypes[].

**Table 130 - SMI Referenced Properties/Methods for CIM\_FileSystemSetting (Attached to FileSystem)**

Properties	Flags	Requirement	Description & Notes
ObjectSize		Optional	This is an array that specifies the expected size of a typical object of the type specified by the corresponding entry in ObjectTypes[].
ObjectSizeMin		Optional	This is an array that specifies the minimum size of an object of the type specified by the corresponding entry in ObjectTypes[] that will be supported by the LocalFileSystem configured by this FileSystemSetting element.
ObjectSizeMax		Optional	This is an array that specifies the maximum size of an object of the type specified by the corresponding entry in ObjectTypes[] that can be supported by the LocalFileSystem configured by this FileSystemSetting element.
FilenameStreamFormats		Optional	This is an array that specifies the stream formats (e.g., UTF-8) supported for filenames by the LocalFileSystem configured by this FileSystemSetting element.
FilenameFormats		Optional	This is an array that specifies the formats (e.g. DOS 8.3 names) supported for filenames by the LocalFileSystem configured by this FileSystemSetting element.
FilenameLengthMax		Optional	This specifies the maximum length of a filename that will be supported by the FileSystem configured by this FileSystemSetting element.
FilenameReservedCharacterSet		Optional	This string or character array specifies the characters reserved (i.e., not allowed) for use in filenames that will be required by the FileSystem configured by this FileSystemSetting element.
SupportedLockingSemantics		Optional	This array specifies the set of file access/locking semantics supported by the FileSystem configured by this FileSystemSetting element.
SupportedAuthorizationProtocols		Optional	This array specifies the kind of file authorization protocols supported by the FileSystem configured by this FileSystemSetting element.
SupportedAuthenticationProtocols		Optional	This array specifies the set of file authentication protocols that can be supported by the FileSystem configured by this FileSystemSetting element.

**9.5.12 CIM\_FileSystemSetting (Predefined FS Settings)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 131 describes class CIM\_FileSystemSetting (Predefined FS Settings).

**Table 131 - SMI Referenced Properties/Methods for CIM\_FileSystemSetting (Predefined FS Settings)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for this FileSystemSetting element.
ElementName		Mandatory	A provider supplied user-friendly name for this FileSystemSetting element.
ActualFileSystemType		Mandatory	This identifies the type of filesystem that this FileSystemSetting represents. It shall match the corresponding property of FileSystemCapabilities.
DataExtentsSharing		Optional	This allows the creation of data blocks (or storage extents) that are shared between files.
CopyTarget		Optional	This specifies if support should be provided for using the created filesystem as a target of a Copy operation.
FilenameCaseAttributes		Mandatory	This specifies the support provided for using upper and lower case characters in a filename.
ObjectTypes		Mandatory	This is an array that specifies the different types of objects that this filesystem may be used to provide and provides further details in corresponding entries in other attributes.
NumberOfObjectsMin		Optional	This is an array that specifies the minimum number of objects of the type specified by the corresponding entry in ObjectTypes[] that will be supportable by a LocalFileSystem configured by this FileSystemSetting element.
NumberOfObjectsMax		Optional	This is an array that specifies the maximum number of objects of the type specified by the corresponding entry in ObjectTypes[] that can be supported by a LocalFileSystem configured by this FileSystemSetting element.
NumberOfObjects		Optional	This is an array that specifies the expected number of objects of the type specified by the corresponding entry in ObjectTypes[].
ObjectSize		Optional	This is an array that specifies the expected size of a typical object of the type specified by the corresponding entry in ObjectTypes[].
ObjectSizeMin		Optional	This is an array that specifies the minimum size of an object of the type specified by the corresponding entry in ObjectTypes[] that will be supportable by a LocalFileSystem configured by this FileSystemSetting element.
ObjectSizeMax		Optional	This is an array that specifies the maximum size of an object of the type specified by the corresponding entry in ObjectTypes[] that can be supported by a LocalFileSystem configured by this FileSystemSetting element.
FilenameStreamFormats		Optional	This is an array that specifies the stream formats (e.g., UTF-8) supported for filenames by a filesystem with this setting.
FilenameFormats		Optional	This is an array that specifies the formats (e.g. DOS 8.3 names) supported for filenames by a filesystem with this setting.
FilenameLengthMax		Optional	This specifies the maximum length of a filename supported by a filesystem with this setting.

**Table 131 - SMI Referenced Properties/Methods for CIM\_FileSystemSetting (Predefined FS Settings)**

Properties	Flags	Requirement	Description & Notes
FilenameReservedCharacterSet		Optional	This string or character array specifies the characters reserved (i.e., not allowed) for use in filenames that will be required by a filesystem with this setting.
SupportedLockingSemantics		Optional	This array specifies the set of file access/locking semantics supported by a filesystem with this setting.
SupportedAuthorizationProtocols		Optional	This array specifies the kind of file authorization protocols supported by a filesystem with this setting.
SupportedAuthenticationProtocols		Optional	This array specifies the kind of file authentication protocols supported by a filesystem with this setting.

**9.5.13 CIM\_HostedDependency (Attached to File System)**

Created By: Extrinsic: CreateFileSystem

Modified By: Extrinsic: ModifyFileSystem

Deleted By: Extrinsic: DeleteFileSystem

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true.

Table 132 describes class CIM\_HostedDependency (Attached to File System).

**Table 132 - SMI Referenced Properties/Methods for CIM\_HostedDependency (Attached to File System)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Scoping File Server ComputerSystem.
Dependent		Mandatory	The Local Access Setting that is scoped by the file server ComputerSystem.

**9.5.14 CIM\_HostedDependency (Predefined Capabilities)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'.

Table 133 describes class CIM\_HostedDependency (Predefined Capabilities).

**Table 133 - SMI Referenced Properties/Methods for CIM\_HostedDependency (Predefined Capabilities)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Scoping file server ComputerSystem.
Dependent		Mandatory	The LocallyAccessibleFileSystemCapabilities that is scoped by the file server ComputerSystem.

**9.5.15 CIM\_HostedDependency (Predefined Setting)**

Created By: Static

Modified By: Static



Deleted By: Static  
Requirement: Optional

Table 134 describes class CIM\_HostedDependency (Predefined Setting).

**Table 134 - SMI Referenced Properties/Methods for CIM\_HostedDependency (Predefined Setting)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Scoping file server ComputerSystem.
Dependent		Mandatory	The Local Access Setting that is scoped by the file server ComputerSystem.

#### 9.5.16 CIM\_HostedFileSystem

Created By: Extrinsic: CreateFileSystem  
Modified By: Extrinsic: ModifyFileSystem  
Deleted By: Extrinsic: DeleteFileSystem  
Requirement: Mandatory

Table 135 describes class CIM\_HostedFileSystem.

**Table 135 - SMI Referenced Properties/Methods for CIM\_HostedFileSystem**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The ComputerSystem that hosts a LocalFileSystem. The Dedicated property must be one of 24 (NAS Head), 25 (SC NAS), 16 (File Server).
PartComponent		Mandatory	The hosted filesystem.

#### 9.5.17 CIM\_HostedService

Created By: Static  
Modified By: Static  
Deleted By: Static  
Requirement: Mandatory

Table 136 describes class CIM\_HostedService.

**Table 136 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The Filesystem Configuration Service.
Antecedent		Mandatory	The hosting ComputerSystem. This can be the top level system or a component ComputerSystem of the Multiple Computer System profile.

#### 9.5.18 CIM\_LocalAccessAvailableToFS

Created By: Extrinsic: CreateFileSystem  
Modified By: Extrinsic: ModifyFileSystem  
Deleted By: Extrinsic: DeleteFileSystem

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true.

Table 137 describes class CIM\_LocalAccessAvailableToFS.

**Table 137 - SMI Referenced Properties/Methods for CIM\_LocalAccessAvailableToFS**

Properties	Flags	Requirement	Description & Notes
LocalAccessPoint		Conditional	Conditional requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true. The name used by the file server to identify the filesystem. Sometimes referred to as a mount-point. For many UNIX-based systems, this will be a qualified full pathname. For Windows systems this could also be the drive letter used for the LogicalDisk that the filesystem is resident on.
FileSystem		Mandatory	The LocalFileSystem that is being made available to the file server ComputerSystem.
FileServer		Mandatory	The file server ComputerSystem that will be able to export shares from this LocalFileSystem.

### 9.5.19 CIM\_LocalFileSystem

The following properties of LocalFileSystem are defined by the MOF, but the way we model LocalFileSystem has changed significantly. The setting/configuration properties are not supported using these properties, and so all of these are "Not Supported". The run-time properties will be supported by a statistics/performance profile and that has yet to be defined.

Created By: Extrinsic: CreateFileSystem

Modified By: Extrinsic: ModifyFileSystem

Deleted By: Extrinsic: DeleteFileSystem

Requirement: Mandatory

Table 138 describes class CIM\_LocalFileSystem.

**Table 138 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
LocalAccessDefinitionRequired		Mandatory	This boolean property indicates whether or not a LocalFileSystem with this FileSystemSetting must be made locally accessible ("mounted") from a file server ComputerSystem before it can be shared or otherwise made available to operational clients.
PathNameSeparatorString		Mandatory	This indicates the string of characters used to separate directory components of a canonically formatted path to a file from the root of the filesystem. This string is expected to be specific to the ActualFileSystemType and so is vendor/implementation dependent. However, by surfacing it we make it possible for a client to parse a pathname into the hierarchical sequence of directories that compose it.

**Table 138 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
DirectoryServiceUsage		Optional	<p>This enumeration indicates whether the filesystem supports security principal information and therefore requires support from a file server that uses one or more directory services. If the filesystem requires such support, there must be a concrete subclass of Dependency between the LocalFileSystem element and the specified file server ComputerSystem. The values supported by this property are:</p> <p>'Not Used' indicates that the filesystem will not support security principal information and so will not require support from a directory service.</p> <p>'Optional' indicates that the filesystem may support security principal information. If it does, it will require support from a directory service and the Dependency association described above must exist.</p> <p>'Required' indicates that the filesystem supports security principal information and will require support from a directory service. The Dependency association described above must exist.</p>
CSCreationClassName		Mandatory	The CIM class name of the hosting ComputerSystem.
CSName		Mandatory	The Name property of the hosting ComputerSystem.
CreationClassName		Mandatory	The CIM class name of the this element.
Name		Mandatory	A unique name for this LocalFileSystem in the context of the hosting ComputerSystem.
EnabledState		Optional	Current state of enablement of the LocalFileSystem.
OtherEnabledState		Optional	Vendor-specific state of the LocalFileSystem indicated by EnabledState = 1("Other").
TimeOfLastStateChange		Optional	A timestamp indicating when the state was last changed.
RequestedState		Optional	Not supported.
OperationalStatus		Mandatory	The current operational status of the LocalFileSystem.
Root		Optional	A path that specifies the "mount point" of the filesystem in an unitary computer system that is both the host of the filesystem and is the file server that makes it available.
BlockSize		Mandatory	The size of a block in bytes that the implementation used as a fixed block size when creating this filesystem.
FileSystemSize		Mandatory	The total current size of the filesystem in blocks.
AvailableSpace		Mandatory	The space available currently in the filesystem in blocks.
ReadOnly		Optional	Indicates that this is a read-only filesystem that does not allow modifications.
EncryptionMethod		Optional	Indicates if files are encrypted and the method of encryption.
CompressionMethod		Optional	Indicates if files are compressed before being stored, and the methods of compression.
CaseSensitive		Optional	Whether this filesystem is sensitive to the case of characters in filenames.
CasePreserved		Optional	Whether this filesystem preserves the case of characters in filenames when saving and restoring.

**Table 138 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
CodeSet		Optional	The codeset used in filenames.
MaxFileNameLength		Optional	The length of the longest filename supported by the implementation.
ClusterSize		Optional	Not supported.
FileSystemType		Optional	This is a string that matches FileSystemSetting.ActualFileSystemType property used to create the filesystem.  Pragmatically, this property should be ignored.
NumberOfFiles		Optional	The actual current number of files in the filesystem. This value is an approximation as it can vary continuously when the filesystem is in use.
IsFixedSize		Optional	Indicates that the filesystem cannot be expanded or shrunk.
ResizeIncrement		Optional	The size by which to increase the size of the filesystem when requested.
RequestStateChange()		Optional	Not supported.

**9.5.20 CIM\_LocallyAccessibleFileSystemCapabilities**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'.

Table 139 describes class CIM\_LocallyAccessibleFileSystemCapabilities.

**Table 139 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the CIM_LocallyAccessibleFileSystemCapabilities associated to a Filesystem Configuration Service.
ElementName		Mandatory	A user-friendly name for this CIM_LocallyAccessibleFileSystemCapabilities element.

**Table 139 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemCapabilities**

Properties	Flags	Requirement	Description & Notes
SupportedProperties		Mandatory	<p>An array of property names of the LocallyAccessibleFileSystemSetting that this CIM_LocallyAccessibleFileSystemCapabilities element supports.</p> <p>2 'FailurePolicy'</p> <p>3 'RetriesMax'</p> <p>4 'InitialEnabledState'</p> <p>5 'RequestRetryPolicy'</p> <p>6 'TransmissionRetriesMax'</p> <p>7 'RetransmissionTimeout'</p> <p>8 'CachingOptions'</p> <p>9 'ReadBufferSize'</p> <p>10 'WriteBufferSize'</p> <p>11 'AttributeCaching'</p> <p>12 'ReadWritePolicy'</p> <p>13 'LockPolicy'</p> <p>14 'EnableOnSystemStart'</p> <p>15 'ReadWritePref'</p> <p>16 'ExecutePref'</p> <p>17 'RootAccessPref'.</p>
SupportedObjectsForAttributeCaching		Optional	<p>If AttributeCaching is supported, this specifies the array of objects that can be set up for caching. A subset of these entries will become the entries of the AttributeCachingObjects property in the Setting.</p> <p>These classes represent types of objects stored in a filesystem implementation -- files and directories as well as others that may be defined in the future. The corresponding Setting properties, AttributeCaching, AttributeCachingTimeMin, and AttributeCachingTimeMax provide the supported features for the type of object. 'None' and 'All' cannot both be specified; if either one is specified, it must be the first entry in the array and the entry is interpreted as the default setting for all objects. If neither 'None' or 'All' are specified, the caching settings for other objects are defaulted by the implementation. If 'Rest' is specified, the entry applies to all known object types other than the named ones. If 'Unknown' is specified it applies to object types not known to this application (this can happen when foreign file systems are mounted).</p> <p>0 'Unknown'</p> <p>1 'None'</p> <p>2 'All'</p> <p>3 'Rest'</p> <p>4 'File'</p> <p>5 'Directory'.</p>

#### 9.5.21 CIM\_LocallyAccessibleFileSystemSetting

Created By: Extrinsic: CreateFileSystem

Modified By: Extrinsic: ModifyFileSystem

Deleted By: Extrinsic: DeleteFileSystem or ModifyFileSystem

Requirement: Required if LocalFileSystem.LocalAccessDefinitionRequired=true.

Table 140 describes class CIM\_LocallyAccessibleFileSystemSetting.

**Table 140 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for a LocallyAccessibleFileSystemSetting.
ElementName		Mandatory	A user-friendly name for this LocallyAccessibleFileSystemSetting element.
InitialEnabledState		Optional	<p>InitialEnabledState is an integer enumeration that indicates the enabled/disabled states initially set for a locally accessible filesystem (LAFS). The element functions by passing commands onto the underlying filesystem, and so cannot indicate transitions between requested states because those states cannot be requested. The following text briefly summarizes the various enabled/disabled initial states:</p> <p>'Enabled' (2) indicates that the element will execute commands, will process any queued commands, and will queue new requests.</p> <p>'Disabled' (3) indicates that the element will not execute commands and will drop any new requests.</p> <p>'In Test' (7) indicates that the element will be in a test state.</p> <p>'Deferred' (8) indicates that the element will not process any commands but will queue new requests.</p> <p>'Quiesce' (9) indicates that the element is enabled but in a restricted mode. The element's behavior is similar to the Enabled state, but it only processes a restricted set of commands. All other requests are queued.</p>
OtherEnabledState		Optional	A string describing the element's initial enabled/disabled state when the InitialEnabledState property is set to 1 ("Other"). This property MUST be set to NULL when InitialEnabledState is any value other than 1.
FailurePolicy		Optional	An enumerated value that specifies if the operation to make a FileSystem locally accessible to a scoping ComputerSystem should be attempted one or more times in the foreground or tried repeatedly in the background until it succeeds. The number of attempts would be limited by the corresponding RetriesMax property of the setting.
RetriesMax		Optional	An integer specifying the maximum number of attempts that should be made by the scoping ComputerSystem to make a filesystem locally accessible. A value of "0" specifies an implementation-specific default.
RequestRetryPolicy		Optional	An enumerated value representing the policy that is supported by the operational file server on a request to the operational filesystem that either failed or left the file server hanging. If the request is being performed in the foreground, the options are to try once and fail if a timeout happens, or, to try repeatedly. If the request can be performed in the background, the request will be tried repeatedly until stopped.
TransmissionRetriesMax		Optional	An integer specifying the maximum number of retransmission attempts to be made from the operational file server to the operational filesystem when the transmission of a request fails or makes the file server hang. A value of "0" specifies an implementation-specific default. This is only relevant if there is a transmission channel between the file server and the underlying filesystem.

**Table 140 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
RetransmissionTimeoutMin		Optional	An integer specifying the minimum number of milliseconds that the operational file server must wait before assuming that a request to the operational filesystem has failed. "0" indicates an implementation-specific default. This is only relevant if there is a transmission channel between the operational file server and the operational filesystem.
CachingOptions		Optional	An enumerated value that specifies if a local cache is supported by the operational file server when accessing the underlying operational filesystem.
BuffersSupport		Optional	An array or enumerated values that specifies the buffering mechanisms supported by the operational file server for accessing the underlying operational filesystem." If supported, other properties will establish the level of support. If the property is NULL or the empty array, buffering is not supported.
ReadBufferSizeMin		Optional	An integer specifying the minimum number of bytes that must be allocated to each buffer used for reading. A value of "0" specifies an implementation-specific default.
ReadBufferSizeMax		Optional	An integer specifying the maximum number of bytes that may be allocated to each buffer used for reading. A value of "0" specifies an implementation-specific default.
WriteBufferSizeMin		Optional	An integer specifying the minimum number of bytes that must be allocated to each buffer used for writing. A value of "0" specifies an implementation-specific default.
WriteBufferSizeMax		Optional	An integer specifying the maximum number of bytes that may be allocated to each buffer used for writing. A value of "0" specifies an implementation-specific default.
AttributeCaching		Optional	<p>An array of enumerated values that specify whether attribute caching is (or is not) supported by the operational file server when accessing specific types of objects from the underlying operational filesystem. The object type and the support parameters are specified in the corresponding AttributeCachingObjects, AttributeCachingTimeMin, and AttributeCachingTimeMax array properties.</p> <p>Filesystem object types that can be accessed locally are represented by an entry in these arrays. The entry in the AttributeCaching array can be "On", "Off", or "Unknown". Implementation of this feature requires support from other system components, so it is quite possible that specifying "On" may still not result in caching behavior. "Unknown" indicates that the access operation will try to work with whatever options the operational file server and filesystem can support. In all cases, AttributeCachingTimeMin and AttributeCachingTimeMax provide the minimum and maximum time for which the attributes can be cached. When this Setting is used as a Goal, the client may specify "Unknown", but the Setting in the created object should contain the supported setting, whether "On" or "Off".</p>

**Table 140 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
AttributeCachingObjects		Optional	An array of enumerated values that specify the attribute caching support provided to various object types by the operational file server when accessing the underlying operational filesystem. These", types represent the types of objects stored in a FileSystem -- files and directories as well as others that may be defined in the future. The corresponding properties, AttributeCaching, AttributeCachingTimeMin, and AttributeCachingTimeMax provide the supported features for the type of object. "None" and "All" cannot both be specified; if either one is specified, it must be the first entry in the array and the entry is interpreted as the default setting for all objects. If neither "None" or "All" are specified, the caching settings for other objects are defaulted by the implementation. If "Rest" is specified, the entry applies to all known object types other than the named ones. If "Unknown" is specified it applies to object types not known to this application (this can happen when foreign file systems are mounted).
AttributeCachingTimeMin		Optional	An array of integers specifying, in milliseconds, the minimum time for which an object of the type specified by the corresponding AttributeCaching property must be retained in the attribute cache. When used as a Goal, a value of "0" indicates an implementation-specific default.
AttributeCachingTimeMax		Optional	An array of integers specifying, in milliseconds, the maximum time for which an object of the type specified by the corresponding AttributeCaching property must be retained in the attribute cache. When used as a Goal, a value of "0" indicates an implementation-specific default.
ReadWritePolicy		Optional	An enumerated value that specifies the Read-Write policy set on the operational filesystem and supported by the operational file server when accessing it. 'Read Only' specifies that the access to the operational filesystem by the operational file server is set up solely for reading. 'Read/Write' specifies that the access to the operational filesystem by the operational file server is set up for both reading and writing. 'Force Read/Write' specifies that 'Read-Only' has been overridden by a client with write access to the operational filesystem. This option is intended for use when the associated FileSystem has been made 'Read Only' by default, as might happen if it were created to be the target of a Synchronization or Mirror operation.
LockPolicy		Optional	An enumerated value that specifies the Locking that will be enforced on the operational filesystem by the operational file server when accessing it. 'Enforce None' does not enforce locks. 'Enforce Write' does not allow writes to locked files. 'Enforce Read/Write' does not allow reads or writes to locked files.
EnableOnSystemStart		Optional	An enumerated value that specifies if local access from the operational file server to the operational filesystem should be enabled when the file server is started.
ReadWritePref		Optional	An instance of a CIM_Privilege, encoded as a string, that expresses the client's expectations about access to elements contained in the operational filesystem. The provider is expected to surface this access using the CIM privilege model.



**Table 140 - SMI Referenced Properties/Methods for CIM\_LocallyAccessibleFileSystemSetting**

Properties	Flags	Requirement	Description & Notes
ExecutePref		Optional	An enumerated value that specifies if support should be provided on the operational file server for executing elements contained in the operational filesystem accessed through this local access point. This may require setting up specialized paging or execution buffers either on the operational file server or on the operational filesystem side (as appropriate for the implementation). Note that this does not provide any rights to actually execute any element but only specifies support for such execution, if permitted.
RootAccessPref		Optional	An instance of a CIM_Privilege, encoded as a string, that expresses the client's expectations about privileged access by appropriately privileged System Administrative users on the operational file server ("root" or "superuser") to the operational filesystem and its elements. The provider is expected to surface this access using the CIM privilege model.  Support for the privileged access might require setup at both the operational file server as well as the operational filesystem, so there is no guarantee that the request can be satisfied.

**9.5.22 CIM\_SettingsDefineCapabilities (Predefined FS Settings)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 141 describes class CIM\_SettingsDefineCapabilities (Predefined FS Settings).

**Table 141 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (Predefined FS Settings)**

Properties	Flags	Requirement	Description & Notes
PropertyPolicy		Mandatory	PropertyPolicy defines whether or not the non-null, non-key properties of the associated FileSystemSetting element are treated independently or as a correlated set.
ValueRole		Mandatory	ValueRole specifies the semantics of the non-null, non-key properties of the associated FileSystemSetting element, such as whether they are supported or unsupported, and if supported, whether they are a default and/or an optimal value or an average of some kind.
ValueRange		Mandatory	ValueRange specifies the semantics of the non-null, non-key properties of the associated FileSystemSetting element, such as whether they are point properties, or whether they represent maximum or minimum values for the properties. If some properties already have maximums and/or minimums specified by another FileSystemSetting instance, this could specify increments of the property value that are supported.
GroupComponent		Mandatory	A Filesystem Capabilities element that is defined by a collection of filesystem settings.
PartComponent		Mandatory	A filesystem setting that provides a point or a partial definition for a Filesystem Capabilities element.

**9.5.23 CIM\_SettingsDefineCapabilities (Predefined Local Access Settings)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if FileSystemConfigurationCapabilities.LocalAccessibilitySupport is either 'Local Access Required, Defaulted' or 'Local Access Required, Not Defaulted'.

Table 142 describes class CIM\_SettingsDefineCapabilities (Predefined Local Access Settings).

**Table 142 - SMI Referenced Properties/Methods for CIM\_SettingsDefineCapabilities (Predefined Local Access Settings)**

Properties	Flags	Requirement	Description & Notes
PropertyPolicy		Mandatory	PropertyPolicy defines whether or not the non-null, non-key properties of the associated LocallyAccessibleFileSystemSetting instance are treated independently or as a correlated set.
ValueRole		Mandatory	ValueRole specifies the semantics of the non-null, non-key properties of the associated LocallyAccessibleFileSystemSetting instance, such as whether they are supported or unsupported, and if supported, whether they are a default and/or an optimal value or an average of some kind.
ValueRange		Mandatory	ValueRange specifies the semantics of the non-null, non-key properties of the associated LocallyAccessibleFileSystemSetting instance, such as whether they are point properties, or whether they represent maximum or minimum values for the properties. If some properties already have maximums and/or minimums specified by another LocallyAccessibleFileSystemSetting instance, this could specify increments of the property value that are supported.
GroupComponent		Mandatory	A Capabilities element of the filesystem that is defined by a collection of LocallyAccessibleFileSystemSetting elements, each being scoped to the File Server ComputerSystem with which it can be used.
PartComponent		Mandatory	A LocallyAccessibleFileSystemSetting that provides a point or a partial definition for a LocallyAccessibleFileSystemCapabilities element.

---



---

**STABLE**

---

---

## EXPERIMENTAL

### 10 Filesystem Performance Profile

#### 10.1 Description

##### 10.1.1 Synopsis

**Profile Name:** Filesystem Performance (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** FileSystemStatisticsService

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 143 describes the related profiles for Filesystem Performance.

**Table 143 - Related Profiles for Filesystem Performance**

Profile Name	Organization	Version	Requirement	Description
Filesystem	SNIA	1.7.0	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "102" (Local Filesystem statistics support).
File Export	SNIA	1.7.0	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "103" (Exported File Share statistics support).
NAS Network Port	SNIA	1.7.0	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "104" (Exporting Port statistics support).

NOTE Each of these component profiles is mandatory if the element in question is to be metered. For example, in order to keep statistics on exported file shares, it will be necessary for File Shares to be modeled through the use of the File Export Profile.

##### 10.1.2 Overview

The Filesystem Performance Profile defines classes and methods for managing filesystem-related performance information. It is a component profile for use with autonomous profiles that directly support filesystems, which in this release of SMI-S specifically includes the NAS Head and the Self-Contained NAS Profiles.

One of the key application disciplines for managing storage is Performance Management. In order to manage performance, a number of processes need to be in place, including the ability to measure the performance and saturation points of components within the storage network.

There are currently no common statistics defined that can be used to manage multiple vendor filesystem-related entities (such as File Servers) from a performance perspective. This component profile defines specific measurements and methods to make common statistics available to client applications regarding filesystem-related entities. Examples of such statistics include:

- The read, write and other I/O operation counts for a filesystem or a file share,
- The cumulative elapsed time required for the I/O operations to complete,

- The number of bytes transferred per unit of time.

Particular areas related to Performance Management that can make use of the statistics provided by the Filesystem Performance Profile include:

- Filesystem utilization (e.g., "hot-spot" and trend analyses; tracking usage efficiency by monitoring response times and IOPS/throughput rates; identifying over-utilization and contention that is leading to performance degradation).
- Diagnostics and problem determination (e.g., identifying bottlenecks, "point(s) of pain", etc., especially at an upper level within the overall "I/O operation stack").
- Tuning (e.g., determining allocation/reallocation of particular filesystems and/or file placements in the efforts to meet overall performance goals and/or other Service Level Agreements; determining the impact of the underlying storage and applicable network provisioning upon filesystem performance and utilization).
- Workload characterization (e.g., characterizing particular filesystem usage with possible correlation to associated applications).
- Modeling and planning (e.g., enabling the use of empirical metrics as the input/basis for various modeling and planning exercises related to filesystem and overall storage concerns).

Performance Measurement within the context of filesystems is the key deliverable that is the focus of this profile. Of particular importance, the statistics provided by the Filesystem Performance Profile can help facilitate a "top-down" approach within the areas noted above (i.e., by reflecting performance information that is directly related to and seen by/at a "top-most" component within the overall I/O operation processing stack).

NOTE Performance analysis is broader than simply filesystems and related entities such as File Servers. Complete analysis requires performance information from hosts, fabric and the underlying storage systems. These are (or will be) addressed separately as part of the appropriate profiles (e.g., the Block Server Performance Profile, which includes further discussion regarding Performance Management).

The Filesystem Performance Profile provides statistics, which are associated with fundamental elements that can comprise a filesystem-related entity (such as a NAS Head or a Self-Contained NAS). These elements include:

- Filesystems
- Exported file shares
- Network-interface ports used to export file shares

In order to monitor and manage the aforementioned elements, it is necessary to identify performance counters for each of these elements and to externalize an interface so that client applications can retrieve the counter values when they so desire. The function of this profile is to support such client applications.

The Filesystem Performance Profile augments the profiles for those autonomous profiles within this release of SMI-S that directly support filesystems. Instead of being an isolated component profile, this profile adds modeling constructs to existing profiles. Together these enhancements make up the Filesystem Performance Profile (as would be registered in the Server Profile as a RegisteredProfile).

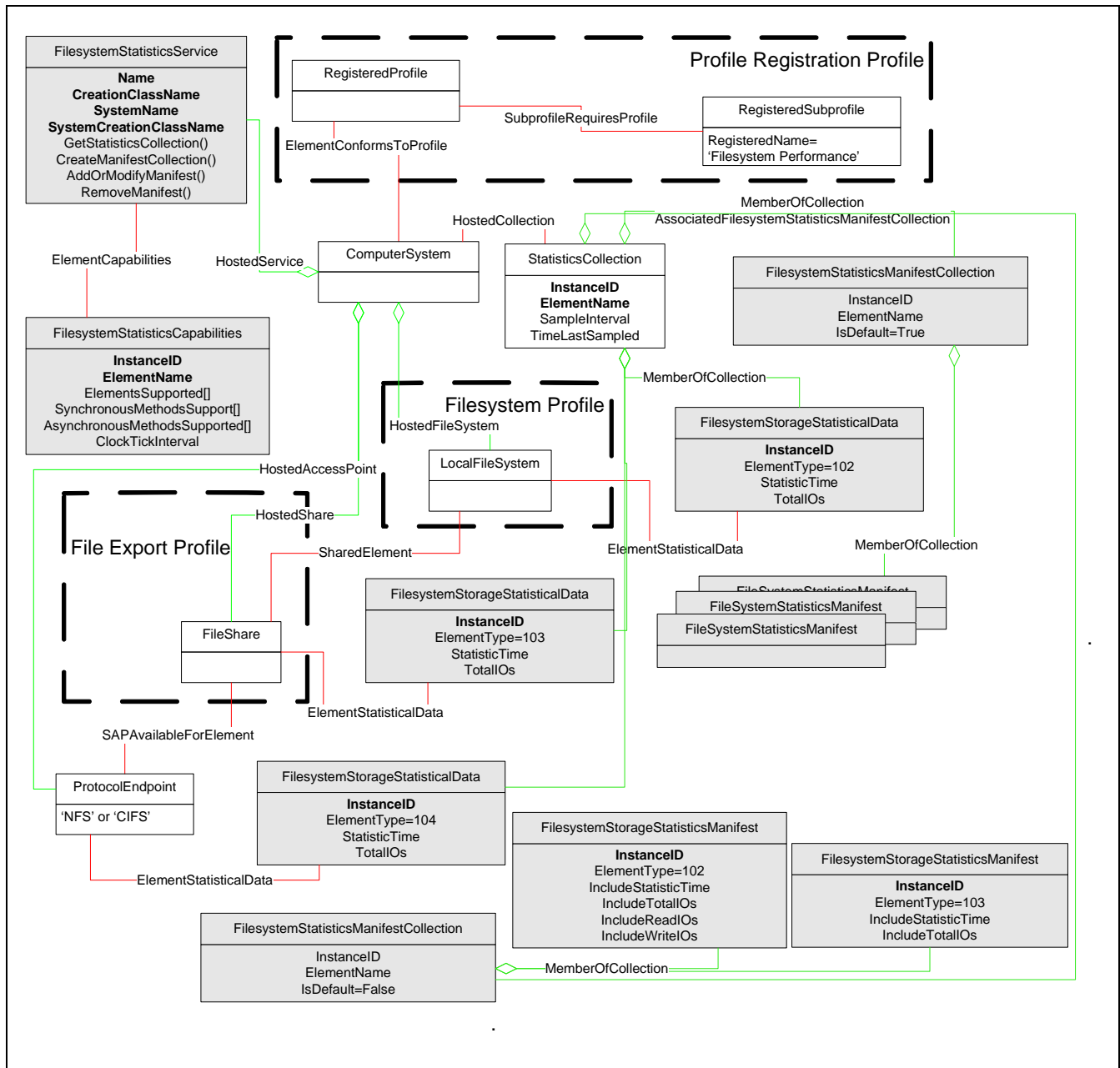
## 10.2 Implementation

### 10.2.1 Performance Additions Overview

Figure 16 provides an overview of the model. The shaded grey boxes show the new classes added by the Filesystem Performance Profile.

## Filesystem Performance Profile

NOTE Not all properties defined for the statistics classes are shown within Figure 16: "Filesystem Performance Profile Summary Instance Diagram". That is, there are additional properties (both mandatory and optional) that are included within the statistical classes. These properties can be found in 10.5 "CIM Elements".



### Figure 16 - Filesystem Performance Profile Summary Instance Diagram

Figure 16: "Filesystem Performance Profile Summary Instance Diagram" shows a single instance of StatisticsCollection for the entire profile. The ComputerSystem (i.e., the "top level" computer system depicted within the figure) is that of the autonomous profile (e.g., a NAS Head or a Self-Contained NAS) which utilizes the Filesystem Performance Profile.

The `StatisticsCollection` is the anchor point from which all statistics being kept by the profile can be found. Statistics are defined as a `FileSystemStatisticalData` class, instances of which hold the statistics for particular metered elements (e.g., filesystems and file shares). The particular type of metered element is recorded in the instance of `FileSystemStatisticalData` within the `ElementType` property.

All of the statistics instances are related to the elements that they meter via the ElementStatisticalData association (e.g., FileSystemStatisticalData for a File Share can be found from the File Share by traversing the ElementStatisticalData association).

All of the statistics instances kept within the profile are associated to the one StatisticsCollection instance. Access to all of the statistics for the profile is through the StatisticsCollection. The StatisticsCollection has a HostedCollection association to the "top level" computer system of the profile.

Note that statistics may be kept for a number of elements within the profile, including elements within component profiles. The particular elements that are metered are:

- **Local filesystem.** This provides a summary of all statistics for a particular filesystem (i.e., an instance of LocalFileSystem). For example, all file read I/O operations (ReadIOs) directed to a particular filesystem. These statistics are kept within the FileSystemStatisticalData instances, with one for each filesystem within the system.
- **Exported file share.** This provides a summary of all statistics for a particular file share that is exported (i.e., an instance of FileShare as described within the File Export Profile). For example, all file read I/O operations (ReadIOs) directed to a particular file share that is exported to the network. These statistics are kept within the FileSystemStatisticalData instances, with one for each FileShare within the system.
- **Exporting port.** This provides a summary of all statistics for a particular port through which a file share being exported can be accessed (i.e., an instance of ProtocolEndpoint through which a FileShare can be accessed as described within the File Export Profile). For example, all file read I/O operations (ReadIOs) directed to a particular file share exporting port. These statistics are kept within the FileSystemStatisticalData instances, with one for each file share exporting port within the system.

Finally, Figure 16: "Filesystem Performance Profile Summary Instance Diagram" illustrates the FileSystemStatisticsService for Bulk retrieval of all the statistics data and the creation of manifest collections. These methods (which are provided in a manner akin to that provided by the Block Server Performance Profile) will be discussed later. They are shown here for completeness. Associated with the FileSystemStatisticsService is a FileSystemStatisticsCapabilities instance that identifies the specific capabilities implemented by the filesystem performance statistics support. Specifically, it includes an "ElementsSupported" property that identifies the elements for which statistics are kept; the FileSystemStatisticsCapabilities instance also identifies the various retrieval mechanisms (e.g., Extrinsic, Association Traversal, Indications and/or Query) that are implemented (i.e., supported) by the filesystem statistics support.

### 10.2.2 Summary of FileSystemStatisticalData support by Profile

Table 144 defines the Element Types (for FileSystemStatisticalData instances) that may be supported by profile.

**Table 144 - Summary of Element Types by Profile**

ElementType	NAS Head	Self-Contained NAS
Local filesystem	YES	YES
Exported File Share	YES	YES
Exporting Port	YES	YES

YES means that this specification defines the element type for the profile, but actual support by any given implementation would be implementation dependent. NO means that this specification does not specify this element type for the profile.

### 10.2.3 Profile Registration Profile Support for the Filesystem Performance Profile

At the top of Figure 16: "Filesystem Performance Profile Summary Instance Diagram" there is a dashed box that illustrates a part of the Profile Registration Profile for the autonomous profile (e.g., a NAS Head or a Self-Contained NAS) that utilizes the Filesystem Performance Profile. The part illustrated represents the particulars for the Filesystem Performance Profile. If performance support has been implemented, then there shall be a RegisteredProfile instance for the Filesystem Performance Profile.

### 10.2.4 Default Manifest Collection

Associated with the instances of the StatisticsCollection shall be a provider-supplied (Default) CIM\_FileSystemManifestCollection that represents the statistics properties that are kept by the profile. The default manifest collection is indicated by the IsDefault property (=True) of the CIM\_FileSystemManifestCollection. For each metered object (element) of the profile implementation, the default manifest collection will have exactly one manifest that will identify which properties are included for that metered object. If an object is not metered, then there shall not be a manifest for that element type. If an element type (e.g., Local filesystem) is metered, then there shall be a manifest for that element type.

### 10.2.5 Client Defined Manifest Collection

Manifest collections are either provider-supplied (CIM\_FileSystemManifestCollection.IsDefault=True) for the profile implementation or client-defined collections (CIM\_FileSystemManifestCollection.IsDefault=False). Client-defined collections are used to indicate the specific statistics properties that the client would like to retrieve using the GetStatisticsCollection method. For a discussion of provider-supplied manifest collections, see 10.2.4.

Client-defined manifest collections are a mechanism for restricting the amount of data returned on a GetStatisticsCollection request. A client-defined manifest collection is identified by the IsDefault property of the collection set to False. For each element type of the filesystem statistics class (e.g., Local filesystem, exported file share, etc.), a manifest can be defined that identifies which specific properties of the particular statistics class element type are to be returned on a GetStatisticsCollection request. Each of the element types of the filesystem statistics class may have no or one manifest in any given manifest collection. This is illustrated in Figure 16: "Filesystem Performance Profile Summary Instance Diagram".

In Figure 16: "Filesystem Performance Profile Summary Instance Diagram", manifest classes are defined for filesystems (LocalFilesystem) and exported file shares (FileShare). Each property of the manifest is a Boolean that indicates whether the property is to be returned (true) or omitted (false).

Multiple client-defined manifest collections can be defined in the profile. Consequently, different clients or different client applications can define different manifests for different application needs. A manifest collection can completely omit a whole set of statistics pertaining to a particular element type; for example, no ProtocolEndPoint statistics (i.e., filesystem performance statistics associated with the element type of "Exporting Port", which represents a port through which a File Share can be accessed from the network) are included within the client-defined manifest collection shown in Figure 16: "Filesystem Performance Profile Summary Instance Diagram". Since manifest collections are "client objects", they are named (ElementName) by the client for the client's convenience. The CIM server will generate an instance ID to uniquely identify the manifest collection in the CIM Server.

Client-defined manifest collections are created using the CreateManifestCollection method. Manifests are added or modified using the AddOrModifyManifest method. A manifest may be removed from the manifest collection by using the RemoveManifests method.

**NOTE** Use of manifest collections is optional with the GetStatisticsCollection method. If NULL for the manifest collection is passed on input, then all statistics instances are assumed (i.e., all available statistics will be returned).

## 10.2.6 Capabilities Support for Filesystem Performance Profile

There are two dimensions to determining what is supported with a Filesystem Performance Profile implementation. First, there are the RegisteredProfiles supported by the autonomous profile (e.g., a NAS Head or a Self-Contained NAS Profile) that utilizes the Filesystem Performance Profile. In order to support statistics for a particular class of metered element, the corresponding object shall be modeled. So, if a NAS Head (for example) has not implemented the File Export Profile, then it shall not implement the FileSystemStatisticalData for "Exported File Share" in the Filesystem Performance Profile (and implementation of the File Export Profile does not guarantee implementation of the FileSystemStatisticalData for exported file shares).

Both of these dimensions are captured in the FileSystemStatisticsCapabilities class instance. This class instance is not created nor modified by Clients; rather, it is populated by the provider and has three properties of interest (as discussed within the following sections). The second dimension is techniques supported for retrieving statistics and manipulating manifest collections.

For the methods-supported properties described below (namely, SynchronousMethodsSupported and AsynchronousMethodsSupported), any or all of the respective values can be missing (e.g., the arrays can be NULL). If all of the methods supported are NULL, then manifest collections are not supported and neither GetStatisticsCollection nor Query are supported for the retrieval of statistics. This leaves enumerations or association traversals as the only methods for retrieving the statistics.

### 10.2.6.1 ElementsSupported

This property within the FileSystemStatisticsCapabilities class defines a list of element types for which statistical data is available. For this release of SMI-S, the values of interest are "Local Filesystem", "Exported File Share" and "Exporting Port".

To be a valid implementation of the Filesystem Performance Profile, at least one of the values listed for ElementsSupported shall be supported. ElementsSupported is an array, such that all of the values can be identified.

### 10.2.6.2 SynchronousMethodsSupported

This property within the FileSystemStatisticsCapabilities class defines the synchronous mechanisms that are supported for retrieving statistics and for defining and modifying filters for statistics retrieval. For this release of SMI-S, the values of interest are "Exec Query", "Indications", "Query Collection", "GetStatisticsCollection", "Manifest Creation", "Manifest Modification", and "Manifest Removal".

### 10.2.6.3 AsynchronousMethodsSupported

This property within the FileSystemStatisticsCapabilities class defines the asynchronous mechanisms that are supported for retrieving statistics. For this release of SMI-S, this should be NULL.

### 10.2.6.4 ClockTickInterval

An internal clocking interval for all timer counters kept in the system implementation, measured in microseconds (i.e., the unit of measure in the timers, measured in microseconds). Time counters are considered to be monotonically increasing counters that contain "ticks". Each tick represents one clock tick interval.

For example, if ClockTickInterval contained a value of 32, then each time counter tick would represent 32 microseconds.

## 10.2.7 Health and Fault Management Consideration

Not defined in this standard.

## 10.2.8 Cascading Considerations

Not defined in this standard



## 10.3 Methods of the Profile

### 10.3.1 Extrinsic Methods of the Profile

#### 10.3.1.1 Overview

The methods supported by this profile are summarized in Table 145 and detailed within the sections that follow it.

**Table 145 - Creation, Deletion and Modification Methods in the Filesystem Performance Profile**

Method	Created Instances	Deleted Instances	Modified Instances
GetStatisticsCollection	None	None	None
CreateManifestCollection	FileSystemStatisticsManifestCollection AssociatedFileSystemStatisticsManifestCollection	None	None
AddOrModifyManifest	FileSystemStatisticsManifest(subclass) MemberOfCollection	None	FileSystemStatisticsManifest(subclass)
RemoveManifest	None	FileSystemStatisticsManifest(subclass) MemberOfCollection	None

#### 10.3.1.2 GetStatisticsCollection

This extrinsic method retrieves statistics in a well-defined bulk format. The set of statistics returned by this method is determined by the list of element types passed into the method and the manifests for those types contained in the supplied manifest collection. The statistics are returned through a well-defined array of strings that can be parsed to retrieve the desired statistics as well as limited information about the elements that those metrics describe.

```

GetStatisticsCollection(
    [IN (false), OUT, Description(Reference to the job(shall be null in this
                                version of SMI-S.))]
    CIM_ConcreteJob REF Job,
    [IN, Description(Element types for which statistics should be returned)
    ValueMap { "1", "102", "103", "104", "..", "0x8000.." },
    Values { "Other", "Local Filesystem", "Exported File Share", "Exporting Port",
            "DMTF Reserved", "Vendor Specific" }]
    uint16 ElementType[],
    [IN, Description ( "An array of strings that specify the particular "Other"
                      element(s) when the ElementType property above includes
                      the ElementType value of 1 (i.e., "Other"). Each
                      string within this array identifies a separate "Other"
                      element and duplicate string values are NOT allowed.
                      This property should be set to NULL when the
                      ElementType property does not include the value of
                      1."))]
    string OtherElementTypeDescriptions[],
    [IN, Description(The manifest collection that contains the manifests which list
                      the metrics that should be returned for each element
                      type)]
    CIM_FileSystemStatisticsManifestCollection REF ManifestCollection,

```

```
[IN, Description("Specifies the format of the Statistics output parameter")
ValueMap { "2" } ,
Values ( "CSV" )]
uint16 StatisticsFormat,
[OUT, Description(The statistics for all the elements as determined by the
                    Elements and ManifestCollection parameters)]
string Statistics[] );
```

Error returns are:

```
{ "Job Completed with No Error", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter",
"Method Reserved", "Method Parameters Checked - Job Started", "Element Not Supported", "Statistics
Format Not Supported", "Method Reserved", "Vendor Specific"}
```

NOTE In this version of the standard, Job Control is not supported for the GetStatisticsCollection method. This method should always return NULL for the Job parameter.

If the ElementTypes[] array is empty, then no data is returned. If the ElementTypes[] array is NULL, then the ElementTypes[] parameter is ignored and all data specified in the manifest collection is returned.

If the manifest collection is empty, then no data is returned. If the manifest collection parameter is NULL, then the default manifest collection is used. (Note: In SMI-S, a default manifest collection shall exist if the GetStatisticalCollection method is supported).

NOTE The ElementTypes[] and ManifestCollection parameters may identify different sets of element types. The effect of this will be for the implementation to return statistics for the element types that are in both lists (that is, the intersection of the two lists). This intersection could be empty. In this case, no data will be returned.

For the current version of SMI-S, the only recognized value for StatisticsFormat is "CSV". The method may support other values, but they are not specified by SMI-S (i.e., they would be vendor specific).

Given a client has an inventory of the metered objects with Statistics InstanceIDs that may be used to correlate with the FileSystemStatisticalData instances, a simple CSV format is sufficient and the most efficient human-readable format for transferring bulk statistics. More specifically, the following rules constrain that format and define the content of the String[] Statistics output parameter to the Get Statistics Collection() method:

- The Statistics[] array may contain multiple statistics records per array entry. In such cases, the total length of the concatenated record strings will not exceed 64K bytes. And a single statistics record will not span Array entries.
- There shall be exactly one statistics record per line in the bulk Statistics parameter. A line is terminated by:
  - a line-feed character
  - the end of a String Array Element (i.e., a statistics record cannot overlap elements of the String[] Statistics output parameter).
- Each statistics record shall contain the InstanceID of the FileSystemStatisticalData instance, the value map (number) of the ElementType of the metered object, and one value for each property that the relevant FileSystemStatisticsManifest specifies as "true".
- Each value in a record shall be separated from the next value by a Semi-colon (";"). This is to support internationalization of the CSV format. A provider creating a record in this format should not include white space between values in a record. A client reading a record it has received would ignore white-space between values.

- The InstanceID value is an opaque string that shall correspond to the InstanceID property from FileSystemStatisticalData instance.

For the convenience of client software that needs to be able to correlate InstanceIDs between different GetStatisticsCollection method invocations, the InstanceID for FileSystemStatisticalData instance shall be unique across all instances of the FileSystemStatisticalData class. It is not sufficient that InstanceID is unique across subclasses of FileSystemStatisticalData.

- The ElementType value shall be a decimal string representation of the Element Type number (e.g., "102" for Local Filesystem). The StatisticTime shall be a string representation of DateTime. All other values shall be decimal string representations of their statistical values.
- Null values shall be included in records for which a statistic is returned (specified by the manifest or by a lack of manifest for a particular element type) but there is no meaningful value available for the statistic. A NULL statistic is represented by placing a semi-colon (;) in the record without a value at the position where the value would have otherwise been included. A record in which the last statistic has a NULL value shall end in a semi-colon (;).
- The first three values in a record shall be the InstanceID, ElementType and StatisticTime values from the FileSystemStatisticalData instance. The remaining values shall be returned in the order in which they are defined by the MOF for the FileSystemStatisticsManifest class or subclass the record describes.

As an additional convention, a provider should return all the records for a particular element type in consecutive String elements, and the order of the element types should be the same as the order in which the element types were specified in the input parameter to GetStatisticsCollection().

Example output as it might be transmitted in CIM-XML. It shows records for 5 local filesystems and 5 exported file shares, assuming that 6 statistics were specified in the FileSystemStatisticsManifest instance for both local filesystems and exported file shares. The sixth statistic is unavailable for local filesystems, and the fourth statistic is unavailable for exported file shares:

```
<METHODRESPONSE NAME="GetStatisticsCollection">
  <RETURNVALUE PARAMTYPE="uint32">
    <VALUE>
      0
    </VALUE>
  </RETURNVALUE>
  <PARAMVALUE NAME="Statistics" PARAMTYPE="string">
    <VALUE.ARRAY>
      <VALUE>
        LOCALFILESYSTEMSTATS1;102;20060811133015.0000010-
          300;11111;22222;33333;44444;55555;
        LOCALFILESYSTEMSTATS2;102;20060811133015.0000020-
          300;11111;22222;33333;44444;55555;
        LOCALFILESYSTEMSTATS3;102;20060811133015.0000030-
          300;11111;22222;33333;44444;55555;
        LOCALFILESYSTEMSTATS4;102;20060811133015.0000040-
          300;11111;22222;33333;44444;55555;
        LOCALFILESYSTEMSTATS5;102;20060811133015.0000050-
          300;11111;22222;33333;44444;55555;
      </VALUE>
      <VALUE>
        EXPORTFILESHARESTATS1;103;20060811133015.0000100-
          300;11111;22222;33333;;55555;66666
      </VALUE>
    </VALUE.ARRAY>
  </PARAMVALUE>
</METHODRESPONSE>
```

```
EXPORTFILESHARESTATS2;103;20060811133015.0000110-
    300;11111;22222;33333;;55555;66666
EXPORTFILESHARESTATS3;103;20060811133015.0000120-
    300;11111;22222;33333;;55555;66666
EXPORTFILESHARESTATS4;103;20060811133015.0000130-
    300;11111;22222;33333;;55555;66666
EXPORTFILESHARESTATS5;103;20060811133015.0000140-
    300;11111;22222;33333;;55555;66666

</VALUE>
</VALUE.ARRAY>
</PARAMVALUE>
</METHODRESPONSE>
```

### 10.3.1.3 CreateManifestCollection

This extrinsic method creates a new manifest collection whose members serve as a filter for metrics retrieved through the GetStatisticsCollection method.

```
CreateManifestCollection(
    [IN, Description(The collection of statistics that will be filtered using the new
        manifest collection)]
    CIM_StatisticsCollection REF Statistics,
    [IN, Description(Client-defined name for the new manifest collection)]
    string ElementName,
    [OUT, Description(Reference to the new manifest collection)]
    CIM_FileSystemManifestCollection REF ManifestCollection );
```

Error returns are:

```
{ "Ok", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter",
    "Method Reserved", "Vendor Specific" }
```

### 10.3.1.4 AddOrModifyManifest

This is an extrinsic method that either creates or modifies a statistics manifest for this statistics service. A client supplies a manifest collection within which the new manifest collection will be placed or an existing manifest will be modified, the element type of the statistics that the manifest will filter, and a list of statistics that should be returned for that element type using the GetStatisticsCollection method.

```
AddOrModifyManifest(
    [IN, Description(Manifest collection that the manifest is or should be a member
        of)]
    CIM_FileSystemStatisticsManifestCollection REF ManifestCollection,
    [IN, Description(The element type whose statistics the manifest will filter)]
    ValueMap { "1", "102", "103", "104", "..", "0x8000.." },
    Values { "Other", "Local Filesystem", "Exported File Share", "Exporting Port",
        "DMTF Reserved", "Vendor Specific" },
    uint16 ElementType,
    [IN, Description ( "A string describing the type of element when the ElementType
        property above is set to 1 (i.e., "Other"). This
        property should be set to NULL when the ElementType
        property is any value other than 1.")]
    string OtherElementTypeDescription,
```

```
[IN, Description(The client-defined string that identifies the manifest created or
                  modified by this method)
string ElementName,
[IN, Description(The statistics that will be included by the manifest filter; that
                  is, the statistics that will be supplied through the
                  GetStatisticsCollection method)
string StatisticsList[],
[OUT, Description(The Manifest that is created or modified on the successful
                  execution of this method)]
CIM_FileSystemManifest REF Manifest );
```

Error returns are:

```
{ "Success", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter",
  "Method Reserved", "Element Not Supported", "Metric not
  supported", "ElementType Parameter Missing", "Method
  Reserved", "Vendor Specific" }
```

If the StatisticsList[] array is empty, then only InstanceID and ElementType will be returned when the manifest is referenced. If the StatisticsList[] array parameter is NULL, then all supported properties is assumed (i.e., all supported properties will be included).

NOTE This would be the FileSystemStatisticsManifest from the default manifest collection.

#### 10.3.1.5 RemoveManifests

This is an extrinsic method that removes manifests from the manifest collection.

```
RemoveManifests(
[IN, Description(Manifest collection from which the manifests will be removed)]
CIM_FileSystemStatisticsManifestCollection REF ManifestCollection,
[IN, Description(List of manifests to be removed from the manifest collection)
CIM_FileSystemStatisticsManifest REF Manifest[] );
```

Error returns are:

```
{ "Success", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid
  Parameter", "Method Reserved", "Manifest not found",
  "Method Reserved", "Vendor Specific" }
```

### 10.3.2 Intrinsic Methods of this Profile

NOTE Basic Write intrinsic methods are not specified for StatisticsCollection, HostedCollection, FileSystemStatisticalData, MemberOfCollection or ElementStatisticalData.

#### 10.3.2.1 DeleteInstance (of a FileSystemStatisticsManifestCollection)

This will delete the FileSystemStatisticsManifestCollection where IsDefault=False, the AssociatedFileSystemStatisticsManifestCollection association to the StatisticsCollection and all manifests collected by the manifest collection (and the MemberOfCollection associations to the FileSystemStatisticsManifestCollection).

#### 10.3.2.2 Association Traversal

One of the ways of retrieving statistics is through association traversal from the StatisticsCollection to the individual Statistics following the MemberOfCollection association. This shall be supported by all implementations of the Filesystem Performance Profile and would be available to clients if the provider does not support the EXEC QUERY or GetStatisticsCollection approaches.

## 10.4 Use Cases

### 10.4.1 Summary of Statistics Support by Element

Not all statistics properties are kept for all elements. Table 146 illustrates the statistics properties that are kept for each of the metered elements.

**Table 146 - Summary of Statistics Support by Element**

Statistic Property	Local Filesystem	Exported File Share	Exporting Port	Other
StatisticTime	R	R	R	R
TotalIOs	R	R	R	R
TotalBytesTransferred	R	R	R	N
ReadIOs	R	R	N	N
WriteIOs	R	R	N	N
OtherIOs	R	R	N	N
MetadataReadIOs	O	O	N	N
MetadataWriteIOs	O	O	N	N
TotalIOTimeCounter	O	O	O	N
TotalIdleTimeCounter	O	O	O	N
ReadIOTimeCounter	O	O	N	N
BytesRead	O	O	N	N
WriteIOTimeCounter	O	O	N	N
BytesWritten	O	O	N	N
MetadataBytesRead	O	O	N	N
MetadataBytesWritten	O	O	N	N
PercentDurableOpens	N	O	N	N
PercentResilientOpens	N	O	N	N
PercentPersistentOpens	N	O	N	N
AverageReadResponseTime	N	O	N	N
AverageWriteResponseTime	N	O	N	N
AverageRequestResponseTime	N	O	N	N
BytesReadPerSec	N	O	N	N
TotalBytesReceived	N	O	N	N
BytesReceivedPerSec	N	O	N	N
TotalBytesSent	N	O	N	N
BytesSentPerSec	N	O	N	N
BytesTranferredPerSec	N	O	N	N
BytesWrittenPerSec	N	O	N	N
FilesOpenedPerSec	N	O	N	N

**Table 146 - Summary of Statistics Support by Element**

Statistic Property	Local Filesystem	Exported File Share	Exporting Port	Other
TotalOpenFileCount	N	O	N	N
CurrentPendingRequests	N	O	N	N
ReadRequestsProcessedPerSec	N	O	N	N
TotalRequestsReceived	N	O	N	N
RequestsReceivedPerSec	N	O	N	N
TotalDurableHandleReopenCount	N	O	N	N
TotalFailedDurableHandleReopenCount	N	O	N	N
TotalFailedResilientHandleReopenCount	N	O	N	N
CurrentOpenFileCount	N	O	N	N
TotalResilientHandleReopenCount	N	O	N	N
TotalPersistentHandleReopenCount	N	O	N	N
TotalFailedPersistentHandleReopenCount	N	O	N	N
TreeConnectCount	N	O	N	N
WriteRequestsProcessedPerSec	N	O	N	N
TotalMetadataRequestsReceived	N	O	N	N
MetadataRequestsReceivedPerSec	N	O	N	N
AverageTimePerDataRequest	N	O	N	N
AverageBytesPerDataRequest	N	O	N	N
AverageBytesPerReadRequest	N	O	N	N
AverageBytesPerWriteRequest	N	O	N	N
AverageReadQueueLength	N	O	N	N
AverageWriteQueueLength	N	O	N	N
AverageDataQueueLength	N	O	N	N
DataBytesPerSec	N	O	N	N
DataRequestsPerSec	N	O	N	N
CurrentDataQueueLength	N	O	N	N

The legend is:

R - Required

O - Optional

N - Not specified

A complete list of definitions of the metered elements as defined by the ElementType property of FileSystemStatisticalData is below:

- ElementType = 1 (Other) - This is used by the provider to specify a filesystem-related metered element other than one explicitly declared (e.g., "Local Filesystem" below) within the list of element types supported by the Filesystem Performance Profile in this release of SMI-S. If the ElementType is "Other", then information describing the metered element should be provided in the "OtherElementTypeDescription" string property.
- ElementType = 102 (Local Filesystem) - This is a filesystem that would be a LocalFilesystem in the Filesystem Profile. It is a target for I/O operations that would include file I/O operations for storing and retrieving the contents of a file maintained by the filesystem, I/O operations directed to directories maintained by the filesystem, and other I/O operations performed to manage the filesystem and its contents.
- ElementType = 103 (Exported File Share) - This is a FileShare in the File Export Profile; it is a file share that is exported to a network.
- ElementType = 104 (Exporting Port) - This is a port through which a file share being exported can be accessed. It is a ProtocolEndPoint through which a FileShare can be accessed as described within the File Export Profile.

#### 10.4.2 Formulas and Calculations

Table 4 identifies the set of statistics that are recommended for various elements associated with filesystems. Once collected, these metrics can be further enhanced through the definition of formulas and calculations that create additional "derived" statistics.

Table 147 defines a set of such derived statistics as pertain to a calculated time interval. These calculated statistics are by no means the only possible derivations but serve as examples of commonly requested statistics.

**Table 147 - Formulas and Calculations - Calculated Statistics for a Time Interval**

New statistic	Formula
TimeInterval	delta StatisticTime
I/O rate	delta TotalIOs / TimeInterval
I/O average response time	delta TotalIOTimeCounter / delta TotalIOs
Read average response time	delta ReadIOTimeCounter / delta ReadIOs
Write average response time	delta WriteIOTimeCounter / delta WriteIOs
Average Read Size	delta BytesRead / delta ReadIOs
Average Write Size	delta BytesWritten / delta WriteIOs
% Read	100 * (delta ReadIOs / delta TotalIOs)
% Write	100 * (delta WriteIOs / delta TotalIOs)

#### 10.4.3 Filesystem Performance Supported Capabilities Patterns

The Filesystem Performance Profile in this release of SMI-S formally recognizes the Capabilities patterns summarized in Table 148.



**Table 148 - Filesystem Performance Profile Supported Capabilities Patterns**

Element Supported	SynchronousMethods Supported	AsynchronousMethods Supported
Any (at least one)	NULL	NULL
Any (at least one)	Neither GetStatisticsCollection nor Exec Query	NULL
Any (at least one)	GetStatisticsCollection	NULL
Any (at least one)	Any	NULL
Any (at least one)	Exec Query	NULL
Any (at least one)	GetStatisticsCollection, Exec Query	NULL
Any (at least one)	"Manifest Creation", "Manifest Modification", and "Manifest Removal"	NULL
Any (at least one)	"Indications", "Query Collection"	NULL

An implementation will support GetStatisticsCollection, Query, GetStatisticsCollection and Query or neither. But if the implementation supports GetStatisticsCollection, it shall support Synchronous execution.

If manifest collections are supported, then ALL three methods shall be supported (creation, modification and removal).

## 10.5 CIM Elements

Table 149 describes the CIM elements for Filesystem Performance.

**Table 149 - CIM Elements for Filesystem Performance**

Element Name	Requirement	Description
10.5.1 CIM_AssociatedFileSystemStatisticsManifestCollection (Client defined collection)	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported. This is an association between the StatisticsCollection and a client defined manifest collection.
10.5.2 CIM_AssociatedFileSystemStatisticsManifestCollection (Provider defined collection)	Mandatory	This is an association between the StatisticsCollection and a provider supplied (predefined) manifest collection that defines the filesystem statistics properties supported by the profile implementation.
10.5.3 CIM_ElementCapabilities	Mandatory	This associates the FileSystemStatisticsCapabilities to the FileSystemStatisticsService.
10.5.4 CIM_ElementStatisticalData (Exported File Share Stats)	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "103" (Exported File Share statistics support).  This associates a FileSystemStatisticalData instance to the exported File Share for which the statistics are collected.

**Table 149 - CIM Elements for Filesystem Performance**

Element Name	Requirement	Description
10.5.5 CIM_ElementStatisticalData (Exporting Port Stats)	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "104" (Exporting Port statistics support).  This associates a FileSystemStatisticalData instance to the exporting Port for which the statistics are collected.
10.5.6 CIM_ElementStatisticalData (Local Filesystem Stats)	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "102" (Local Filesystem statistics support).  This associates a FileSystemStatisticalData instance to the local filesystem for which the statistics are collected.
10.5.7 CIM_ElementStatisticalData (OTHER Element Type Stats)	Conditional	Conditional requirement: This is mandatory if CIM_FileSystemStatisticsCapabilities.ElementTypesSupported = "1" (OTHER element type statistics support).  This associates a FileSystemStatisticalData instance to a provider-specified other element for which the statistics are collected.
10.5.8 CIM_FileSystemStatisticalData	Mandatory	The CIM_FileSystemStatisticalData class defines the filesystem statistics properties that may be kept for a metered element of a system that provides filesystem support (such as a NAS Head or a Self-Contained NAS). Examples of such metered elements include LocalFileSystem (Local Filesystem) and FileShare (Exported File Share).
10.5.9 CIM_FileSystemStatisticsCapabilities	Mandatory	This defines the statistics capabilities supported by the implementation of the profile.
10.5.10 CIM_FileSystemStatisticsManifest (Client Defined)	Conditional	Conditional requirement: Clients can modify manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported. An instance of this class defines the filesystem statistics properties of interest to the client for one element type.
10.5.11 CIM_FileSystemStatisticsManifest (Provider Support)	Mandatory	An instance of this class defines the filesystem statistics properties supported by the profile implementation for one element type.
10.5.12 CIM_FileSystemStatisticsManifestCollection (Client Defined)	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported. An instance of this class defines one client defined collection of filesystem statistics manifests (one manifest for each element type).
10.5.13 CIM_FileSystemStatisticsManifestCollection (Provider Defined)	Mandatory	An instance of this class defines the predefined collection of default filesystem statistics manifests (one manifest for each element type).
10.5.14 CIM_FileSystemStatisticsService	Mandatory	This is a Service that provides (optional) services of bulk statistics retrieval and manifest set manipulation methods.

**Table 149 - CIM Elements for Filesystem Performance**

Element Name	Requirement	Description
10.5.15 CIM_HostedCollection (Client Defined)	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported. This would associate a client defined FileSystemStatisticsManifestCollection to the top level system for the profile (e.g., a NAS Head).
10.5.16 CIM_HostedCollection (Default)	Mandatory	This would associate a default FileSystemStatisticsManifestCollection to the top level system for the profile (e.g., a NAS Head).
10.5.17 CIM_HostedCollection (Provider Supplied)	Mandatory	This would associate the StatisticsCollection to the top level system for the profile (e.g., NAS Head).
10.5.18 CIM_HostedService	Mandatory	This associates the FileSystemStatisticsService to the ComputerSystem that hosts it.
10.5.19 CIM_MemberOfCollection (Member of client defined collection)	Conditional	Conditional requirement: Clients can modify manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported. This would associate Manifests to client-defined manifest collections.
10.5.20 CIM_MemberOfCollection (Member of predefined collection)	Mandatory	This would associate predefined Manifests to the default manifest collection.
10.5.21 CIM_MemberOfCollection (Member of statistics collection)	Mandatory	This would associate all filesystem statistics instances to the StatisticsCollection.
10.5.22 CIM_StatisticsCollection	Mandatory	This would be a collection point for all filesystem statistics that are kept for metered elements of a system that provides filesystem support (such as a NAS Head or a Self-Contained NAS).

#### 10.5.1 CIM\_AssociatedFileSystemStatisticsManifestCollection (Client defined collection)

The CIM\_AssociatedFileSystemStatisticsManifestCollection associates an instance of a CIM\_FileSystemStatisticsManifestCollection to the instance of CIM\_StatisticsCollection to which it applies. Client defined manifest collections identify the Manifests (statistic properties) for retrieval of filesystem statistics.

CIM\_AssociatedFileSystemStatisticsManifestCollection is not subclassed from anything.

There will be one instance of the CIM\_AssociatedFileSystemStatisticsManifestCollection class, for each client defined manifest collection that has been created.

Created By: Extrinsic: CreateManifestCollection

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by CIM\_FileSystemStatisticsCapabilities.SynchronousMethodsSupported.

Table 150 describes class CIM\_AssociatedFileSystemStatisticsManifestCollection (Client defined collection).

**Table 150 - SMI Referenced Properties/Methods for CIM\_AssociatedFileSystemStatisticsManifestCollection (Client defined collection)**

Properties	Flags	Requirement	Description & Notes
Statistics		Mandatory	The StatisticsCollection to which the manifest collection applies.
ManifestCollection		Mandatory	A client defined manifest collection.

### 10.5.2 CIM\_AssociatedFileSystemStatisticsManifestCollection (Provider defined collection)

The CIM\_AssociatedFileSystemStatisticsManifestCollection associates an instance of a CIM\_FileSystemStatisticsManifestCollection to the instance of CIM\_StatisticsCollection to which it applies. The default manifest collection defines the CIM\_FileSystemStatisticalData properties that are supported by the profile implementation.

CIM\_AssociatedFileSystemStatisticsManifestCollection is not subclassed from anything.

One instance of the CIM\_AssociatedFileSystemStatisticsManifestCollection shall exist for the default manifest collection if the Filesystem Performance Profile is implemented.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 151 describes class CIM\_AssociatedFileSystemStatisticsManifestCollection (Provider defined collection).

**Table 151 - SMI Referenced Properties/Methods for CIM\_AssociatedFileSystemStatisticsManifestCollection (Provider defined collection)**

Properties	Flags	Requirement	Description & Notes
Statistics		Mandatory	The StatisticsCollection to which the manifest collection applies.
ManifestCollection		Mandatory	The default manifest collection.

### 10.5.3 CIM\_ElementCapabilities

CIM\_ElementCapabilities represents the association between ManagedElements (i.e., CIM\_FileSystemStatisticsService) and their Capabilities (e.g., CIM\_FileSystemStatisticsCapabilities). Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM\_ElementCapabilities association for the referenced instance of Capabilities. ElementCapabilities describes the existence requirements and context for the referenced instance of ManagedElement. Specifically, the ManagedElement shall exist and provides the context for the Capabilities.

CIM\_ElementCapabilities is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 152 describes class CIM\_ElementCapabilities.

**Table 152 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element (FileSystemStatisticsService).
Capabilities		Mandatory	The Capabilities instance associated with the FileSystemStatisticsService.

#### 10.5.4 CIM\_ElementStatisticalData (Exported File Share Stats)

CIM\_ElementStatisticalData is an association that relates an exported File Share to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM\_ElementStatisticalData association for the referenced instance of FileSystemStatistics. ElementStatisticalData describes the existence requirements and context for the FileSystemStatistics, relative to a specific File Share that is being exported.

CIM\_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is mandatory if CIM\_FileSystemStatisticsCapabilities.ElementTypesSupported = "103" (Exported File Share statistics support).

Table 153 describes class CIM\_ElementStatisticalData (Exported File Share Stats).

**Table 153 - SMI Referenced Properties/Methods for CIM\_ElementStatisticalData (Exported File Share Stats)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to an exported FileShare for which the Statistics apply.
Stats		Mandatory	A reference to the FileSystemStatisticalData that hold the statistics for the exported FileShare.

#### 10.5.5 CIM\_ElementStatisticalData (Exporting Port Stats)

CIM\_ElementStatisticalData is an association that relates an exporting Port to its statistics. This exporting Port is a ProtoEndPoint through which a file share that is being exported can be accessed. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM\_ElementStatisticalData association for the referenced instance of FileSystemStatistics. ElementStatisticalData describes the existence requirements and context for the FileSystemStatistics, relative to a specific exporting Port.

CIM\_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is mandatory if CIM\_FileSystemStatisticsCapabilities.ElementTypesSupported = "104" (Exporting Port statistics support).

Table 154 describes class CIM\_ElementStatisticalData (Exporting Port Stats).

**Table 154 - SMI Referenced Properties/Methods for CIM\_ElementStatisticalData (Exporting Port Stats)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to a ProtocolEndPoint port for which the Statistics apply.
Stats		Mandatory	A reference to the FileSystemStatisticalData that hold the statistics for the exporting Port.

### 10.5.6 CIM\_ElementStatisticalData (Local Filesystem Stats)

CIM\_ElementStatisticalData is an association that relates a local filesystem to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM\_ElementStatisticalData association for the referenced instance of FileSystemStatistics. ElementStatisticalData describes the existence requirements and context for the FileSystemStatistics, relative to a specific local filesystem.

CIM\_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is mandatory if CIM\_FileSystemStatisticsCapabilities.ElementTypesSupported = "102" (Local Filesystem statistics support).

Table 155 describes class CIM\_ElementStatisticalData (Local Filesystem Stats).

**Table 155 - SMI Referenced Properties/Methods for CIM\_ElementStatisticalData (Local Filesystem Stats)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to a LocalFileSystem for which the Statistics apply.
Stats		Mandatory	A reference to the FileSystemStatisticalData that hold the statistics for the local filesystem.

### 10.5.7 CIM\_ElementStatisticalData (OTHER Element Type Stats)

CIM\_ElementStatisticalData is an association that relates a provider-specified other element to its statistics. This other element is a filesystem-related managed element whose type is not explicitly declared within the list of ElementTypesSupported values defined within CIM\_FileSystemStatisticsCapabilities. Information describing the metered element in this case should also be provided in the CIM\_FileSystemStatisticalData.OtherElementTypeDescription property for the referenced instance of the FileSystemStatistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM\_ElementStatisticalData association for the referenced instance of FileSystemStatistics. ElementStatisticalData describes the existence requirements and context for the FileSystemStatistics, relative to the specific metered element.

CIM\_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is mandatory if CIM\_FileSystemStatisticsCapabilities.ElementTypesSupported = "1" (OTHER element type statistics support).

Table 156 describes class CIM\_ElementStatisticalData (OTHER Element Type Stats).

**Table 156 - SMI Referenced Properties/Methods for CIM\_ElementStatisticalData (OTHER Element Type Stats)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A reference to the provider-specified managed element for which the Statistics apply.
Stats		Mandatory	A reference to the FileSystemStatisticalData that hold the statistics for the provider-specified managed element.

### 10.5.8 CIM\_FileSystemStatisticalData

CIM\_FileSystemStorageStatisticalData is subclassed from CIM\_StatisticalData.

Instances of this class will exist for each of the metered elements if the "ElementTypesSupported" property of the CIM\_FileSystemStatisticsCapabilities indicates that the metered element is supported. For example, if "Local Filesystem" is identified in the "ElementTypesSupported" property, then this indicates support for metering of the local filesystem.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 157 describes class CIM\_FileSystemStatisticalData.

**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	The InstanceID for a FileSystemStatisticalData instance shall be unique across all instances of the FileSystemStatisticalData class.
StatisticTime		Mandatory	The time that the most recent measurement was taken, relative to the object (managed element) where the statistics were collected. (Time stamp in CIM 2.2 specification format).
ElementType		Mandatory	Defines the role that the metered element (object) played for which this statistics record was collected. This value is required AND the current version of SMI-S specifies the following values:  ValueMap {"1", "102", "103", "104"}  Values { "Other", "Local Filesystem", "Exported File Share", "Exporting Port"}.
OtherElementTypeDescription		Mandatory	A string describing the type of element when the ElementType property of this class (or any of its subclasses) is set to 1 (i.e., "Other"). This property should be set to NULL when the ElementType property is any value other than 1.
TotalIOs		Mandatory	The cumulative count of file I/O operations for the object, including metadata I/O operations.

**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
TotalBytesTransferred		Conditional	Conditional requirement: This property is required if the ElementType is 102, 103, or 104. The cumulative count of bytes transferred for all of the file I/O operations as defined in "TotalIOs" above.  Note: This is not specified for the "Other" ElementType.
ReadIOs		Conditional	Conditional requirement: This property is required if the ElementType is 102 or 103. The cumulative count of file I/O operations that were directed to the object and that performed a transfer of data from the file contents.  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
WriteIOs		Conditional	Conditional requirement: This property is required if the ElementType is 102 or 103. The cumulative count of file I/O operations that were directed to the object and that performed a transfer of data to the file contents.  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
OtherIOs		Conditional	Conditional requirement: This property is required if the ElementType is 102 or 103. The cumulative count of file I/O operations that were directed to the object and that did not perform a transfer of data either to or from the file contents. This count excludes metadata I/O operations (both read and write). File "open", "close", and "lock" I/O operations are examples of an "OtherIO" I/O operation.  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
MetadataReadIOs		Optional	The cumulative count of file I/O operations that were directed to the object and that performed a read transfer of metadata. "Get Attributes" and "Read Directory" I/O operations are examples of a Metadata read I/O operation.  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
MetadataWriteIOs		Optional	The cumulative count of file I/O operations that were directed to the object and that performed a write transfer of metadata. "Set Attributes" I/O operations are an example of a Metadata write I/O operation.  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
TotalIOTimeCounter		Optional	The cumulative elapsed I/O operation time (number of ClockTickIntervals) for all file I/O operations as defined in "TotalIOs" above. The I/O operation response time is added to this counter at the completion of each measured I/O operation using ClockTickInterval units. The TotalIOTimeCounter value can be divided by the total number of I/O operations (TotalIOs) to obtain an I/O operation average response time.  Note: This is not specified for the "Other" ElementType.
TotalIdleTimeCounter		Optional	The cumulative elapsed idle time using ClockTickInterval units. That is, the cumulative number of ClockTickIntervals for all idle time within the object, with "idle time" being that time during which no I/O operations were being processed by the object.  Note: This is not specified for the "Other" ElementType.



**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
ReadIOTimeCounter		Optional	The cumulative elapsed I/O operation time for all Read I/O operations (that is, the cumulative elapsed time for all Read I/O operations as defined in "ReadIOs" above).  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
BytesRead		Optional	The cumulative count of bytes read (that is, the cumulative count of bytes transferred by all Read I/O operations as defined in "ReadIOs" above).  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
WriteIOTimeCounter		Optional	The cumulative elapsed I/O operation time for all Write I/O operations (that is, the cumulative elapsed time for all Write I/O operations as defined in "WriteIOs" above).  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
BytesWritten		Optional	The cumulative count of bytes written (that is, the cumulative count of bytes transferred by all Write I/O operations as defined in "WriteIOs" above).  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
MetadataBytesRead		Optional	The cumulative count of metadata bytes read (that is, the cumulative count of bytes transferred by all Metadata read I/O operations as defined in "MetadataReadIOs" above).  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
MetadataBytesWritten		Optional	The cumulative count of metadata bytes written (that is, the cumulative count of bytes transferred by all Metadata write I/O operations as defined in "MetadataWriteIOs" above).  Note: This is not specified for the "Exporting Port" and the "Other" ElementTypes.
PercentDurableOpens		Optional	The percentage of total opens for which clients requested durability.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
PercentResilientOpens		Optional	The percentage of total opens for which clients requested resiliency.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
PercentPersistentOpens		Optional	The percentage of total handles for which clients requested persistency.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageReadResponseTime		Optional	The average number of seconds that elapse between the time at which a read request to this share is received and the time at which the SMB2 File Server sends the corresponding response.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.

**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
AverageWriteResponseTime		Optional	The average number of seconds that elapse between the time at which a write request to this share is received and the time at which the SMB2 File Server sends the corresponding response.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageRequestResponseTime		Optional	The average number of seconds that elapse between the time at which the SMB2 File Server receives a request for this share and the time at which the SMB2 File Server sends the corresponding response.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
BytesReadPerSec		Optional	The rate, in seconds, at which data is being read from this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalBytesReceived		Optional	The number of bytes that have been received for requests related to this share. This value includes application data as well as SMB2 protocol data (such as packet headers).  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
BytesReceivedPerSec		Optional	The rate at which bytes are being received for requests related to this share. This value includes application data as well as SMB2 protocol data (such as packet headers).  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalBytesSent		Optional	The number of bytes that have been sent by the SMB2 File Server related to this share to its clients since the server started. This value includes both data bytes and protocol bytes.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
BytesSentPerSec		Optional	The rate, in seconds, at which bytes are being sent from the SMB2 File Server related to this share to its clients. This value includes both data bytes and protocol bytes.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
BytesTranferredPerSec		Optional	The sum of bytes transferred/sec related to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
BytesWrittenPerSec		Optional	The rate, in seconds, at which data is being written to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
FilesOpenedPerSec		Optional	The rate, in seconds, at which files are being opened for the SMB2 File Server's clients on this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.

**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
TotalOpenFileCount		Optional	The number of files that have been opened by the SMB2 File Server on behalf of its clients on this share since the server started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
CurrentPendingRequests		Optional	The number of requests related to this share that are waiting to be processed by the SMB2 File Server.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
ReadRequestsProcessedPerSec		Optional	Read requests processed/sec related to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalRequestsReceived		Optional	The number of requests that have been received for this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
RequestsReceivedPerSec		Optional	The rate at which requests are being received for this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalDurableHandleReopenCount		Optional	The number of durable opens on this share that have been recovered after a temporary network disconnect since the SMB2 File Server started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalFailedDurableHandleReopenCount		Optional	The number of durable opens on this share that could not be recovered after a temporary network disconnect since the SMB2 File Server Started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalFailedResilientHandleReopenCount		Optional	The number of resilient opens on this share that could not be recovered after a temporary network disconnect since the SMB2 File Server Started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
CurrentOpenFileCount		Optional	The number of file handles that are currently open in this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalResilientHandleReopenCount		Optional	The number of resilient opens on this share that have been recovered after a temporary network disconnect since the SMB2 File Server started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalPersistentHandleReopenCount		Optional	The number of persistent opens on this share that have been recovered after a temporary network disconnect since the SMB2 File Server started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.

**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
TotalFailedPersistentHandleReopenCount		Optional	The number of persistent opens on this share that could not be recovered after a temporary network disconnect since the SMB2 File Server Started.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TreeConnectCount		Optional	The number of tree connects to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
WriteRequestsProcessedPerSec		Optional	Write requests processed/sec related to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
TotalMetadataRequestsReceived		Optional	The total number of metadata requests received related to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
MetadataRequestsReceivedPerSec		Optional	The rate, in seconds, at which metadata requests are being sent to this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageTimePerDataRequest		Optional	The average number of seconds that elapse between the time at which a read or write request to this share is received and the time at which the SMB2 File Server processes the request.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageBytesPerDataRequest		Optional	The average number of bytes per read or write request.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageBytesPerReadRequest		Optional	The average number of bytes per read request.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageBytesPerWriteRequest		Optional	The average number of bytes per write request.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageReadQueueLength		Optional	The average number of read requests that were queued for this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageWriteQueueLength		Optional	The average number of write requests that were queued for this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
AverageDataQueueLength		Optional	The average number of read and write requests that were queued for this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.

**Table 157 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticalData**

Properties	Flags	Requirement	Description & Notes
DataBytesPerSec		Optional	The rate, in seconds, at which data is being written to or read from this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
DataRequestsPerSec		Optional	The rate, in seconds, at which read or write requests are received for this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.
CurrentDataQueueLength		Optional	The current number of read or write requests outstanding on this share.  Note: This is not specified for the "Local File System", "Exporting Port" and the "Other" ElementTypes.

**10.5.9 CIM\_FileSystemStatisticsCapabilities**

An instance of the CIM\_FileSystemStatisticsCapabilities class defines the specific support provided with the filesystem statistics implementation. Note: There would be zero or one instance of this class in a profile. There would be none if the profile did not support the Filesystem Performance Profile. There would be exactly one instance if the profile did support the Filesystem Performance Profile.

CIM\_FileSystemStatisticsCapabilities class is subclassed from CIM\_Capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 158 describes class CIM\_FileSystemStatisticsCapabilities.

**Table 158 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
ElementTypesSupported		Mandatory	ValueMap { "1", "102", "103", "104"}, Values {"Other", "Local Filesystem", "Exported File Share", "Exporting Port"}.
SynchronousMethodsSupported		Mandatory	This property is mandatory, but the array may be empty. ValueMap { "2", "3", "4", "5", "6", "7", "8"}, Values {"Exec Query", "Indications", "QueryCollection", "GetStatisticsCollection", "Manifest Creation", "Manifest Modification", "Manifest Removal"}.

**Table 158 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsCapabilities**

Properties	Flags	Requirement	Description & Notes
AsynchronousMethodsSupported		Optional	Not supported in current version of SMI-S.
ClockTickInterval		Mandatory	<p>An internal clocking interval for all timers in the subsystem, measured in microseconds (Unit of measure in the timers, measured in microseconds).</p> <p>Time counters are monotonically increasing counters that contain "ticks". Each tick represents one ClockTickInterval. If ClockTickInterval contained a value of 32 then each time counter tick would represent 32 microseconds.</p>

**10.5.10 CIM\_FileSystemStatisticsManifest (Client Defined)**

The CIM\_FileSystemStatisticsManifest class is a Concrete class that defines the CIM\_FileSystemStorageStatisticalData properties that should be returned on a GetStatisticsCollection request.

CIM\_FileSystemStatisticsManifest is subclassed from CIM\_ManagedElement.

In order for a client defined instance of the CIM\_FileSystemStatisticsManifest class to exist, all of the manifest collection manipulation functions shall be identified in the "SynchronousMethodsSupported" property of the CIM\_FileSystemStatisticsCapabilities (FileSystemStatisticsCapabilities.SynchronousMethodsSupported = "6") instance, AND a client must have created at least ONE instance of CIM\_FileSystemStatisticsManifestCollection.

Created By: Extrinsic: AddOrModifyManifest

Modified By: Extrinsic: AddOrModifyManifest

Deleted By: Extrinsic: RemoveManifests

Requirement: Clients can modify manifests as identified by CIM\_FileSystemStatisticsCapabilities.SynchronousMethodsSupported.

Table 159 describes class CIM\_FileSystemStatisticsManifest (Client Defined).

**Table 159 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifest (Client Defined)**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A Client defined string that identifies the manifest.
InstanceID		Mandatory	The instance Identification. Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class.
ElementType		Mandatory	<p>This value is required AND the current version of SMI-S specifies the following values:</p> <p>ValueMap {"1", "102", "103", "104"}</p> <p>Values { "Other", "Local Filesystem", "Exported File Share", "Exporting Port"}.</p>
OtherElementTypeDescription		Mandatory	A string describing the type of element when the ElementType property of this class (or any of its subclasses) is set to 1 (i.e., "Other"). This property should be set to NULL when the ElementType property is any value other than 1.

**Table 159 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifest (Client Defined)**

Properties	Flags	Requirement	Description & Notes
IncludeStatisticTime		Mandatory	
IncludeTotalIOs		Mandatory	
IncludeTotalBytesTransferred		Mandatory	
IncludeReadIOs		Mandatory	
IncludeWriteIOs		Mandatory	
IncludeOtherIOs		Mandatory	
IncludeMetadataReadIOs		Mandatory	
IncludeMetadataWriteIOs		Mandatory	
IncludeTotalIOTimeCounter		Mandatory	
IncludeTotalIdleTimeCounter		Mandatory	
IncludeReadIOTimeCounter		Mandatory	
IncludeBytesRead		Mandatory	
IncludeWriteIOTimeCounter		Mandatory	
IncludeBytesWritten		Mandatory	
IncludeMetadataBytesRead		Mandatory	
IncludeMetadataBytesWritten		Mandatory	
IncludePercentDurableOpens		Mandatory	
IncludePercentResilientOpens		Mandatory	
IncludePercentPersistentOpens		Mandatory	
IncludeAverageReadResponseTime		Mandatory	
IncludeAverageWriteResponseTime		Mandatory	
IncludeAverageRequestResponseTime		Mandatory	
IncludeBytesReadPerSec		Mandatory	
IncludeTotalBytesReceived		Mandatory	
IncludeBytesReceivedPerSec		Mandatory	
IncludeTotalBytesSent		Mandatory	
IncludeBytesSentPerSec		Mandatory	
IncludeBytesTranferredPerSec		Mandatory	
IncludeBytesWrittenPerSec		Mandatory	
IncludeFilesOpenedPerSec		Mandatory	
IncludeTotalOpenFileCount		Mandatory	
IncludeCurrentPendingRequests		Mandatory	
IncludeReadRequestsProcessedPerSec		Mandatory	
IncludeTotalRequestsReceived		Mandatory	
IncludeRequestsReceivedPerSec		Mandatory	

**Table 159 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifest (Client Defined)**

Properties	Flags	Requirement	Description & Notes
IncludeTotalDurableHandleReopenCount		Mandatory	
IncludeTotalFailedDurableHandleReopenCount		Mandatory	
IncludeTotalFailedResilientHandleReopenCount		Mandatory	
IncludeCurrentOpenFileCount		Mandatory	
IncludeTotalResilientHandleReopenCount		Mandatory	
IncludeTotalPersistentHandleReopenCount		Mandatory	
IncludeTotalFailedPersistentHandleReopenCount		Mandatory	
IncludeTreeConnectCount		Mandatory	
IncludeWriteRequestsProcessedPerSec		Mandatory	
IncludeTotalMetadataRequestsReceived		Mandatory	
IncludeMetadataRequestsReceivedPerSec		Mandatory	
IncludeAverageTimePerDataRequest		Mandatory	
IncludeAverageBytesPerDataRequest		Mandatory	
IncludeAverageBytesPerReadRequest		Mandatory	
IncludeAverageBytesPerWriteRequest		Mandatory	
IncludeAverageReadQueueLength		Mandatory	
IncludeAverageWriteQueueLength		Mandatory	
IncludeAverageDataQueueLength		Mandatory	
IncludeDataBytesPerSec		Mandatory	
IncludeDataRequestsPerSec		Mandatory	
IncludeCurrentDataQueueLength		Mandatory	

**10.5.11 CIM\_FileSystemStatisticsManifest (Provider Support)**

The CIM\_FileSystemStatisticsManifest class is a Concrete class that defines the CIM\_FileSystemStatisticalData properties that are supported by the Provider. These Manifests are established by the Provider for the default manifest collection.

CIM\_FileSystemStatisticsManifest is subclassed from CIM\_ManagedElement.

At least one Provider supplied instance of the CIM\_FileSystemStatisticsManifest class shall exist, if the Filesystem Performance Profile is supported.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory



Table 160 describes class CIM\_FileSystemStatisticsManifest (Provider Support).

**Table 160 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifest (Provider Support)**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A Provider defined string that identifies the manifest in the context of the Default Manifest Collection.
InstanceID		Mandatory	The instance Identification. Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class.
ElementType		Mandatory	This value is required AND the current version of SMI-S specifies the following values:  ValueMap {"1", "102", "103", "104"}  Values { "Other", "Local Filesystem", "Exported File Share", "Exporting Port"}.
OtherElementTypeDescription		Mandatory	A string describing the type of element when the ElementType property of this class (or any of its subclasses) is set to 1 (i.e., "Other"). This property should be set to NULL when the ElementType property is any value other than 1.
IncludeStatisticTime		Mandatory	
IncludeTotalIOs		Mandatory	
IncludeTotalBytesTransferred		Mandatory	
IncludeReadIOs		Mandatory	
IncludeWriteIOs		Mandatory	
IncludeOtherIOs		Mandatory	
IncludeMetadataReadIOs		Mandatory	
IncludeMetadataWriteIOs		Mandatory	
IncludeTotalIOTimeCounter		Mandatory	
IncludeTotalIdleTimeCounter		Mandatory	
IncludeReadIOTimeCounter		Mandatory	
IncludeBytesRead		Mandatory	
IncludeWriteIOTimeCounter		Mandatory	
IncludeBytesWritten		Mandatory	
IncludeMetadataBytesRead		Mandatory	
IncludeMetadataBytesWritten		Mandatory	
IncludePercentDurableOpens		Mandatory	
IncludePercentResilientOpens		Mandatory	
IncludePercentPersistentOpens		Mandatory	
IncludeAverageReadResponseTime		Mandatory	
IncludeAverageWriteResponseTime		Mandatory	
IncludeAverageRequestResponseTime		Mandatory	

**Table 160 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifest (Provider Support)**

Properties	Flags	Requirement	Description & Notes
IncludeBytesReadPerSec		Mandatory	
IncludeTotalBytesReceived		Mandatory	
IncludeBytesReceivedPerSec		Mandatory	
IncludeTotalBytesSent		Mandatory	
IncludeBytesSentPerSec		Mandatory	
IncludeBytesTranferredPerSec		Mandatory	
IncludeBytesWrittenPerSec		Mandatory	
IncludeFilesOpenedPerSec		Mandatory	
IncludeTotalOpenFileCount		Mandatory	
IncludeCurrentPendingRequests		Mandatory	
IncludeReadRequestsProcessedPerSec		Mandatory	
IncludeTotalRequestsReceived		Mandatory	
IncludeRequestsReceivedPerSec		Mandatory	
IncludeTotalDurableHandleReopenCount		Mandatory	
IncludeTotalFailedDurableHandleReopenCount		Mandatory	
IncludeTotalFailedResilientHandleReopenCount		Mandatory	
IncludeCurrentOpenFileCount		Mandatory	
IncludeTotalResilientHandleReopenCount		Mandatory	
IncludeTotalPersistentHandleReopenCount		Mandatory	
IncludeTotalFailedPersistentHandleReopenCount		Mandatory	
IncludeTreeConnectCount		Mandatory	
IncludeWriteRequestsProcessedPerSec		Mandatory	
IncludeTotalMetadataRequestsReceived		Mandatory	
IncludeMetadataRequestsReceivedPerSec		Mandatory	
IncludeAverageTimePerDataRequest		Mandatory	
IncludeAverageBytesPerDataRequest		Mandatory	
IncludeAverageBytesPerReadRequest		Mandatory	
IncludeAverageBytesPerWriteRequest		Mandatory	
IncludeAverageReadQueueLength		Mandatory	
IncludeAverageWriteQueueLength		Mandatory	
IncludeAverageDataQueueLength		Mandatory	
IncludeDataBytesPerSec		Mandatory	
IncludeDataRequestsPerSec		Mandatory	
IncludeCurrentDataQueueLength		Mandatory	

### 10.5.12 CIM\_FileSystemStatisticsManifestCollection (Client Defined)

An instance of a client defined CIM\_FileSystemStatisticsManifestCollection defines the set of Manifests to be used in the retrieval of filesystem statistics by the GetStatisticsCollection method.

CIM\_FileSystemStatisticsManifestCollection is subclassed from CIM\_SystemSpecificCollection.

In order for a client defined instance of the CIM\_FileSystemStatisticsManifestCollection class to exist, then all the manifest collection manipulation functions shall be identified in the "SynchronousMethodsSupported" property of the CIM\_FileSystemStatisticsCapabilities instance and a client must have created a Manifest Collection.

Created By: Extrinsic: CreateManifestCollection

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by  
CIM\_FileSystemStatisticsCapabilities.SynchronousMethodsSupported.

Table 161 describes class CIM\_FileSystemStatisticsManifestCollection (Client Defined).

**Table 161 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifestCollection (Client Defined)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A client defined user-friendly name for the manifest collection. It is set during creation of the Manifest Collection through the ElementName parameter of the CreateManifestCollection method.
IsDefault		Mandatory	Denotes whether or not this manifest collection is a provider defined default manifest collection. For the client defined manifest collections this is set to "false".

### 10.5.13 CIM\_FileSystemStatisticsManifestCollection (Provider Defined)

An instance of a default CIM\_FileSystemStatisticsManifestCollection defines the set of Manifests that define the properties supported for each ElementType supported for the implementation. It can also be used by clients in retrieval of Filesystem statistics by the GetStatisticsCollection method.

CIM\_FileSystemStatisticsManifestCollection is subclassed from CIM\_SystemSpecificCollection.

At least ONE CIM\_FileSystemStatisticsManifestCollection shall exist if the Filesystem Performance Profile is implemented. This would be the default manifest collection that defines the properties supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 162 describes class CIM\_FileSystemStatisticsManifestCollection (Provider Defined).

**Table 162 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsManifestCollection (Provider Defined)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	For the default manifest collection, this should be set to "DEFAULT".
IsDefault		Mandatory	Denotes whether or not this manifest collection is a provider defined default manifest collection. For the default manifest collection this is set to "true".

#### 10.5.14 CIM\_FileSystemStatisticsService

The CIM\_FileSystemStatisticsService class provides methods for statistics retrieval and Manifest Collection manipulation.

The CIM\_FileSystemStatisticsService class is subclassed from CIM\_Service.

There shall be an instance of the CIM\_FileSystemStatisticsService, if the Filesystem Performance Profile is implemented. It is not necessary to support any methods of the service, but the service shall be populated.

The methods that are supported can be determined from the SynchronousMethodsSupported and AsynchronousMethodsSupported properties of the CIM\_FileSystemStatisticsCapabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 163 describes class CIM\_FileSystemStatisticsService.

**Table 163 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsService**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	

**Table 163 - SMI Referenced Properties/Methods for CIM\_FileSystemStatisticsService**

Properties	Flags	Requirement	Description & Notes
GetStatisticsCollection()		Conditional	Conditional requirement: Clients can get statistics collections using the GetStatisticsCollection as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or Clients can get statistics collections using the GetStatisticsCollection as identified by CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported containing '5' (GetStatisticsCollection). This method retrieves all statistics kept for the profile as directed by a manifest collection.
CreateManifestCollection()		Conditional	Conditional requirement: Clients can create manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported containing '6' (Manifest Creation). This method is used to create client defined manifest collections.
AddOrModifyManifest()		Conditional	Conditional requirement: Clients can modify manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or Clients can modify manifests as identified by CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported containing '7' (Manifest Modification). This method is used to add or modify filesystem statistics manifests in a client defined manifest collection.
RemoveManifests()		Conditional	Conditional requirement: Clients can remove manifests as identified by CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or Clients can remove manifests as identified by CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or CIM_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported containing '8' (Manifest Removal). This method is used to remove a filesystem statistics manifest from a client defined manifest collection.

#### 10.5.15 CIM\_HostedCollection (Client Defined)

CIM\_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Filesystem Performance Profile, it is used to associate a client-defined FileSystemStatisticsManifestCollections to the top level Computer System.

CIM\_HostedCollection is subclassed from CIM\_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by CIM\_FileSystemStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by CIM\_FileSystemStatisticsCapabilities.AsynchronousMethodsSupported.

Table 164 describes class CIM\_HostedCollection (Client Defined).

**Table 164 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Client Defined)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	A client defined FileSystemStatisticsManifestCollection.

#### 10.5.16 CIM\_HostedCollection (Default)

CIM\_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Filesystem Performance Profile, it is used to associate the default (provider-defined) FileSystemStatisticsManifestCollection to the top level Computer System.

CIM\_HostedCollection is subclassed from CIM\_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 165 describes class CIM\_HostedCollection (Default).

**Table 165 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Default)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	The provider defined FileSystemStatisticsManifestCollection.

#### 10.5.17 CIM\_HostedCollection (Provider Supplied)

CIM\_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Filesystem Performance Profile, it is used to associate the StatisticsCollection to the top level Computer System.

CIM\_HostedCollection is subclassed from CIM\_HostedDependency.

Created By: Static  
 Modified By: Static  
 Deleted By: Static  
 Requirement: Mandatory

Table 166 describes class CIM\_HostedCollection (Provider Supplied).

**Table 166 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Provider Supplied)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	The StatisticsCollection.

#### 10.5.18 CIM\_HostedService

CIM\_HostedService is an association between a Service (CIM\_FileSystemStatisticsService) and the System (ComputerSystem) on which the functionality resides. Services are weak with respect to their hosting System. Heuristic: A Service is hosted on the System where the Filesystems or SoftwareFeatures that implement the Service are located.

CIM\_HostedService is subclassed from CIM\_HostedDependency.

Created By: Static  
 Modified By: Static  
 Deleted By: Static  
 Requirement: Mandatory

Table 167 describes class CIM\_HostedService.

**Table 167 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Service hosted on the System.

#### 10.5.19 CIM\_MemberOfCollection (Member of client defined collection)

This use of MemberOfCollection is to Collect all Manifests instances in a client-defined manifest collection.

Created By: Extrinsic: AddOrModifyManifest  
 Modified By: Static  
 Deleted By: Extrinsic: RemoveManifests  
 Requirement: Clients can modify manifests as identified by  
 CIM\_FileSystemStatisticsCapabilities.SynchronousMethodsSupported.

Table 168 describes class CIM\_MemberOfCollection (Member of client defined collection).

**Table 168 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Member of client defined collection)**

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	A client defined manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

#### 10.5.20 CIM\_MemberOfCollection (Member of predefined collection)

This use of MemberOfCollection is to Collect all Manifests instances in the default manifest collection.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 169 describes class CIM\_MemberOfCollection (Member of predefined collection).

**Table 169 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Member of predefined collection)**

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The provider defined default manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

#### 10.5.21 CIM\_MemberOfCollection (Member of statistics collection)

This use of MemberOfCollection is to collect all FileSystemStorageStatisticalData instances (in the StatisticsCollection). Each association is created as a side effect of the metered object getting created.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 170 describes class CIM\_MemberOfCollection (Member of statistics collection).

**Table 170 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Member of statistics collection)**

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The collection of all filesystem statistics data instances.
Member		Mandatory	The individual filesystem statistics data Instance that is part of the set.

#### 10.5.22 CIM\_StatisticsCollection

The CIM\_StatisticsCollection collects all filesystem statistics kept by the profile. There is one instance of the CIM\_StatisticsCollection class and all individual metered element statistics can be accessed by using association traversal (using MemberOfCollection) from the StatisticsCollection.

CIM\_StatisticsCollection is subclassed from CIM\_SystemSpecificCollection.



Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 171 describes class CIM\_StatisticsCollection.

**Table 171 - SMI Referenced Properties/Methods for CIM\_StatisticsCollection**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SampleInterval		Mandatory	Minimum recommended polling/sampling interval for a system that provides filesystem support (e.g., NAS Head or Self-Contained NAS). It is set by the provider and cannot be modified.
TimeLastSampled		Mandatory	Time statistics table by object was last updated (Time Stamp in SMI 2.2 specification format).

---

---

**EXPERIMENTAL**

---

---



---



---

## EXPERIMENTAL

### 11 Filesystem Quotas Profile

#### 11.1 Description

##### 11.1.1 Synopsis

**Profile Name:** FileSystem Quotas (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** LocalFileSystem

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 172 describes the related profiles for FileSystem Quotas.

**Table 172 - Related Profiles for FileSystem Quotas**

Profile Name	Organization	Version	Requirement	Description
Filesystem	SNIA	1.7.0	Mandatory	
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2
Job Control	SNIA	1.5.0	Optional	

##### 11.1.2 Overview

The Filesystem Quotas Profile is intended to provide management of quotas on the use of filesystem resources--raw space and inodes especially--by the common filesystem principals. User, group and tree quotas are modeled. **Trees** means **directories** (rooted directory hierarchy structures) within filesystems. Some systems allow quotas only on directories that have some special distinguishing feature, others allow them on arbitrary directories.

Quota systems work by keeping statistics-in real time-on the space used by each monitored principal/container pair e.g. a user and her home share. They then trigger events when filesystem writes cause the space used by the principal to exceed some threshold. There are four common varieties of quota thresholds:

1. Hard. Hitting a hard quota threshold causes subsequent writes by the principal to the affected container to fail. In POSIX, the error returned to the client is ENOSPC (no space). An indication is also thrown.
2. Soft. A soft threshold causes the system to issue a warning. Systems variously issue warnings via CLI, SNMP traps, pop-up windows at the user station, audit log entries, email and other proprietary methods. This profile provides required indications for this purpose.
3. Soft, with grace period. This type of quota is really a refinement of soft quotas. It functions like a soft quota until the grace period expires, at which point it begins behaving like a hard quota.
4. Monitoring. When a monitoring quota is in effect, there are no thresholds, and no warnings are issued. This type of quota is useful because the underlying system keeps track of the relevant usage statistics in real time for all quotas. A monitoring quota permits an administrator to simply issue a command to print a quota report instead of having to do a complete filesystem scan to get the usage information of interest.

In general, it is not possible to have a quota system that meets the above semantics without some level of access to the data path. More loosely coupled systems may need to relax the semantics of the hard limit,

for example, and may not actually trigger an event until a file is closed, for example. This profile allows these semantic variations.

Some systems allow "default" quotas for users, groups and/or trees. A default user quota, by way of example, is used for every user of the system who does not have a quota entry specific to them.

### 11.1.3 Tree Quotas

Tree quotas are quotas on directory trees with an identifiable root. Mounts and symlinks are not followed. In other words, a directory which contains nothing but mount points and symbolic links may satisfy a very small quota, even though the amount of data addressable by entries in the container is large.

Hard links are another matter. The policy is system-dependent, but a common behavior is that if a file or directory is hard-linked in two separate trees with separate tree quotas, the space used is charged against both quotas.

Pure tree quotas apply no matter which principal does the writing. There are some caveats however.

- Root on some systems is not constrained by quotas.
- An entity with sufficient privilege may not be constrained by quotas on some systems (e.g. a Windows user with BackupOperator privilege).

Some systems may support tree quotas only on directories with certain special characteristics. Directories may be constrained to being top-level, for example. This profile does not specify a means for determining whether a given directory may have a tree quota set on it.

### 11.1.4 User Quotas

User quotas are limits on how much space a principal with a given User ID can use. They may be either global or restricted by namespace tree, as well as by filesystem.

### 11.1.5 Group Quotas

Group quotas set limits on how much all the members of a group with a given Group ID can use in the aggregate. They are not, therefore, quotas which apply to each member of a group. This follows Unix usage. Group quotas only work on systems which have the concept of a primary group id (PGID), as the system needs to know which group to charge writes against. As NTFS does not have the concept of a primary group, it does not do group quotas. (Note: There is a primary group field that can be discovered on a file in NTFS. This is for POSIX support, however, and does not always contain anything meaningful)

Group quotas may be global or applicable to a given namespace tree and/or filesystem.

### 11.1.6 Container Boundaries

Quotas may be system-wide, or scoped to an individual filesystem. Not all systems support both of these, however, so provision is made for both. The SupportedPrincipalTypes array in the FSQuotaCapabilities class distinguishes between User, Group, Tree, User-Tree and Group-Tree quotas as follows:

- User Quotas and Group quotas are described in 11.1.4 and 11.1.5.
- A Tree Quota is a limit on the storage (or other limit such as number of inodes) used by a directory tree. This quota is not specific to a user or a group but applies to all users and groups creating files and directories within the tree.
- A User-Tree quota is a limit on the storage used by a user within a directory tree and is applied in parallel with the Tree Quota (both must be satisfied).
- A Group-Tree quota is a limit on the aggregate storage used by a group of users within a directory tree and is applied in parallel with the Tree Quota (both must be satisfied).

If a Tree is configured with a Tree Quota as well as User-Tree and Group-Tree quotas, they must all be satisfied.

### 11.1.7 Quota types

Quotas are usually limits on disk space. Some systems, however, also support limits on the number of files and/or directories.

### 11.1.8 Class design considerations

#### 11.1.8.1 New Classes

This profile uses several new classes—`FSDomainIdentity`, `FSQuotaCapabilities`, `FSQuotaReportRecord`, `FSQuotaConfigEntry`, `FSQuotaManagementService`, and `FSQuotaIndication`

##### 11.1.8.1.1 `FSDomainIdentity`

Due to the in-band nature of quota tracking, the identifier of the principal being managed needs to be small and easily optimized. Traditional systems use Unix UIDs and GIDs, which are 32-bit quantities, or SIDs which are short strings. To tie these into CIM, this new class is specified. Each instance contains a string with the UID, GID or SID, respectively, in it, and enums for the type of domain and principal.

##### 11.1.8.1.2 `FSQuotaCapabilities`

This Capabilities subclass lists the properties in a `FSQuotaConfigEntry` that are supported by the underlying system. The client shall not attempt to set any properties which are not listed as supported in the instance of this class associated to the service. It shall instead always populate unsupported properties with null.

There shall exist exactly one instance of this class per `FSQuotaManagementService` instance.

##### 11.1.8.1.3 `FSQuotaReportRecord`

When running a quota report, the underlying system generally issues a text file, each line or group of lines representing the status of a filesystem principal with respect to one quota configuration entry. There may be hundreds of thousands of these records, and they are not keyed, meaning that there is no way to go back and fetch any given one of them. Therefore `FSQuotaReportRecord` is derived from a new proposed abstract root class called `ReportRecord`, which carries the `Indication` qualifier. Note that this qualifier does not mean that these classes are subclasses of `CIM_Indication`. It's used because it's the only way, currently, to construct a class in CIM which does not require a key.

##### 11.1.8.1.4 `FSQuotaConfigEntry`

An `FSQuotaConfigEntry` instance represents a single quota entry supported by the system. For example, one `FSQuotaConfigEntry` instance may specify that on filesystem “foo,” in directory “/bar,” the user “joe” is restricted to 1GB of space, and should receive a warning at 0.97GB.

A quota entry has been defined as a complex of several classes and associations. If implementation experience turns up performance considerations related to this, this design may need to be revisited.

Many systems do not support any method of identifying individual `FSQuotaConfigEntry` instances, as they simply represent lines in a text file, and the underlying system may not care about duplicates or conflicts. However, `FSQuotaConfigEntry` instances need to be modified; this corresponds to editing the corresponding line in the file. Therefore, if the underlying system does not expose a key, one may be created by composing the `PrincipalID` property, a unique reference to the `FileSystem` or `ComputerSystem` to which the entry applies (from the association `FSQuotaAppliesToElement`), the `TreeName` property (if a tree quota), the measured quantity type (the `ResourceType` property), the quota type (`QuotaType` property), and its default status (the `Default` property). An implementation may expose the algorithm used to compose the key so that the client may decompose it, but this is not required by this version of the profile. Upon creation of a new quota instance, clients shall verify that no quota with the same key already exists. Upon modification of an instance, clients shall modify all instances whose keys match that instance key.

- **PrincipalID:** This indicates a user by the user's UID or SID, or a group by the group's GID or SID, or it can be the empty string. It is interpreted in the context of a directory service specified for the file server ComputerSystem that is associated to the FileSystem by a Dependency association.
- **InstanceID.** This property is a unique identifier for this FSQuotaConfigEntry. This property shall be unique in the presence of multiple instances of ComputerSystem and FileSystem, and multiple properties of QuotaType, Default, ResourceType and PrincipalID. It may be constructible by the client, but this profile does not specify this format.

#### 11.1.8.1.5 FSQuotaManagementService

The FSQuotaManagementService provides the interface to the underlying system for most operations which are overtly related to quotas. There shall be at most one instance of a FSQuotaManagementService for each underlying ComputerSystem.

#### 11.1.8.1.6 FSQuotaIndication

The FSQuotaIndication class provides information about threshold crossing events, meaning that a quota has just been exceeded.

#### 11.1.9 Instance Diagram

Figure 17 shows the Filesystem Quotas instance diagram.

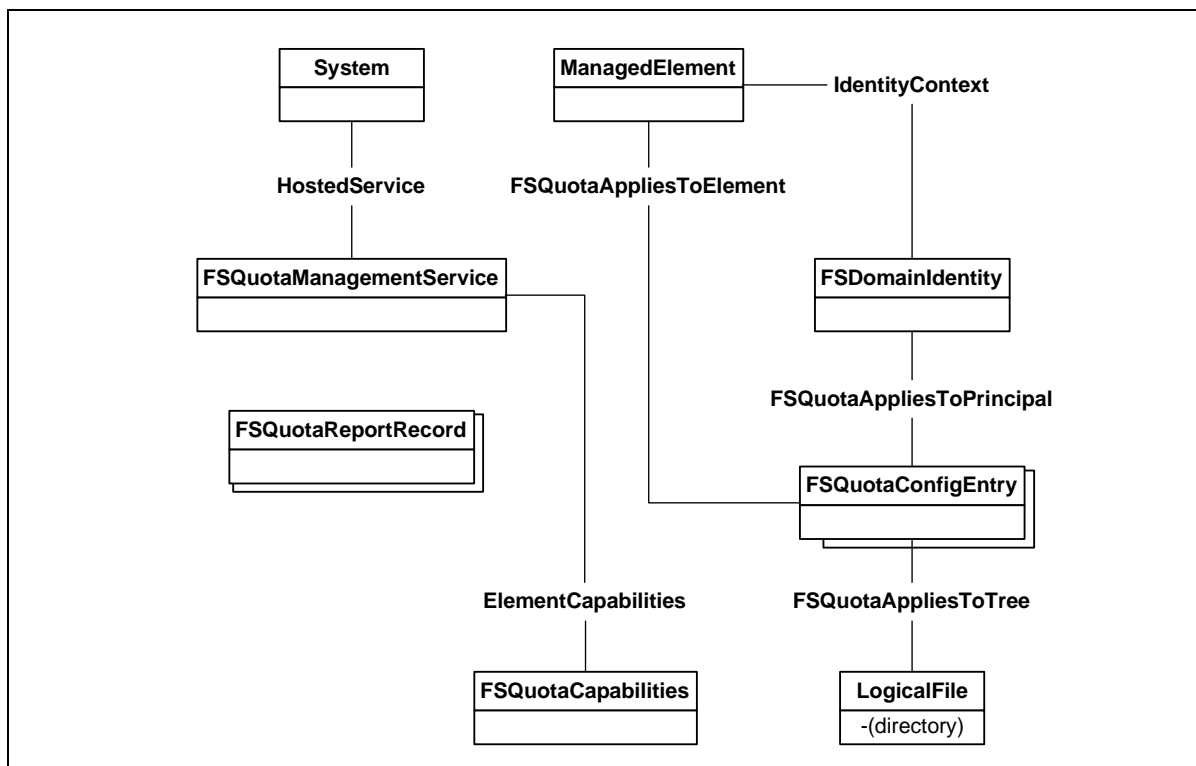


Figure 17 - Filesystem Quotas Instance Diagram

## 11.2 Health and Fault Management Considerations

Not defined in this standard

## 11.3 Methods of the Profile

All profile methods are contained in the FSQuotaManagementService.

### 11.3.1 FindQuotaEntries

```
uint32 FindQuotaEntries(
    IN string IdentityId,
    IN ManagedElement REF Element,
    IN string Tree,
    IN uint16 QuotaType,
    OUT EmbeddedInstance("CIM_FSQuotaConfigEntry")string QuotaEntries[],
```

Given a combination of inputs, FindQuotaEntries( ) searches the quota entry database on the managed device for quota entries that match, and returns a list. On systems that support it, long-running queries may return a job.

Possible quota entries are:

#### 1) IdentityId

IdentityId is an optional string that can specify the UID, GID, or SID or can specify a pattern. The following rules apply to IdentityId:

- a) If IdentityId is NULL or the empty string, no identity-based quotas should be returned.
- b) If IdentityID is NULL, default quotas will be returned.
- c) If IdentityId is "\*", this matches all identity-based quotas entries.
- d) IdentityId may also be a specific ID, or the prefix of an ID followed by "\*". Standard prefix string matching is used to determine which entries match in this case.

#### 2) Element

Element is an optional reference to a ManagedElement (declared as CIM\_ManagedElement REF Element). The following rules apply to Element:

- a) When a ComputerSystem is passed for Element, the returned entries may be default or other entries for both the ComputerSystem and its contained FileSystems.
- b) When a FileSystem is passed in for Element, only entries pertaining to that FileSystem shall be returned. This may include default entries applicable to that FileSystem.
- c) If NULL is passed in for Element, the FSQuotaManagementService assumes that the Computer-System it is hosted on is the Element.

Element is an optional reference to a ComputerSystem or a LocalFileSystem, but is declared as a reference to a ManagedElement, which is the only common parent class.

#### 3) Tree

Tree is an optional string that identifies a tree (or trees) contained within a filesystem. The following rules apply to Tree:

- a) A null or empty string indicates that no tree quota entries should be returned.
- b) A "\*" tree parameter matches all tree quota entries defined within the filesystem(s) indicated by Element, if any.
- c) If Tree contains a path, it matches that path only. On some systems, this may result in multiple matches, one for the same-named tree in each of several filesystems.

- d) Prefix string matching may be used for the Tree parameter. E.g. the path "/x/y/\*" will match tree quotas on both "/x/y/m" and "/x/y/p".

#### 4) QuotaType

QuotaType is an enumerated optional parameter that describes what type of quota entries should be returned. The following rules apply to QuotaType:

- a) NULL or Unknown matches any entry (i.e., it is a wildcard).
- b) If Tree is specified, then entries which are pure tree quotas (not user-tree or group-tree) match.
- c) If User-Tree or Group-Tree is specified, then only user-tree or group-tree quotas match.

### 11.3.2 DeleteQuotaEntry

```
uint32 DeleteQuotaEntry(IN string EntryID);
```

This routine deletes a given quota entry from the managed device's quota entry database. Recall that the ManagedElement's name is specified as part of a QuotaEntry's InstanceID, above. A CIMOM managing multiple devices may use that to find which device to address when deleting the actual entry.

### 11.3.3 ModifyQuotaEntry

```
uint32 ModifyQuotaEntry(
    IN string EntryId,
    IN EmbeddedInstance("CIM_FSQuotaConfigEntry") string QuotaEntry,
    OUT CIM_Job REF Job;
);
```

Given the InstanceID of an existing quota entry, ModifyQuotaEntry( ) replaces it with a new entry specified as an EmbeddedInstance.

### 11.3.4 AddQuotaEntry

```
uint32 AddQuotaEntry(
    IN EmbeddedInstance("CIM_FSQuotaConfigEntry") string QuotaEntry,
    OUT CIM_Job REF Job;
);
```

This routine adds a new quota entry to the quota entry database on the appropriate managed element.

The ConflictingEntriesUsage property in FSQuotaCapabilities (see 11 "Filesystem Quotas Profile") will govern what happens if an entry already exists with the same combination of PrincipalID, ManagedElement, TreeName, ResourceType, QuotaType, and Default.

### 11.3.5 GetQuotaReport

```
uint32 GetQuotaReport(
    IN CIM_ManagedElement REF Element,
    IN string Tree,
    IN string User,
    IN EmbeddedInstance("CIM_FSDomainIdentity") string Group,
    IN, OUT string Cursor,
    IN, OUT uint64 NQuotas,
    OUT CIM_Job REF Job,
```



```

        OUT EmbeddedInstance("CIM_FSQuotaReportRecord") string ReportRecs[];
    };

```

This routine gets a quota report from a managed element. As there may be millions of records in this report, a chunking mechanism is provided so that the client does not become overwhelmed by the quantity of data furnished by the server.

The initial call to GetQuotaReport shall pass in NULL as a Cursor. Subsequent calls shall pass back the cursor exactly as received from the server, without modification, as an indication of where to continue the report from.

Clients which do not wish to use the chunking mechanism may pass a number such as  $2^{63} - 1$  in NQuotas.

On many systems, the initial phase of preparing a quota report may take some time. A job is returned in this case.

### 11.3.6 EnableQuotas

```

uint32 EnableQuotas(
    IN Boolean OnOff,
    IN CIM_ManagedElement element,
    OUT CIM_Job REF Job
);

```

This routine turns quota management on or off on a given managed element. This routine shall always be supported when the element is a CIM\_ComputerSystem. On systems that support it, the ManagedElement may alternatively be a filesystem. If an attempt is made to change the state on an unsupported ManagedElement, the routine shall return an appropriate error ("Operation unsupported for individual MEs of this type").

### 11.3.7 InitializeQuotas

```

uint32 InitializeQuotas(
    IN CIM_ComputerSystem REF Server,
    OUT CIM_Job REF Job);

```

Some systems require an explicit initialization step before quotas may be used. If this step takes some time, a job shall be returned. Systems which do not require this step shall return "Success".

## 11.4 Use Cases

Because quota management capabilities vary so widely from device to device, clients must be prepared to receive "unsupported" errors. These can be kept to a minimum by inspecting the QuotaCapabilities of the managed device. See the QuotaGetCapabilities routine in 11.4.1.

There are five fundamental operations on quotas:

1. Initialize the quota management system
2. Turn quota tracking on or off
3. Add or modify a quota table entry
4. Read the quota table
5. Get a report on quota usage for one or all entries in the quota table

The QuotaManagementService class contains extrinsics for all of these things, so each task reduces to getting the service instance and invoking the desired method.

The following example code is advisory.

#### 11.4.1 Common subroutines

In the Filesystem Quotas sample routines, the following subroutines are used (and declared inline here):

```

sub CIM_QuotaManagementService QuotaGetQMService(
    IN REF CIM_System system);
{
    services = Associators(system,
        "CIM_HostedService",
        "CIM_QuotaManagementService",
        "Antecedent",
        "Dependent",
        false, false, NULL);

    return services[0];
}

sub CIM_QuotaCapabilities QuotaGetCapabilities(
    IN REF CIM_System system)
{
    service = QuotaGetQMService(system);

    caps = Associators(service,
        "CIM_ElementCapabilities",
        "CIM_QuotaCapabilities",
        "CIM_ManagedElement",
        "ManagedElement",
        "Capabilities",
        false, false, NULL);

    return caps[0];
}

sub boolean QuotaSupportsPrincipalType(
    IN REF CIM_System system,
    IN uint16 type)
{
    capabilities = QuotaGetCapabilities(system);

    for(i = 0; capabilities.SupportedPrincipalTypes[i] != NULL; ++i) {
        if (capabilities.SupportedPrincipalTypes[i] == type) {
            return TRUE;
        }
    }
    return FALSE;
}

```

All of the following routines may return errors indicating that the supplied managed element is not supported. In most cases this will be because the operation (e.g. initializing quotas) is a system-wide operation, and cannot be done on a per-filesystem basis.

#### 11.4.2 Initialize quotas

```
sub uint_16 InitializeQuotas(
    IN REF CIM_System system)
{
    qms = QuotaGetQMService(system);

    result = qms->InitializeQuotas(system, job);

    //
    // See the Job Control profile for information on
    // handling the job if one is returned.
    //
    return result;
}
```

#### 11.4.3 Enable or disable quota tracking

```
//
// enable or disable quotas
//
// See the mof for the EnableQuotas extrinsic for possible
// return values
//
sub uint16 EnableQuotas(IN REF CIM_System system,
    IN REF CIM_ManagedElement me,
    IN boolean onoff)
{
    qms = QuotaGetQMService(system);
    result = qms->EnableQuotas(onoff, me, job);

    //
    // See the Job Control profile for information on
    // handling the job if one is returned.
    //
    return result;
}
```

#### 11.4.4 Add a quota entry

```
sub uint16 AddQuotaEntry(IN REF CIM_System system,
    IN REF CIM_ManagedElement me,
    IN String tree,
    IN REF CIM_DomainIdentity principal,
    IN uint64 hardlimit,
    IN uint64 softlimit,
    IN uint64 graceperiod,
```

## Filesystem Quotas Profile

```

        IN boolean active,
        IN string restype,
        IN uint16 quotatype,
        IN REF logicalfile,
        IN REF me,
        IN boolean default)
{
    service = QuotaGetQMService(system);
    entry = CreateInstance("CIM_FSQuotaConfigEntry");
    entry->HardLimit = hardlimit;
    entry->SoftLimit = softlimit;
    entry->SoftLimitGracePeriod = graceperiod;
    entry->Active = active;
    switch (restype) {
        case "Bytes": entry->ResourceType = 2;
        case "Files": entry->ResourceType = 3;
        case "Directories": entry->ResourceType = 4;
        case "Files+Directories": entry->ResourceType = 5;
        case "Inodes": entry->ResourceType = 6;
        default: entry->ResourceType = 0;
    }
    switch (quotatype) {
        case "User": entry->QuotaType = 2;
        case "Group": entry->QuotaType = 3;
        case "Tree": entry->QuotaType = 4;
        default: entry->QuotaType = 0;
    }
    if (principal != NULL) {
        entry->PrincipalID = principal->PrincipalID;
    }
    else
        entry->PrincipalID = NULL;
    if (logicalfile != NULL) {
        entry->TreeName = logicalfile->Name;
    }
    else
        entry->TreeName = NULL;
    entry->ManagedElement = me;
    entry->Default = default;
    entry->InstanceID = NULL;

    result = service->AddQuotaEntry(entry, job);

    //
    // analyse error, if there is one: the above code
    // cannot return '1' or '3', so only '2' is left.
    // And that means there's already an identical
    // entry, so declare victory and move on.
    //

```

```

        return result;          // could return 0, if you prefer
    }

```

#### 11.4.5 Delete a quota entry

```

//
// See the mof for the DeleteQuotaEntry extrinsic for possible
// return values
//
sub uint16 DeleteQuotaEntry(IN REF CIM_System system,
                           IN string entryid,
                           OUT REF CIM_Job job)
{
    service = QuotaGetQMService(system);
    result = service->DeleteQuotaEntry(entryid);

    return result;
}

```

#### 11.4.6 Modify a quota entry

```

//
// There are many ways to modify a quota entry. Here are
// a couple examples
//
sub uint16 ModifyQuotaHardLimit(IN REF CIM_System system,
                                IN string entryid,
                                IN uint64 newlimit)
{
    service = QuotaGetQMService(system);
    entry = GetInstance(entryid);
    entry->HardLimit = newlimit;
    result = service->ModifyQuotaEntry(entryid, entry, job);
    //
    // See the Job Control profile for information on
    // handling the job if one is returned.
    //
    return result;

sub uint16 SpecificUserToDefault(IN REF CIM_System system,
                                IN string uid)
{
    //
    // change Alice's quota to be the default for
    // all users
    //
    service = QuotaGetQMService(system);

```

```
//
// Need to search through all the quota entry instances
// for the given uid.
//
ges[] = EnumerateInstances("CIM_FSQuotaConfigEntry",
    true, false, false, false, "PrincipalID");
foreach qe (ges[]) {
    if (qe->PrincipalID == uid) {
        qe->PrincipalID = NULL;
        qe->Default = true;
        return 0;
    }
}
return 1;          // not found
}
```

#### 11.4.7 Read the quota entries

```
//
// Warning: on some systems, this may return 10's of
// thousands of entries
//
sub FSQuotaConfigEntry[] ReadQuotaEntries(IN REF CIM_System system)
{
    service = QuotaGetQMService(system);
    service->FindQuotaEntries(NULL, system, NULL, NULL, NULL,
        ges[], job);

    //
    // See the Job Control profile for information on
    // handling the job if one is returned.
    //
    return ges[];
}
```

#### 11.4.8 Get a report on quota usage

```
sub FSQuotaReportRecord[] GetQuotaReport(IN REF CIM_System system)
{
    cursor = NULL;
    service = QuotaGetQMService(system);
    nrecs = 1000;
    r = service->GetQuotaReport(system, NULL, NULL, NULL,
        cursor, nrecs, job, recs[]);
    <manage job>;
    <do something with recs>;
    while (r != "No more data") {
        r = service->GetQuotaReport(system, NULL, NULL, NULL,
            cursor, nrecs, job, recs[]);
        <manage job>;
    }
}
```

```

    <do something with recs>;
  }
}
}

```

## 11.5 CIM Elements

Table 173 describes the CIM elements for FileSystem Quotas.

**Table 173 - CIM Elements for FileSystem Quotas**

Element Name	Requirement	Description
11.5.1 CIM_FSDomainIdentity	Mandatory	A small class containing the unique ID of a user or group in a Unix or Windows domain.
11.5.2 CIM_FSQuotaAppliesToElement	Mandatory	An association between a quota config entry and a managed element.
11.5.3 CIM_FSQuotaAppliesToPrincipal	Mandatory	An association between a quota config entry and a filesystem principal entity.
11.5.4 CIM_FSQuotaAppliesToTree	Mandatory	An association between a quota config entry and a directory.
11.5.5 CIM_FSQuotaCapabilities	Mandatory	The supported targets, quota types, resource types and behaviors of the FSQuotaManagementService associated to this class instance.
11.5.6 CIM_FSQuotaConfigEntry	Mandatory	A single quota entry in the configuration database.
11.5.7 CIM_FSQuotaIndication	Optional	An indication specially referring to quota events. Note that the threshold and current value are passed in the parent class, in ThresholdValue and ObservedValue.
11.5.8 CIM_FSQuotaManagementService	Mandatory	Quota Management Service class.
11.5.9 CIM_FSQuotaReportRecord	Mandatory	A class representing a single line in a quota report generated by a call to the QuotaReport() extrinsic of the FSQuotaManagementService.
11.5.10 CIM_ReportRecord	Mandatory	An abstract keyless class proposed as the root of a tree of report record classes.
SELECT * FROM CIM_FSQuotaIndication WHERE CIM_FSQuotaIndicationWhichLimit = 2	Mandatory	Hard quota threshold crossed.
SELECT * FROM CIM_FSQuotaIndication WHERE CIM_FSQuotaIndication::WhichLimit = 3	Mandatory	Soft quota threshold crossed.

### 11.5.1 CIM\_FSDomainIdentity

Created By: CreateInstance\_or\_Static\_or\_External

Deleted By: Static

Requirement: Mandatory

Table 174 describes class CIM\_FSDomainIdentity.

**Table 174 - SMI Referenced Properties/Methods for CIM\_FSDomainIdentity**

Properties	Flags	Requirement	Description & Notes
PrincipalID		Mandatory	The unique ID of a principal. This may be a UID, GID or a SID.
DomainType		Mandatory	The type of domain the Principal belongs to. Possible values are "Unknown", "Other", "Unix", and "Active Directory".
PrincipalType		Mandatory	The type of Principal represented by this identity instance. Possible values are "Unknown", "Other", "User" and "Group".

### 11.5.2 CIM\_FSQuotaAppliesToElement

Created By: CreateInstance

Modified By: Extrinsic\_or\_External

Deleted By: DeleteInstance

Requirement: Mandatory

Table 175 describes class CIM\_FSQuotaAppliesToElement.

**Table 175 - SMI Referenced Properties/Methods for CIM\_FSQuotaAppliesToElement**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The managed element.
Dependent		Mandatory	The quota config entry.

### 11.5.3 CIM\_FSQuotaAppliesToPrincipal

Created By: CreateInstance

Modified By: Extrinsic\_or\_External

Deleted By: DeleteInstance

Requirement: Mandatory

Table 176 describes class CIM\_FSQuotaAppliesToPrincipal.

**Table 176 - SMI Referenced Properties/Methods for CIM\_FSQuotaAppliesToPrincipal**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The filesystem principal.
Dependent		Mandatory	The quota config entry.

### 11.5.4 CIM\_FSQuotaAppliesToTree

Created By: CreateInstance

Modified By: Extrinsic\_or\_External

Deleted By: DeleteInstance

Requirement: Mandatory



Table 177 describes class CIM\_FSQuotaAppliesToTree.

**Table 177 - SMI Referenced Properties/Methods for CIM\_FSQuotaAppliesToTree**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The filesystem directory tree.
Dependent		Mandatory	The quota config entry.

### 11.5.5 CIM\_FSQuotaCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 178 describes class CIM\_FSQuotaCapabilities.

**Table 178 - SMI Referenced Properties/Methods for CIM\_FSQuotaCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	A unique ID for the capabilities instance.
ElementName		Mandatory	A user-friendly name for the instance in the format SystemName:ManagedElementName:Capabilities.
SupportedTargetTypes		Mandatory	The target types supported by the Service. Possible values are "ComputerSystem" and "FileSystem".
SupportedPrincipalTypes		Mandatory	An array of the types of Principal supported by the Service. Possible values are "User", "Group", "User-tree", "Group-tree" and "Tree".
ConflictingEntriesUsage		Mandatory	The behavior of the system when it encounters quota entries with duplicate keys.
SupportedResourceTypes		Mandatory	An array of resource types that may have quotas placed on them by this Service. Possible values are "Unknown", "Other", "Bytes", "Files", "Directories", "Files+Directories", "Inodes" and "Blocks".
DefaultSupported		Mandatory	An array that indicates which resource types may have default quotas set upon them by this Service. Possible values are the same as for SupportedResourceTypes.
IsActiveSettingPerEntrySupported		Mandatory	Indicates whether quotas may be made active or inactive per entry.
IsMonitoredSettingPerEntrySupported		Mandatory	Indicates whether quota monitoring may be turned on or off per entry.
IsGracePeriodSupported		Mandatory	Indicates whether a grace period may be set on a quota. If it can, then crossing over a soft threshold for more than the period of time specified in the grace period effectively converts the soft threshold to a hard limit, cutting off further allocation of the resource.

### 11.5.6 CIM\_FSQuotaConfigEntry

Created By: Extrinsic\_or\_External

Modified By: Extrinsic\_or\_External

Deleted By: Extrinsic\_or\_External

Requirement: Mandatory

Table 179 describes class CIM\_FSQuotaConfigEntry.

**Table 179 - SMI Referenced Properties/Methods for CIM\_FSQuotaConfigEntry**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	A unique ID for the entry.
HardLimit		Mandatory	The hard limit for this quota.
SoftLimit		Mandatory	The soft limit for this quota.
SoftLimitGracePeriod		Optional	Length of the grace period, if any exists.
Active		Optional	Whether or not this quota is to be actively monitored. If NULL, the system does not support activation of individual quotas.
Monitor		Mandatory	Whether or not this is a monitor-only quota. If this is TRUE, no enforcement of any kind is done.
ResourceType		Mandatory	The type of resource being managed.
QuotaType		Mandatory	The type of quota to create (user, group, etc.).
TreeName		Mandatory	Path of the tree over which this quota is applicable, if any.
PrincipalID		Mandatory	The user or group being constrained by this quota, if any.
FileSystem		Mandatory	A The name of the FileSystem, if any, over which this quota is monitored.
Default		Mandatory	Whether or not this is a default quota.

### 11.5.7 CIM\_FSQuotaIndication

Created By: External

Deleted By: Static

Requirement: Optional

Table 180 describes class CIM\_FSQuotaIndication.

**Table 180 - SMI Referenced Properties/Methods for CIM\_FSQuotaIndication**

Properties	Flags	Requirement	Description & Notes
IdentityID		Mandatory	The InstanceID of the FSDomainIdentity involved in causing the event. If there is none, NULL shall be passed in this property.
EntryID		Mandatory	The InstanceID of the FSQuotaConfigEntry involved in causing the event.
Path		Mandatory	The complete path of the tree involved in causing the event. If there is none, NULL shall be passed in this property.
WhichLimit		Mandatory	Either "hard" or "soft".
ResourceType		Mandatory	"Bytes", "Files", "Directories", "Files+Directories" or "Inodes".
QuotaType		Mandatory	Either "user", "group" or "tree".
Limit		Mandatory	The limit set by the quota entry.

**Table 180 - SMI Referenced Properties/Methods for CIM\_FSQuotaIndication**

Properties	Flags	Requirement	Description & Notes
AmountUsed		Optional	Amount of resource actually used at the time the indication was generated.
FileSystem		Mandatory	The Name of the FileSystem in which the event occurred.

**11.5.8 CIM\_FSQuotaManagementService**

Created By: Static

Deleted By: Static

Requirement: Mandatory

Table 181 describes class CIM\_FSQuotaManagementService.

**Table 181 - SMI Referenced Properties/Methods for CIM\_FSQuotaManagementService**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
FindQuotaEntries()		Mandatory	Retrieve one or more quota entries based on a set of input criteria.
DeleteQuotaEntry()		Mandatory	Delete a specified quota entry.
ModifyQuotaEntry()		Mandatory	Modify a specified quota entry.
AddQuotaEntry()		Mandatory	Add a new quota entry.
GetQuotaReport()		Mandatory	Obtain a report on usage against the quota entries on a system.
EnableQuotas()		Mandatory	Turn quota monitoring on or off.
InitializeQuotas()		Mandatory	Initialize the quota reporting system.

**11.5.9 CIM\_FSQuotaReportRecord**

Created By: Extrinsic

Deleted By: Static

Requirement: Mandatory

Table 182 describes class CIM\_FSQuotaReportRecord.

**Table 182 - SMI Referenced Properties/Methods for CIM\_FSQuotaReportRecord**

Properties	Flags	Requirement	Description & Notes
HardLimit		Optional	The hard threshold associated with this quota report record, if any.
SoftLimit		Optional	The soft threshold associated with this quota report record, if any.

**Table 182 - SMI Referenced Properties/Methods for CIM\_FSQuotaReportRecord**

Properties	Flags	Requirement	Description & Notes
SoftLimitGracePeriod		Optional	The grace period associated with the soft limit associated with this report record, if any.
Active		Optional	Whether the quota associated with this report record is being actively enforced. If not, this indicates the quota is being used for tracking purposes only.
Monitored		Optional	Whether or not thresholds on this quota are being monitored. If a system reports quotas that aren't being monitored, this value may be false.
ResourceType		Mandatory	The type of resource whose use is counted in this quota report record.
QuotaType		Mandatory	The type of Principal to which this quota applies. Possible values are "Unknown", "Other", "User", "Group" and "Tree".
AmountUsed		Mandatory	The amount of resource used by the combination of Principal, Resource type, Tree, and ManagedElement specified in the quota configuration entry that generated this quota report record (and reported in other fields in the record).
TreeName		Optional	The URI of the filesystem tree upon which the quota was set, if any.
PrincipalID		Optional	The FSDomainIdentity for the Principal associated with this quota report record, if any.
FileSystem		Optional	The name of the filesystem over which the quota entry that generated the report record was placed, if any.

**11.5.10CIM\_ReportRecord**

Created By: Static

Deleted By: Static

Requirement: Mandatory

**EXPERIMENTAL**

---

---

## STABLE

### 12 NAS Head Profile

#### 12.1 Description

##### 12.1.1 Synopsis

**Profile Name:** NAS Head (Autonomous Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** ComputerSystem

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 183 describes the related profiles for NAS Head.

**Table 183 - Related Profiles for NAS Head**

Profile Name	Organization	Version	Requirement	Description
Filesystem	SNIA	1.7.0	Mandatory	
File Storage	SNIA	1.4.0	Mandatory	
File Export	SNIA	1.7.0	Mandatory	
NAS Network Port	SNIA	1.7.0	Mandatory	
Access Points	SNIA	1.3.0	Optional	
Multiple Computer System	SNIA	1.2.0	Optional	
Software	SNIA	1.4.0	Optional	
Location	SNIA	1.4.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	
Filesystem Manipulation	SNIA	1.7.0	Optional	
File Export Manipulation	SNIA	1.7.0	Optional	
File Server Manipulation	SNIA	1.7.0	Optional	
Filesystem Performance	SNIA	1.7.0	Optional	Experimental.
FileSystem Quotas	SNIA	1.7.0	Optional	Experimental.
Filesystem Replication Services	SNIA	1.7.0	Optional	Experimental.
Job Control	SNIA	1.5.0	Optional	
SAS Initiator Ports	SNIA	1.7.0	Optional	
FC Initiator Ports	SNIA	1.7.0	Optional	
Device Credentials	SNIA	1.3.0	Optional	
Operational Power	SNIA	1.7.0	Optional	Experimental.
Launch In Context	DMTF	1.0.0	Optional	Experimental. See DSP1102, version 1.0.0
Physical Package	SNIA	1.5.0	Mandatory	

**Table 183 - Related Profiles for NAS Head**

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.7.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

### 12.1.2 Overview

The NAS Head Profile exports File elements (contained in a FileSystem) as FileShares. The storage for the FileSystem is obtained from external SAN storage, for example, a Storage Array that exports Storage Volumes as LUNs. The storage array may also provide storage to other hosts or devices (or other NAS Heads), and the storage on the array might be visible to other external management tools, and may be actively managed independently.

This profile models the necessary filesystem and NAS concepts and defines how the connections to the underlying storage is managed. The details of how a Storage Array exports storage to the NAS Head is not covered in this profile but is covered by the Array Profile.

The NAS Head Profile reuses a significant portion of 16 Storage Virtualizer Profile in *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*.

The NAS Head Profile and its component profiles and packages are illustrated in Figure 18: "NAS Head Profiles".

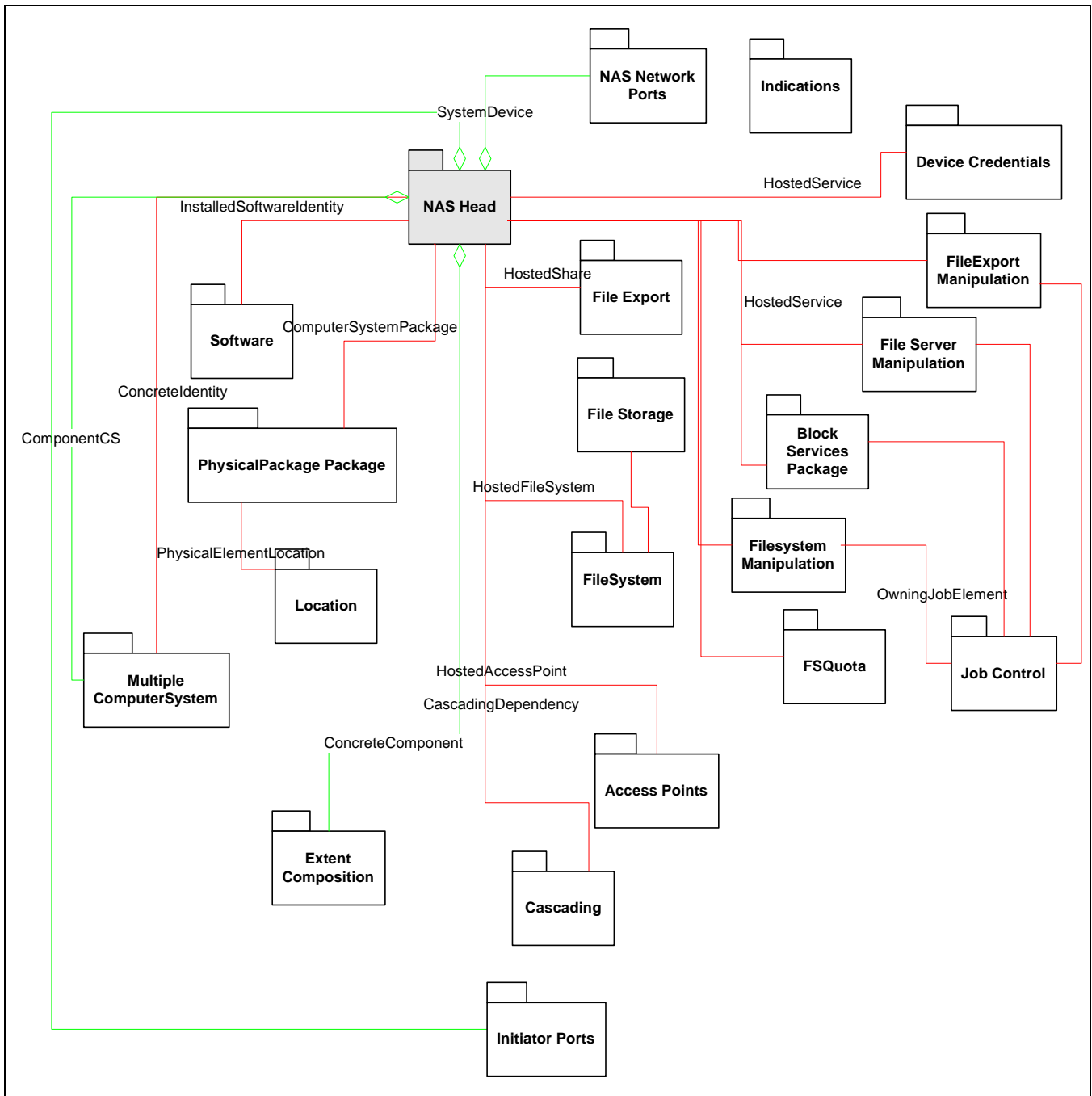


Figure 18 - NAS Head Profiles

### 12.1.3 Implementation

#### 12.1.3.1 Summary Instance Diagram

Figure 19: "NAS Head Instance" illustrates the mandatory classes for the NAS Head Profile. This figure shows all the classes that are mandatory for the NAS Head Profile. Later diagrams will review specific sections of this diagram.

## NAS Head Profile

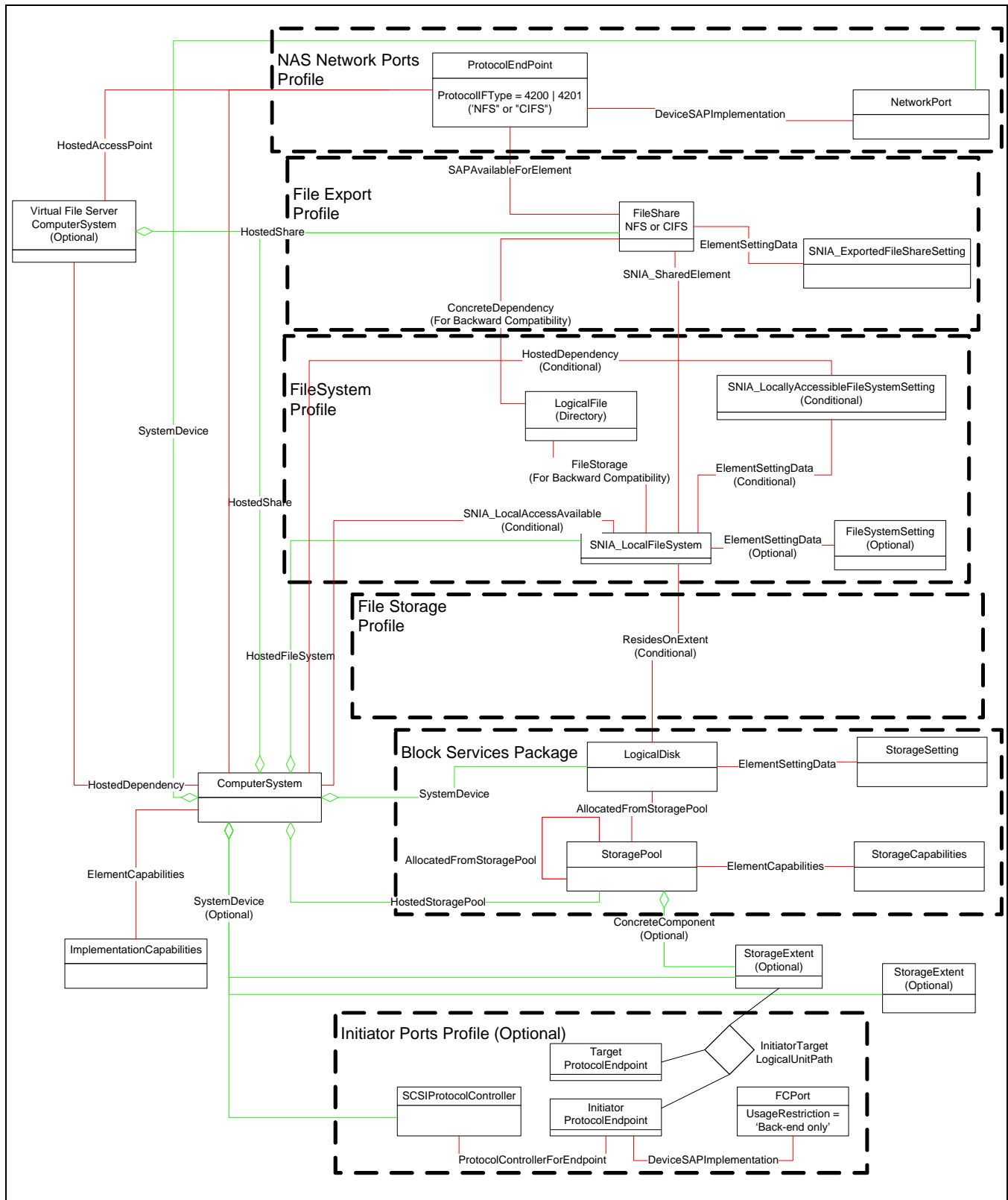


Figure 19 - NAS Head Instance



The NAS Head Profile closely parallels the Storage Virtualizer Profile in how it models storage. Storage is assigned to StoragePools and LogicalDisks are allocated from those storage pools for the purpose of holding local filesystems of the NAS.

As with the Storage Virtualizer Profile, the NAS Head StoragePools have StorageCapabilities associated to the StoragePools via ElementCapabilities. Similarly, LogicalDisks that are allocated from those StoragePools have StorageSettings, which are associated to the LogicalDisk via ElementSettingData. StoragePools are hosted by a ComputerSystem that represents the NAS “top level” system, and the StorageExtents have a SystemDevice association to the “top level” ComputerSystem.

NOTE As with Self-Contained NAS, the StoragePools may be hosted by a component ComputerSystem if the Profile has implemented the Multiple Computer System Profile.

As with the Storage Virtualizer Profile, the “top level” ComputerSystem of the NAS Head does not (and typically isn’t) a real ComputerSystem. It is merely the ManagedElement upon which all aspects of the NAS offering are scoped.

A NAS Head may implement “Virtual File Servers” in addition to, or instead of, implementing File Servers in the Top Level ComputerSystem or one of the Multiple Computer System ComputerSystems. A Virtual File Server shall have a HostedDependency to either the top level NAS ComputerSystem or one of the Multiple Computer System ComputerSystems. NOTE: A Virtual File Server shall not have a ComponentCS association to the top level NAS ComputerSystem.

As with Storage Virtualizer Profile, the NAS Head draws its storage from an open SAN. That is, the actual disk storage is addressable independent of the NAS Head. As a result, the NAS head shall model the Initiator ports and the StorageExtents that it acquires from the SAN. The NAS Head supports at least one of the Initiator Ports Profiles (the dashed box at the bottom of Figure 19: “NAS Head Instance”) to effect the support for backend ports. The NAS Head includes the Block Services Package to effect the logical storage management (the dashed box just above the Initiator Ports dashed box in Figure 19: “NAS Head Instance”).

Everything above the LogicalDisk is specific to NAS (and does not appear in the Storage Virtualizer Profile). LocalFileSystems are created on the LogicalDisks, LogicalFiles within those LocalFileSystems are shared (FileShare) through ProtocolEndpoints associated with NetworkPorts.

NOTE The classes and associations in the dashed boxes are from the required packages and component profiles (as indicated by the labels on the dashed boxes).

The ConcreteDependency association is provided for backward compatibility with previous releases of SMI-S. It represents a relationship between a FileShare and a Directory.

The ResidesOnExtent is conditional on the use by NAS profiles (NAS Head and Self-contained NAS) in the File Storage Profile. In the NAS Head a LocalFileSystem shall map to a LogicalDisk.

Also note that FileSystemSetting (and the corresponding ElementSettingData) are also optional. They are only shown here to illustrate where they would show up in the model should they be implemented.

In the base NAS Head profile, the classes and associations shown in Figure 19: “NAS Head Instance” are automatically populated based on how the NAS Head is configured. Client modification of the configuration (including configuring storage, creating extents, local filesystems and file shares) are functions found in component profiles of the NAS Head Profile.

---

---

## EXPERIMENTAL

An instance of ImplementationCapabilities may be associated to the top level NAS Head ComputerSystem. This Capabilities instance identifies the capacity optimization techniques supported by

the implementation. An implementation may advertise that it supports "None", "SNIA:Thin Provisioning", "SNIA:Data Compression" or "SNIA:Data Deduplication".

## EXPERIMENTAL

### 12.1.3.2 NAS Storage Model

Figure 20: "NAS Storage Instance" illustrates the classes mandatory for modeling of storage for the NAS Head Profile.

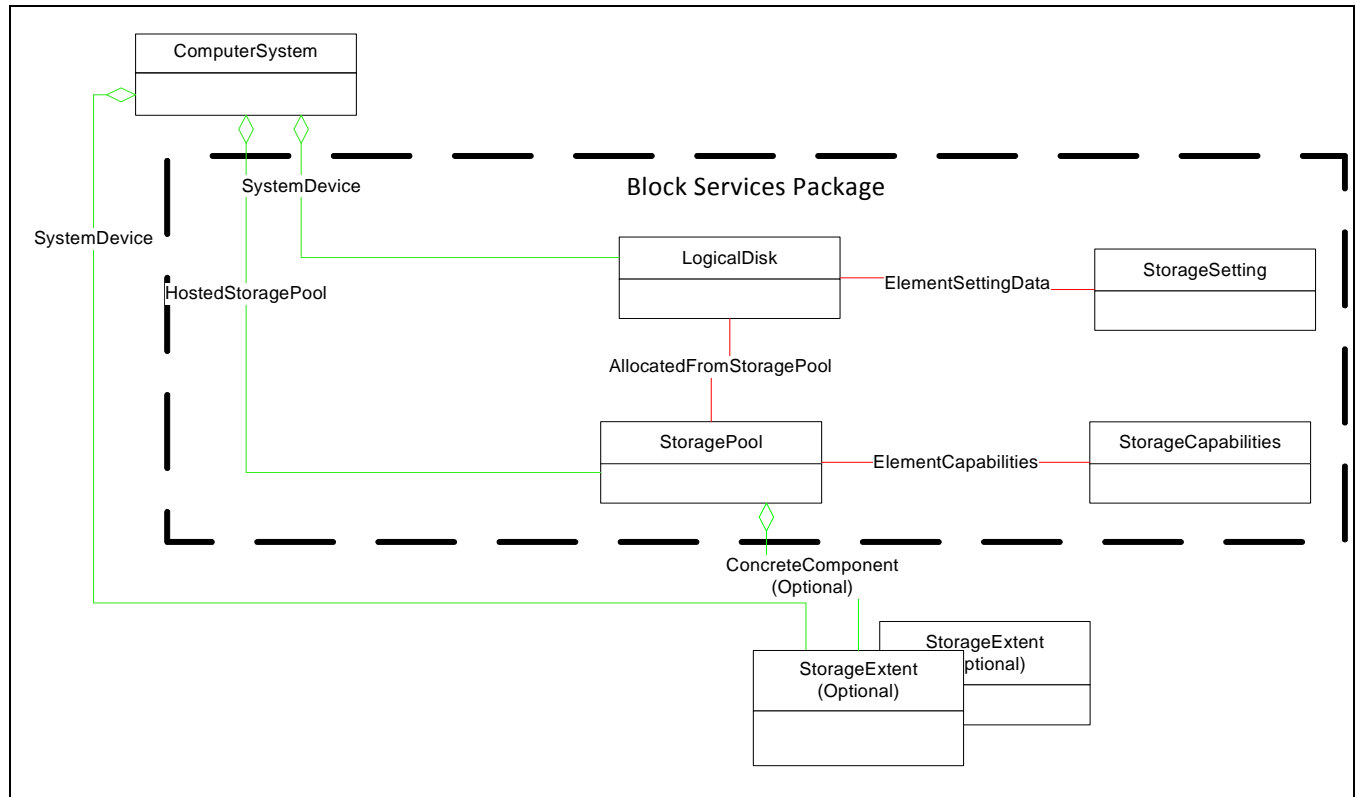


Figure 20 - NAS Storage Instance

The NAS Head Profile uses most of the classes and associations used by the Storage Virtualizer Profile (including those in the Block Services Package). In doing this, it leverages many of the component profiles that are available for Storage Virtualizer Profiles. The classes and associations shown in Figure 20: "NAS Storage Instance" are the minimum mandatory for read only access in the base profile.

Storage for the NAS shall be modeled as logical storage. That is, StoragePools shall be modeled, including the HostedStoragePool and ElementCapabilities to the StorageCapabilities supported by the StoragePool. In addition, for NAS Heads, which get their storage from a SAN, the StorageExtents that compose the primordial StoragePools shall also be modeled with ConcreteComponent associations to the StoragePool to which they belong and they will be primordial.

In addition, in order for storage to be used it shall be allocated to one or more LogicalDisks. A LogicalDisk shall have an AllocatedFromStoragePool association to the StoragePool from which it is allocated. The LogicalDisk shall have an ElementSettingData association to the settings that were used when the LogicalDisk was created.

For manipulation of Storage, see 5 Block Services Package of *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*. LogicalDisks are the ElementType that is supported for storage allocation

functions (e.g., `CreateOrModifyElementFromStoragePool` and `ReturnToStoragePool`), but the Block Services methods for managing LogicalDisks are optional for the NAS Head Profile. The NAS Head Profile also supports (optionally) the Pool manipulation functions (e.g., `CreateOrModifyStoragePool` and `DeleteStoragePool`) of the Block Services Package.

#### 12.1.3.3 NAS Head Use of Filesystem Profile (Mandatory)

The NAS Head Profile uses the Filesystem Profile for modeling of its filesystem constructs. For the NAS Head, the Filesystem Profile shall be supported. See 8 Filesystem Profile for details on this modeling.

#### 12.1.3.4 NAS Head Use of File Storage Profile (Mandatory)

The NAS Head Profile uses the File Storage Profile for modeling of its file storage constructs. For the NAS Head, the Filesystem Profile shall be supported. See 7 File Storage Profile for details on the file storage modeling.

#### 12.1.3.5 NAS Head Use of File Export Profile (Mandatory)

The NAS Head Profile uses the File Export Profile for modeling of its file export constructs. For the NAS Head, the File Export Profile shall be supported. See 4 File Export Profile for details on this modeling.

#### 12.1.3.6 NAS Head Use of NAS Network Ports Profile (Mandatory)

The NAS Head Profile uses the NAS Network Ports Profile for modeling of its file export constructs. For the NAS Head, the NAS Network Ports Profile shall be supported. See 14 NAS Network Port Profile for details on this modeling.

#### 12.1.3.7 Indication Events

##### 12.1.3.7.1 InstModification of ComputerSystem

Table 184 identifies the standard OperationalStatus values and the events that are being indicated.

**Table 184 - InstModification Events for ComputerSystem**

New OperationalStatus	Event / Correlated Indications
OK	The top level NAS system is fully operational.
	An Error in the Top Level NAS system was corrected and the system is now fully functional.
	Self test is complete and the NAS system is fully operational
Degraded	The system is running but with limitations.
Error	The system is experiencing an error and is not functional.
Stopped	The system has been stopped.
No contact	The system status cannot be determined, due to no response from the system.
Starting	The system is starting, but it not yet functional.
Stopping	The system is stopping.
Lost communication	The system status cannot be determined, due to communications problems.

### 12.1.3.7.2 InstModification of LogicalDisk

## EXPERIMENTAL

Table 185 identifies the standard OperationalStatus values and the events that are being indicated.

**Table 185 - InstModification Events for LogicalDisk**

New OperationalStatus	Event / Correlated Indications
OK	The logical disk is fully functional.
Degraded	The logical disk is experiencing errors, but is still supporting data.
	The Logical disk is rebuilding, so performance may suffer.
Error	The logical disk has an error and is not usable.
Starting	The logical disk is being brought online.
Dormant	The logical disk is offline.

## EXPERIMENTAL

## EXPERIMENTAL

### 12.1.3.8 Bellwether Indications

#### 12.1.3.8.1 AlertIndication for ComputerSystem Bellwether

This AlertIndication signals the change in status (OperationalStatus) of a ComputerSystem as a bellwether event. It is supported by a standard message (MessageID=FSM1). Table 186 shows the OperationalStatus values that may signal that changes may have occurred in related elements (Implied Indications Inhibited).

**Table 186 - Bellwether AlertIndication Events for ComputerSystem**

New OperationalStatus	Implied Indications Inhibited
OK, Degraded, Error, Stopped	OperationalStatus changes to Elements with SystemDevice associations to this ComputerSystem (LogicalDisks, ...)
	OperationalStatus changes to Elements with HostedService associations to this ComputerSystem (FileSystemConfigurationService, FileExportService, ...)
	OperationalStatus changes to FileSystems with HostedFileSystem associations to this ComputerSystem.
	OperationalStatus changes to StoragePools with HostedStoragePool associations to this ComputerSystem.
	OperationalStatus changes to ProtocolEndpoints with HostedAccessPoint associations to this ComputerSystem.
	OperationalStatus changes to FileShares with HostedFileShare associations to this ComputerSystem.
No contact, Starting, Stopping, Lost communication	None

### 12.1.3.8.2 AlertIndication for LogicalDisk Bellwether

This AlertIndication signals the change in status (OperationalStatus) of a LogicalDisk as a bellwether event. It is supported by a standard message (MessageID=FSM3). Table 187 shows the OperationalStatus values that may signal that changes may have occurred in related elements (Implied Indications Inhibited).

**Table 187 - Bellwether AlertIndication Events for LogicalDisk**

New OperationalStatus	Implied Indications Inhibited
OK, Degraded, Error, Stopped	OperationalStatus changes to FileSystems with ResidesOn associations to this LogicalDisk.
Unknown	None

## EXPERIMENTAL

---

### 12.2 Health and Fault Management Considerations

The NAS Head supports state information (e.g., OperationalStatus) on the following elements of the model:

- Network Ports (See 14.3.1 "OperationalStatus for Network Ports")
- Back-end Ports (See 16.3.3 "Health and Fault Management Considerations" in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*)
- ComputerSystems (See 22.2.5 Computer System Operational Status in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*)
- FileShares that are exported (See 4.2.1 "OperationalStatus for FileShares")
- LocalFileSystems (See 8.2.1 "OperationalStatus for Filesystems")
- ProtocolEndpoints (See 14.3.2 "OperationalStatus for ProtocolEndpoints")

---



---

## EXPERIMENTAL

### 12.2.1 Standard Messages used by this Profile

The standard messages specific to this profile are listed in Table 188.

**Table 188 - Standard Messages used by NAS Head**

Message ID	Message Name
FSM1	ComputerSystem bellwether alert
FSM3	LogicalDisk bellwether alert
Core12	Drive not responding
Core13	Fan Failure
Core14	Power Supply Failure
Core20	Drive is responding
Core21	Cooling Fan Issues Cleared
Core22	Power Supply Issues Cleared
Core23	Controller Failure
Core24	Controller Issues Cleared
DRM38	Volume or pool degraded
DRM39	Volume or pool failed
DRM40	Volume or pool issues cleared

---



---

## EXPERIMENTAL

### 12.3 Methods of the Profile

#### 12.3.1 Extrinsic Methods of the Profile

Not defined in this standard.

#### 12.3.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

Manipulation functions are supported in component profiles of the profile.

## 12.4 Use Cases

Not defined in this standard.

## 12.5 CIM Elements

Table 189 describes the CIM elements for NAS Head.

**Table 189 - CIM Elements for NAS Head**

Element Name	Requirement	Description
12.5.1 CIM_ComputerSystem (Top Level System)	Mandatory	This declares that at least one computer system entry will pre-exist. The Name property should be the Unique identifier for the NAS Head. Associated to RegisteredProfile.
12.5.2 CIM_ComputerSystem (Virtual File Server)	Optional	This represents a Virtual File Server, if one exists.
12.5.3 CIM_ConcreteComponent	Optional	Represents the association between a Primordial StoragePool and the underlying StorageExtents that compose it.
12.5.4 CIM_ElementCapabilities (ImplementationCapabilities to Service)	Optional	Experimental. Associates the top level NAS Head ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.
12.5.5 CIM_HostedDependency	Optional	Associates a Virtual File Server to the Computer System hosting it. This is required if a Virtual File Server exists.
12.5.6 CIM_ImplementationCapabilities (ImplementationCapabilities)	Optional	Experimental. The capabilities of the profile implementation.
12.5.7 CIM_LogicalDisk (LD for FS)	Mandatory	Represents the single Storage Extent on which the NAS Head will build a LocalFileSystem.
12.5.8 CIM_StorageExtent (Primordial Imported Extent)	Optional	This StorageExtent represents the LUNs (StorageVolumes) imported from a storage device to the NAS Head.
12.5.9 CIM_SystemDevice (Logical Disks)	Mandatory	This association links all LogicalDisks to the scoping system.
12.5.10 CIM_SystemDevice (Storage Extents)	Conditional	Conditional requirement: This is required if primordial StorageExtents exist. This association links all StorageExtents to the scoping system.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus <> PreviousInstance.CIM_ComputerSystem::OperationalStatus	Mandatory	CQL -Change of Status of a NAS ComputerSystem (controller).  PreviousInstance is optional, but may be supplied by an implementation of the Profile.  See 12.1.3.7.1 InstModification of ComputerSystem.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="SNIA" and MessageID="FSM1"	Optional	CQL -This is a bellwether indication of a change of Status of a NAS ComputerSystem (controller) and related classes (LogicalDisks, Services, ProtocolEndpoints, StoragePools, FileShares and FileSystems).  See 12.1.3.8.1 AlertIndication for ComputerSystem Bellwether

**Table 189 - CIM Elements for NAS Head**

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="SNIA" and MessageID="FSM3"	Optional	CQL -This is a bellwether indication of a change of status of a LogicalDisk.  See 12.1.3.8.2 AlertIndication for LogicalDisk Bellwether
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus	Mandatory	CQL -Change of status of a LogicalDisk.  PreviousInstance is optional, but may be supplied by an implementation of the Profile.  See 12.1.3.7.2 InstModification of LogicalDisk.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core12'	Conditional	Experimental. If hardware is capable of supporting, indicates that a DiskDrive has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core20'	Conditional	Experimental. If hardware is capable of supporting, indicates that a DiskDrive has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core13'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Fan has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core21'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Fan has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core14'	Conditional	Experimental. If hardware is capable of supporting, indicates that a PowerSupply has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core22'	Conditional	Experimental. If hardware is capable of supporting, indicates that a PowerSupply has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core23'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Controller has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core24'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Controller has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM38'	Mandatory	Experimental. A LogicalDisk has degraded.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM39'	Mandatory	Experimental. A LogicalDisk has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM40'	Mandatory	Experimental. A LogicalDisk has returned to normal service.

**12.5.1 CIM\_ComputerSystem (Top Level System)**

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'NAS Head', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.6.0'.



Table 190 describes class CIM\_ComputerSystem (Top Level System).

**Table 190 - SMI Referenced Properties/Methods for CIM\_ComputerSystem (Top Level System)**

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	The actual class of this object, e.g., Vendor_NASComputerSystem.
ElementName		Mandatory	User friendly name.
Name		Mandatory	Unique identifier for the NAS Head in a format specified by NameFormat. For example, IP address or Vendor/Model/SerialNo.
OperationalStatus		Mandatory	Overall status of the NAS Head. The standard values are 2 (OK), 3 (Degraded), 6 (Error), 8 (Starting), 9 (Stopping), 10 (Stopped), 12 (No contact) or 13 (Lost Communication).
NameFormat		Mandatory	Format for Name property.
PrimaryOwnerContact	M	Optional	Owner of the NAS Head.
PrimaryOwnerName	M	Optional	Contact details for owner.
Dedicated		Mandatory	This shall be a NAS Head (24).
OtherIdentifyingInfo	C	Mandatory	An array of know identifiers for the NAS Head.
IdentifyingDescriptions	C	Mandatory	An array of descriptions of the OtherIdentifyingInfo. Some of the descriptions would be "Ipv4 Address", "Ipv6 Address" or "Fully Qualified Domain Name".

### 12.5.2 CIM\_ComputerSystem (Virtual File Server)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 191 describes class CIM\_ComputerSystem (Virtual File Server).

**Table 191 - SMI Referenced Properties/Methods for CIM\_ComputerSystem (Virtual File Server)**

Properties	Flags	Requirement	Description & Notes
Dedicated		Mandatory	A Virtual File Server is a File Server (16).
NameFormat		Mandatory	Format for Name property. This shall be "Other".
Name	C	Mandatory	Unique identifier for the NAS Head's Virtual File Servers (E.g., Vendor/Model/SerialNo+FS+Number).
OperationalStatus		Mandatory	Overall status of the Virtual File Server. The standard values are 2 (OK), 3 (Degraded), 6 (Error), 8 (Starting), 9 (Stopping), 10 (Stopped), 12 (No contact) or 13 (Lost Communication).

### 12.5.3 CIM\_ConcreteComponent

Created By: External

Modified By: Static

Deleted By: External

Requirement: Optional

Table 192 describes class CIM\_ConcreteComponent.

**Table 192 - SMI Referenced Properties/Methods for CIM\_ConcreteComponent**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Primordial StoragePool that is built from the StorageExtent.
PartComponent		Mandatory	A StorageExtent that is part of a Primordial StoragePool.

#### 12.5.4 CIM\_ElementCapabilities (ImplementationCapabilities to Service)

Experimental. Associates the top level NAS Head ComputerSystem to the CIM\_ImplementationCapabilities supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 193 describes class CIM\_ElementCapabilities (ImplementationCapabilities to Service).

**Table 193 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (ImplementationCapabilities to Service)**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The ImplementationCapabilities.
ManagedElement		Mandatory	The top level NAS Head ComputerSystem that has ImplementationCapabilities.

#### 12.5.5 CIM\_HostedDependency

Created By: External

Modified By: Static

Deleted By: External

Requirement: Optional

Table 194 describes class CIM\_HostedDependency.

**Table 194 - SMI Referenced Properties/Methods for CIM\_HostedDependency**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The Virtual File Server ComputerSystem. A Virtual File Server ComputerSystem is a File Server and shall have Dedicated=16 (File Server).
Antecedent		Mandatory	The hosting ComputerSystem. The hosting ComputerSystem may be the top level NAS ComputerSystem or an Multiple Computer System (non-top level) system.

#### 12.5.6 CIM\_ImplementationCapabilities (ImplementationCapabilities)

Experimental. The capabilities (features) of the profile implementation.

Created By: Static  
 Modified By: Static  
 Deleted By: Static  
 Requirement: Optional

Table 195 describes class CIM\_ImplementationCapabilities (ImplementationCapabilities).

**Table 195 - SMI Referenced Properties/Methods for CIM\_ImplementationCapabilities (ImplementationCapabilities)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the implementation capability of an implementation.
ElementName		Optional	A provider supplied user-friendly name for this CIM_ImplementationCapabilities element.
SupportedCapacityOptimizations		Mandatory	This array of strings lists the capacity optimization techniques that are supported by the implementation. Valid string values are "none"   "SNIA:Thin Provisioning"   "SNIA:Data Compression"   "SNIA:Data Deduplication".

### 12.5.7 CIM\_LogicalDisk (LD for FS)

Created By: Extrinsic\_or\_External  
 Modified By: Extrinsic\_or\_External  
 Deleted By: Extrinsic\_or\_External  
 Requirement: Mandatory

Table 196 describes class CIM\_LogicalDisk (LD for FS).

**Table 196 - SMI Referenced Properties/Methods for CIM\_LogicalDisk (LD for FS)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	CIM Class of the NAS Head Computer System that is the host of this LogicalDisk.
SystemName		Mandatory	Name of the NAS Head Computer System that hosts this LogicalDisk.
CreationClassName		Mandatory	CIM Class of this instance of LogicalDisk.
DeviceID		Mandatory	Opaque identifier for the LogicalDisk.
OperationalStatus		Mandatory	A subset of operational status that is applicable for LogicalDisks in a NAS Head. The standard values for this are 2 (OK), 3 (Degraded), 6 (Error), 8 (Starting) or 15 (Dormant).
ExtentStatus		Mandatory	This LogicalDisk is neither imported (16) nor exported (17). The standard values for this are 0 (Other), 1 (Unknown), 2 (None/Not Applicable), 3 (Broken), 4 (Data Lost), 5 (Dynamic Reconfig), 6 (Exposed), 7 (Fractionally Exposed), 8 (Partially Exposed), 9 (Protection Disabled), 10 (Readying), 11 (Rebuild), 12 (Recalculate), 13 (Spare in Use), 14 (Verify In Progress) or 15 (In-Band Access Granted).
Primordial		Mandatory	This represents a Concrete Logical Disk that is not primordial.

**Table 196 - SMI Referenced Properties/Methods for CIM\_LogicalDisk (LD for FS)**

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	Identifier for a local LogicalDisk that will be used for a filesystem; since this logical disk will be referenced by a client, it must have a unique name. We cannot constrain the format here, but the OS-specific format described in the Block Services specification is not appropriate, so "Other" is used.
NameFormat		Mandatory	The format of the Name appropriate for LogicalDisks in the NAS Head. This shall be coded as "1" ("other").

**12.5.8 CIM\_StorageExtent (Primordial Imported Extent)**

Created By: Static\_or\_External

Modified By: External

Deleted By: External

Requirement: Optional

Table 197 describes class CIM\_StorageExtent (Primordial Imported Extent).

**Table 197 - SMI Referenced Properties/Methods for CIM\_StorageExtent (Primordial Imported Extent)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CreationClassName for the scoping system.
SystemName		Mandatory	The System Name of the scoping system.
CreationClassName		Mandatory	CreationClassName indicates the name of the class or the subclass.
DeviceID		Mandatory	An ID that uniquely names the StorageExtent in the NAS Head.
BlockSize		Mandatory	The size (in bytes) of blocks.
NumberOfBlocks		Mandatory	The number of Blocks from the imported StorageVolume.
ExtentStatus		Mandatory	This shall contain '16' (Imported).
OperationalStatus		Mandatory	Value shall be 2 3 6 8 15 (OK or Degraded or Error or Starting or Dormant).
Name		Mandatory	Deprecated. Identifier for a remote LUN on a storage array; possibly, the array ID plus LUN Node WWN.
Primordial		Mandatory	The StorageExtent imported from an Array is considered primordial in the NAS Head.

**12.5.9 CIM\_SystemDevice (Logical Disks)**

Created By: Extrinsic\_or\_External\_or\_Static

Modified By: Extrinsic\_or\_External

Deleted By: Extrinsic\_or\_External

Requirement: Mandatory

Table 198 describes class CIM\_SystemDevice (Logical Disks).

**Table 198 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Logical Disks)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this device.
PartComponent		Mandatory	The LogicalDisk that is a part of a computer system.

#### 12.5.10 CIM\_SystemDevice (Storage Extents)

Created By: Extrinsic\_or\_External\_or\_Static

Modified By: Extrinsic\_or\_External

Deleted By: Extrinsic\_or\_External

Requirement: This is required if primordial StorageExtents exist.

Table 199 describes class CIM\_SystemDevice (Storage Extents).

**Table 199 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Storage Extents)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this device.
PartComponent		Mandatory	The primordial StorageExtent that is imported to a computer system in the NAS Head.

**STABLE**

---



---

---

## STABLE

### 13 Self-Contained NAS Profile

#### 13.1 Description

##### 13.1.1 Synopsis

**Profile Name:** Self-contained NAS System (Autonomous Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** ComputerSystem

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 200 describes the related profiles for Self-contained NAS System.

**Table 200 - Related Profiles for Self-contained NAS System**

Profile Name	Organization	Version	Requirement	Description
Filesystem	SNIA	1.7.0	Mandatory	
File Storage	SNIA	1.4.0	Mandatory	
File Export	SNIA	1.7.0	Mandatory	
NAS Network Port	SNIA	1.7.0	Mandatory	
Access Points	SNIA	1.3.0	Optional	
Multiple Computer System	SNIA	1.2.0	Optional	
Software	SNIA	1.4.0	Optional	
Location	SNIA	1.4.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	
Filesystem Manipulation	SNIA	1.7.0	Optional	
File Export Manipulation	SNIA	1.7.0	Optional	
File Server Manipulation	SNIA	1.7.0	Optional	
Filesystem Performance	SNIA	1.7.0	Optional	Experimental.
FileSystem Quotas	SNIA	1.7.0	Optional	Experimental.
Filesystem Replication Services	SNIA	1.7.0	Optional	Experimental.
Job Control	SNIA	1.5.0	Optional	
Disk Drive Lite	SNIA	1.7.0	Optional	
FC Initiator Ports	SNIA	1.7.0	Optional	
iSCSI Initiator Ports	SNIA	1.2.0	Optional	
Device Credentials	SNIA	1.3.0	Optional	
Operational Power	SNIA	1.7.0	Optional	Experimental.

**Table 200 - Related Profiles for Self-contained NAS System**

Profile Name	Organization	Version	Requirement	Description
Launch In Context	DMTF	1.0.0	Optional	Experimental.
Physical Package	SNIA	1.5.0	Mandatory	
Block Services	SNIA	1.7.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

### 13.1.2 Overview

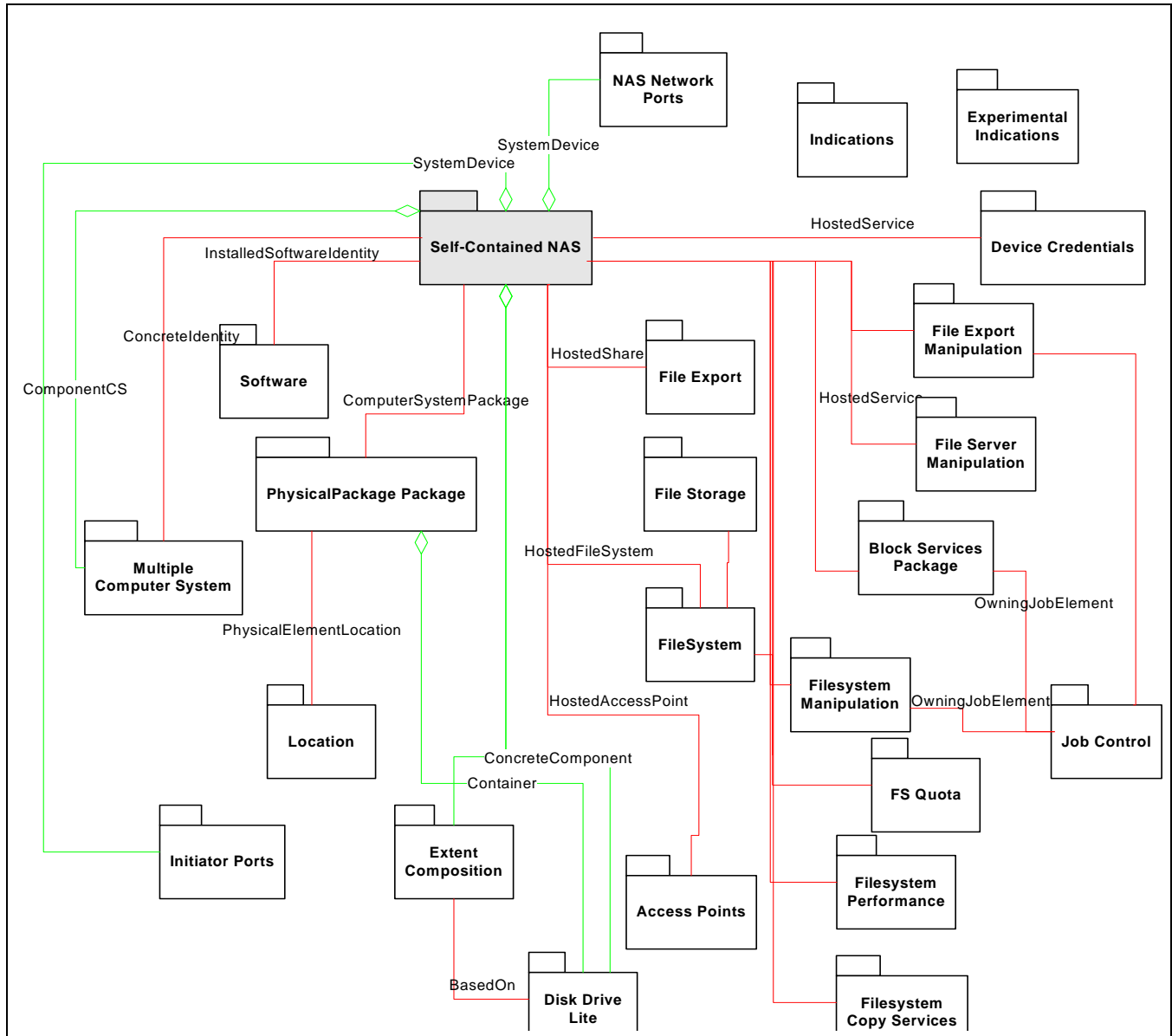
The Self-contained NAS (SC NAS) Profile exports File elements (contained in a filesystem) as FileShares. The storage for the filesystem is obtained from captive storage. In the simplest case, this could be a set of directly connected disks, but it could also be a captive storage array that is not shared with any other hosts or devices (though it could be visible to external management tools and even actively managed independently).

This profile models the necessary filesystem and NAS concepts and defines how the connections to the underlying storage is managed. The details of how a directly attached set of disks is used by the SC NAS Profile is covered as part of the Disk Drive Lite Profile. The details of how an underlying Storage Array might export storage to the SC NAS is not covered in this profile but is covered by 4 Array Profile in *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*.

The Self-Contained NAS Profile reuses a significant portion of 4 Array Profile in *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*.



The Self-Contained NAS Profile and its component profiles and packages are illustrated in Figure 21: "Self-Contained NAS Profiles".



### Figure 21 - Self-Contained NAS Profiles

### 13.1.3 Implementation

### 13.1.3.1 Summary Instance Diagram

Figure 22: "Self-Contained NAS Instance" illustrates the mandatory classes of the Self-Contained NAS Profile. This figure shows all the classes that are mandatory for the Self-contained NAS Profile. Later diagrams will review specific sections of this diagram

## Self-Contained NAS Profile

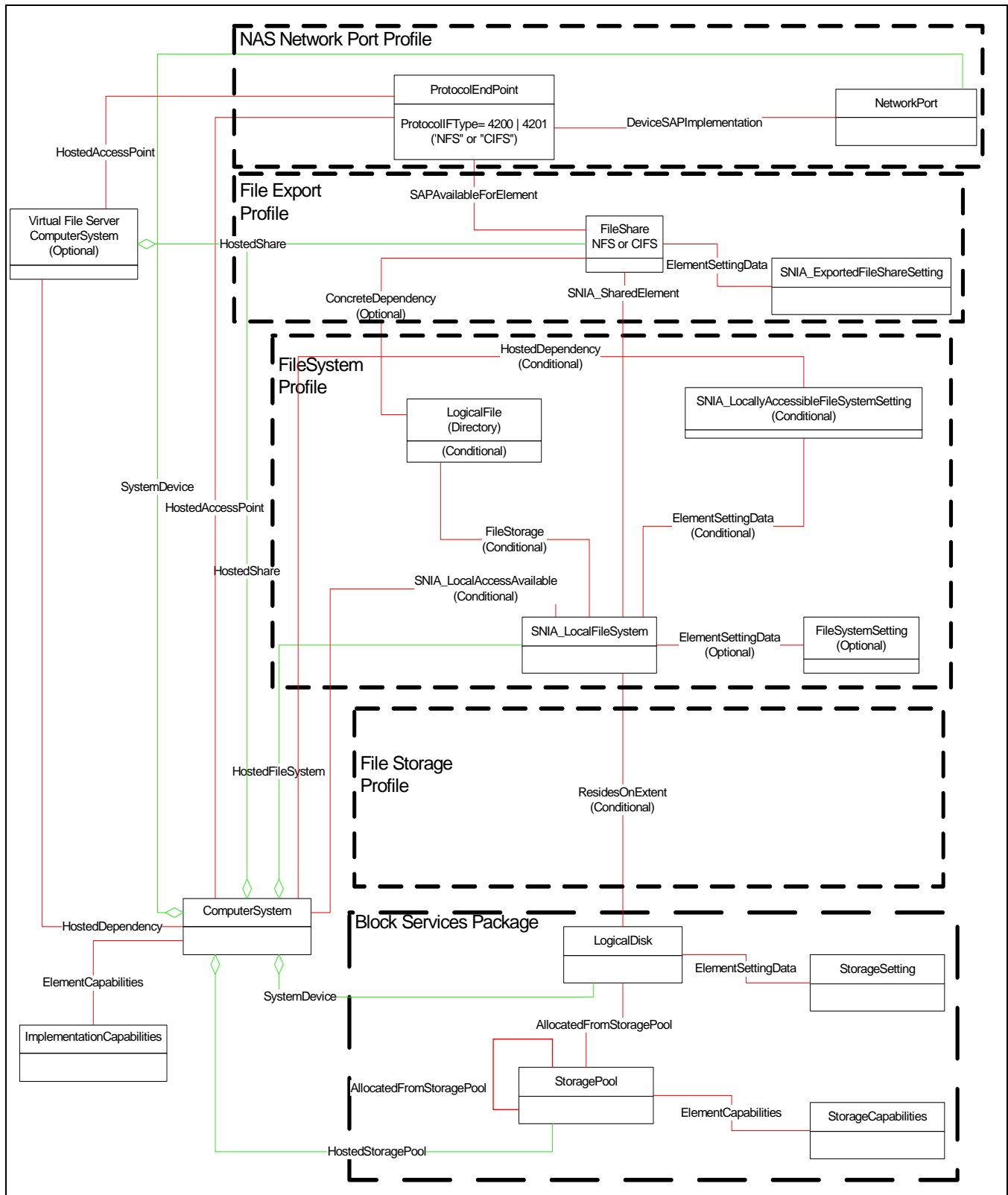


Figure 22 - Self-Contained NAS Instance

The Self-Contained NAS Profile closely parallels the Array Profile in how it models storage. Storage is assigned to StoragePools and LogicalDisks are allocated from those storage pools for the purpose of holding local filesystems of the NAS.

As with the Array Profile, the Self-contained NAS StoragePools have StorageCapabilities associated to the StoragePools via ElementCapabilities. Similarly, LogicalDisks have StorageSettings, which are associated to the LogicalDisk via ElementSettingData. StoragePools are hosted by a ComputerSystem that represents the NAS “top level” system, and the LogicalDisks have a SystemDevice association to the “top level” ComputerSystem.

NOTE As with Arrays, the StoragePools may be hosted by a component ComputerSystem if the profile has implemented the Multiple Computer System Profile.

As with Arrays, the “top level” ComputerSystem of the Self-Contained NAS does not (and typically isn’t) a real ComputerSystem. It is merely the ManagedElement upon which all aspects of the NAS offering are scoped.

A Self-Contained NAS may implement “Virtual File Servers” in addition to, or instead of, implementing File Servers in the Top Level ComputerSystem or one of the Multiple Computer System ComputerSystems. A Virtual File Server shall have a HostedDependency to either the top level NAS ComputerSystem or one of the Multiple Computer System ComputerSystems. NOTE: A Virtual File Server shall not have a ComponentCS association to the top level NAS ComputerSystem.

Everything above the LogicalDisk is specific to NAS (and does not appear in the Array Profile). LocalFileSystems are created on the LogicalDisks, LogicalFiles within those LocalFileSystems are shared (FileShare) through ProtocolEndpoints associated with NetworkPorts.

NOTE The classes and associations in the dashed boxes are from the required packages and component profiles (as indicated by the labels on the dashed boxes).

The ConcreteDependency association is provided for backward compatibility with SMI-S 1.1.0. It represents a relationship between a FileShare and a Directory.

The ResidesOnExtent is conditional on the use by NAS profiles (NAS Head and Self-contained NAS) in the File Storage Profile. In the Self-contained NAS a LocalFileSystem shall map to a LogicalDisk.

Also note that FileSystemSetting (and the corresponding ElementSettingData) are also optional. They are only shown here to illustrate where they would show up in the model should they be implemented.

In the base Self-Contained NAS Profile, the classes and associations shown in Figure 22: “Self-Contained NAS Instance” are automatically populated based on how the Self-Contained NAS is configured. Client modification of the configuration (including configuring storage, creating extents, local filesystems and file shares) are functions found in component profiles of the profile.

---

---

## EXPERIMENTAL

An instance of ImplementationCapabilities may be associated to the top level Self-contained NAS ComputerSystem. This Capabilities instance identifies the capacity optimization techniques supported by the implementation. An implementation may advertise that it supports “None”, “SNIA:Thin Provisioning”, “SNIA:Data Compression” or “SNIA:Data Deduplication”.

---

---

## EXPERIMENTAL

---



---

## EXPERIMENTAL

### 13.1.3.2 Combination Profile Considerations

Some devices combine the function of an array with the function of a Self-contained NAS. There are a number of approaches that may be used to model such a device. One way is to present two seemingly independent profiles in the SAN (e.g., Array and SC NAS). In this case, there may be duplication of instances. These duplicates would be recognized by clients via correlated ids.

Another approach would be to use one, shared top level ComputerSystem that reflects both the SC NAS and the Array in its ComputerSystem.Dedicated property. In this case, care must be taken to ensure the sharing of instances between the profiles do not conflict with their respective profile definitions.

For more information on the rules for combination profiles, see section A.6 of Annex A (normative) Compliance with the SNIA SMI Specification in *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*.

---

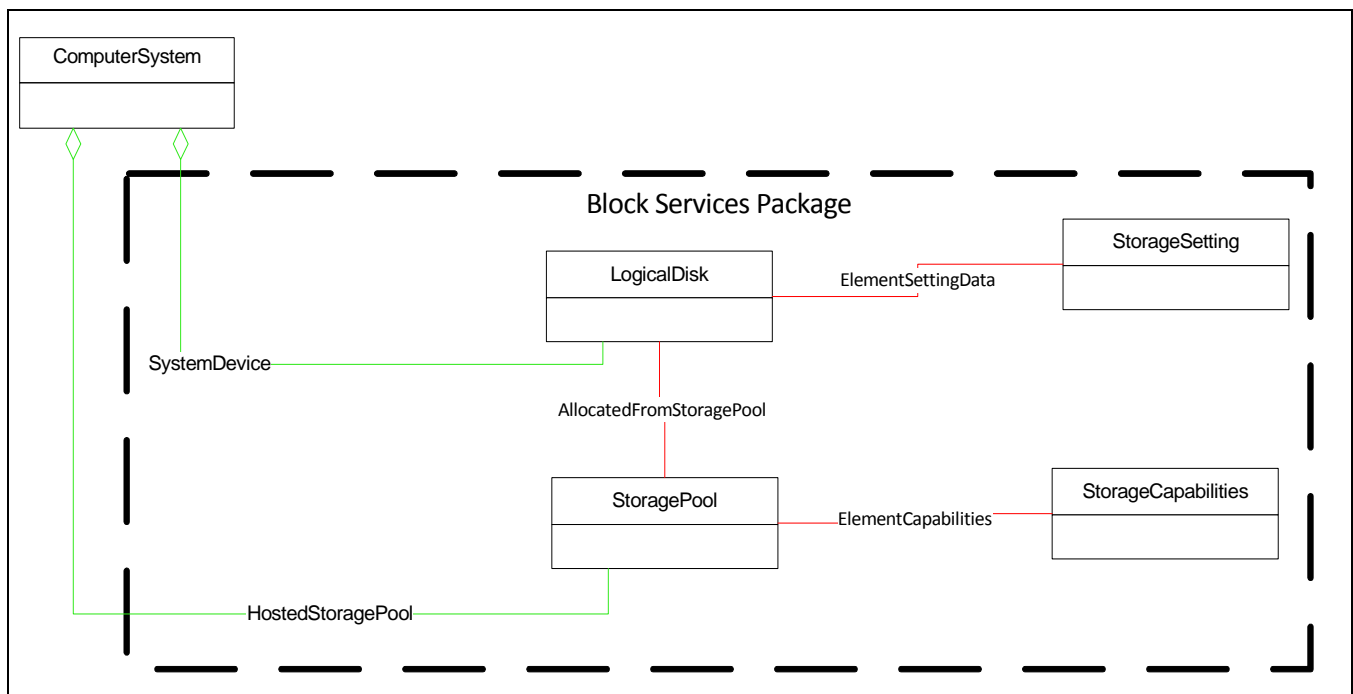


---

## EXPERIMENTAL

### 13.1.3.3 NAS Storage Model

Figure 23: "NAS Storage Instance" illustrates the classes mandatory for modeling of storage for the Self-Contained NAS Profile.



**Figure 23 - NAS Storage Instance**

The Self-Contained NAS Profile uses most of the classes and associations defined in the Array Profile (including those in the Block Services Package). In doing this, it leverages many of the component profiles that are available for Array profiles. The classes and associations shown in Figure 23: "NAS Storage Instance" are the minimum mandatory classes and associations of the Block Services Package for read only access in the base profile.

Storage for the NAS shall be modeled as logical storage. That is, StoragePools shall be modeled, including the HostedStoragePool and ElementCapabilities to the StorageCapabilities supported by the StoragePool. In addition, in order for storage to be used it shall be allocated to one or more LogicalDisks. A LogicalDisk shall have an AllocatedFromStoragePool association to the StoragePool from which it is allocated. And the LogicalDisk shall have an ElementSettingData association to the settings that were used when the LogicalDisk was created.

NOTE At this level, the model for storage is the same for both the Self-Contained NAS Profile and the NAS Head Profile. In the case of the Self-contained NAS, storage for the StoragePools is drawn from Disk Drives. Modeling of Disk Drives is Optional (See 10 Disk Drive Lite Profile of *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*).

For manipulation of Storage, see 5 Block Services Package in the *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*. For Self-Contained NAS, LogicalDisks are the ElementType that is supported for storage allocation functions (e.g., CreateOrModifyElementFromStoragePool and ReturnToStoragePool), but the Block Services methods for managing LogicalDisks are optional for the Self-Contained NAS Profile. The Self-Contained NAS Profile also supports (optionally) the Pool manipulation functions (e.g., CreateOrModifyStoragePool and DeleteStoragePool) of the Block Services Package.

#### 13.1.3.4 Self-Contained NAS Use of Filesystem Profile (Mandatory)

The Self-Contained NAS Profile uses the Filesystem Profile for modeling of its filesystem constructs. For the Self-Contained NAS, the Filesystem Profile shall be supported. See 8 Filesystem Profile for details on this modeling.

#### 13.1.3.5 Self-Contained NAS Use of File Storage Profile (Mandatory)

The Self-Contained NAS Profile uses the File Storage Profile for modeling of its file storage constructs. For the Self-Contained NAS, the Filesystem Profile shall be supported. See 7 File Storage Profile for details on the file storage modeling.

#### 13.1.3.6 Self-Contained NAS Use of File Export Profile (Mandatory)

The Self-Contained NAS Profile uses the File Export Profile for modeling of its file export constructs. For the Self-Contained NAS, the File Export Profile shall be supported. See 4 File Export Profile for details on this modeling.

#### 13.1.3.7 Self-Contained NAS Use of NAS Network Ports Profile (Mandatory)

The Self-Contained NAS Profile uses the NAS Network Ports Profile for modeling of its file export constructs. For the Self-Contained NAS, the NAS Network Ports Profile shall be supported. See NAS Network Port Profile (14) for details on this modeling.

#### 13.1.3.8 Indication Events

##### 13.1.3.8.1 InstModification of ComputerSystem

Table 201 identifies the standard OperationalStatus values and the events that are being indicated.

**Table 201 - InstModification Events for ComputerSystem**

New OperationalStatus	Event / Correlated Indications
OK	The top level NAS system is fully operational.
	An Error in the Top Level NAS system was corrected and the system is now fully functional.
	Self test is complete and the NAS system is fully operational
Degraded	The system is running but with limitations.
Error	The system is experiencing an error and is not functional.

**Table 201 - InstModification Events for ComputerSystem**

New OperationalStatus	Event / Correlated Indications
Stopped	The system has been stopped.
No contact	The system status cannot be determined, due to no response from the system.
Starting	The system is starting, but it not yet functional.
Stopping	The system is stopping.
Lost communication	The system status cannot be determined, due to communications problems.

**13.1.3.8.2 InstModification of LogicalDisk****EXPERIMENTAL**

Table 202 identifies the standard OperationalStatus values and the events that are being indicated.

**Table 202 - InstModification Events for LogicalDisk**

New OperationalStatus	Event / Correlated Indications
OK	The logical disk is fully functional.
Degraded	The logical disk is experiencing errors, but is still supporting data.
	The Logical disk is rebuilding, so performance may suffer.
Error	The logical disk has an error and is not usable.
Starting	The logical disk is being brought online.
Dormant	The logical disk is offline.

**EXPERIMENTAL****EXPERIMENTAL****13.1.3.9 Bellwether Indications****13.1.3.9.1 AlertIndication for ComputerSystem Bellwether**

This AlertIndication signals the change in status (OperationalStatus) of a ComputerSystem as a bellwether event. It is supported by a standard message (MessageID=FSM1). Table 203 shows the

OperationalStatus values that may signal that changes may have occurred in related elements (Implied Indications Inhibited).

**Table 203 - Bellwether AlertIndication Events for ComputerSystem**

New OperationalStatus	Implied Indications Inhibited
OK, Degraded, Error, Stopped	OperationalStatus changes to Elements with SystemDevice associations to this ComputerSystem (LogicalDisks, ...)
	OperationalStatus changes to Elements with HostedService associations to this ComputerSystem (FileSystemConfigurationService, FileExportService, ...)
	OperationalStatus changes to FileSystems with HostedFileSystem associations to this ComputerSystem.
	OperationalStatus changes to StoragePools with HostedStoragePool associations to this ComputerSystem.
	OperationalStatus changes to ProtocolEndpoints with HostedAccessPoint associations to this ComputerSystem.
	OperationalStatus changes to FileShares with HostedFileShare associations to this ComputerSystem.
No contact, Starting, Stopping, Lost communication	None

#### 13.1.3.9.2 AlertIndication for LogicalDisk Bellwether

This AlertIndication signals the change in status (OperationalStatus) of a LogicalDisk as a bellwether event. It is supported by a standard message (MessageID=FSM3). Table 204 shows the OperationalStatus values that may signal that changes may have occurred in related elements (Implied Indications Inhibited).

**Table 204 - Bellwether AlertIndication Events for LogicalDisk**

New OperationalStatus	Implied Indications Inhibited
OK, Degraded, Error, Stopped	OperationalStatus changes to FileSystems with ResidesOn associations to this LogicalDisk.
Unknown	None

## EXPERIMENTAL

### 13.2 Health and Fault Management Considerations

Self-Contained NAS supports state information (e.g., OperationalStatus) on the following elements of the model:

- Network Ports (See 14.3.1 "OperationalStatus for Network Ports")
- Back-end Ports (See 16.3.3 Health and Fault Management Considerations of *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*)
- ComputerSystems (See 22.2.5 Computer System Operational Status of *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*)
- FileShares that are exported (See 4.2.1 OperationalStatus for FileShares)
- LocalFileSystems (See 8.2.1 OperationalStatus for Filesystems)

- ProtocolEndpoints (See 14.3.2 OperationalStatus for ProtocolEndpoints)
- DiskDrive (See 10.3 Health and Fault Management Considerations of *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*)

---



---

## EXPERIMENTAL

### 13.2.1 Standard Messages used by this Profile

The standard messages specific to this profile are listed in Table 205.

**Table 205 - Standard Messages used by NAS Head**

Message ID	Message Name
FSM1	ComputerSystem bellwether alert
FSM3	LogicalDisk bellwether alert
Core12	Drive not responding
Core13	Fan Failure
Core14	Power Supply Failure
Core20	Drive is responding
Core21	Cooling Fan Issues Cleared
Core22	Power Supply Issues Cleared
Core23	Controller Failure
Core24	Controller Issues Cleared
DRM38	Volume or pool degraded
DRM39	Volume or pool failed
DRM40	Volume or pool issues cleared

---



---

## EXPERIMENTAL

### 13.3 Cascading Considerations

Not Applicable.

### 13.4 Methods of the Profile

#### 13.4.1 Extrinsic Methods of the Profile

Not defined in this standard.

#### 13.4.2 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames



- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

Manipulation functions are supported in component profiles of the profile.

### 13.5 Use Cases

Not defined in this standard.

### 13.6 CIM Elements

Table 206 describes the CIM elements for Self-contained NAS System.

**Table 206 - CIM Elements for Self-contained NAS System**

Element Name	Requirement	Description
13.6.1 CIM_ComputerSystem (Top Level System)	Mandatory	This declares that at least one computer system entry will pre-exist. The Name property should be the Unique identifier for the Self-contained NAS System. Associated to RegisteredProfile.
13.6.2 CIM_ComputerSystem (Virtual File Server)	Optional	This represents a Virtual File Server, if one exists.
13.6.3 CIM_ElementCapabilities (ImplementationCapabilities to Service)	Optional	Experimental. Associates the top level Self-contained NAS ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.
13.6.4 CIM_HostedDependency	Optional	Associates a Virtual File Server to the Computer System hosting it. This is required if a Virtual File Server exists.
13.6.5 CIM_ImplementationCapabilities (ImplementationCapabilities)	Optional	Experimental. The capabilities of the profile implementation.
13.6.6 CIM_LogicalDisk (Disk for FS)	Mandatory	Represents LogicalDisks used for building LocalFileSystems.
13.6.7 CIM_SystemDevice (Logical Disks)	Mandatory	This association links all LogicalDisks to the scoping system.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus <> PreviousInstance.CIM_ComputerSystem::OperationalStatus	Mandatory	CQL -Change of Status of a NAS ComputerSystem (controller).  PreviousInstance is optional, but may be supplied by an implementation of the Profile.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="SNIA" and MessageID="FSM1"	Optional	CQL -This is a bellwether indication of a change of Status of a NAS ComputerSystem (controller) and related classes (LogicalDisks, Services, ProtocolEndpoints, StoragePools, FileShares and FileSystems).  See 13.1.3.9.1 AlertIndication for ComputerSystem Bellwether
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus	Mandatory	CQL -Change of status of a LogicalDisk.  PreviousInstance is optional, but may be supplied by an implementation of the Profile.  See 13.1.3.8.2 InstModification of LogicalDisk.

**Table 206 - CIM Elements for Self-contained NAS System**

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="SNIA" and MessageID="FSM3"	Optional	CQL -This is a bellwether indication of a change of status of a LogicalDisk.  See 13.1.3.9.2 AlertIndication for LogicalDisk Bellwether
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core12'	Conditional	Experimental. If hardware is capable of supporting, indicates that a DiskDrive has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core20'	Conditional	Experimental. If hardware is capable of supporting, indicates that a DiskDrive has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core13'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Fan has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core21'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Fan has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core14'	Conditional	Experimental. If hardware is capable of supporting, indicates that a PowerSupply has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core22'	Conditional	Experimental. If hardware is capable of supporting, indicates that a PowerSupply has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core23'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Controller has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='Core24'	Conditional	Experimental. If hardware is capable of supporting, indicates that a Controller has been returned to service or has been replaced.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM39'	Mandatory	Experimental. A LogicalDisk has degraded.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM39'	Mandatory	Experimental. A LogicalDisk has failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='DRM40'	Mandatory	Experimental. A LogicalDisk has returned to normal service.

**13.6.1 CIM\_ComputerSystem (Top Level System)**

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Self-contained NAS System', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.6.0'.

Table 207 describes class CIM\_ComputerSystem (Top Level System).

**Table 207 - SMI Referenced Properties/Methods for CIM\_ComputerSystem (Top Level System)**

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	The actual class of this object, e.g., Vendor_NASComputerSystem.
ElementName		Mandatory	User-friendly name.
Name		Mandatory	Unique identifier for the Self-contained NAS System in a format specified by NameFormat. For example, IP address or Vendor/Model/SerialNo.
OperationalStatus		Mandatory	Overall status of the Self-contained NAS System. The standard values are 2 (OK), 3 (Degraded), 6 (Error), 8 (Starting), 9 (Stopping), 10 (Stopped), 12 (No contact) or 13 (Lost Communication).
NameFormat		Mandatory	Format for Name property.
PrimaryOwnerContact	M	Optional	Owner of the Self-contained NAS System.
PrimaryOwnerName	M	Optional	Contact details for owner.
Dedicated		Mandatory	This shall indicate that this computer system is dedicated to operation as a Self-contained NAS (25).
OtherIdentifyingInfo	C	Mandatory	An array of know identifiers for the Self-contained NAS.
IdentifyingDescriptions	C	Mandatory	An array of descriptions of the OtherIdentifyingInfo. Some of the descriptions would be "Ipv4 Address", "Ipv6 Address" or "Fully Qualified Domain Name".

### 13.6.2 CIM\_ComputerSystem (Virtual File Server)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 208 describes class CIM\_ComputerSystem (Virtual File Server).

**Table 208 - SMI Referenced Properties/Methods for CIM\_ComputerSystem (Virtual File Server)**

Properties	Flags	Requirement	Description & Notes
Dedicated		Mandatory	A Virtual File Server is a File Server (16).
NameFormat		Mandatory	Format for Name property. This shall be "Other".
Name	C	Mandatory	Unique identifier for the Self-contained NAS System's Virtual File Servers (E.g., Vendor/Model/SerialNo+FS+Number).
OperationalStatus		Mandatory	Overall status of the Virtual File Server. The standard values are 2 (OK), 3 (Degraded), 6 (Error), 8 (Starting), 9 (Stopping), 10 (Stopped), 12 (No contact) or 13 (Lost Communication).

### 13.6.3 CIM\_ElementCapabilities (ImplementationCapabilities to Service)

Experimental. Associates the top level Self-contained NAS ComputerSystem to the CIM\_ImplementationCapabilities supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static  
Requirement: Optional

#### 13.6.4 CIM\_HostedDependency

Created By: External  
Modified By: Static  
Deleted By: External  
Requirement: Optional

Table 209 describes class CIM\_HostedDependency.

**Table 209 - SMI Referenced Properties/Methods for CIM\_HostedDependency**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The Virtual File Server ComputerSystem. A Virtual File Server ComputerSystem is a File Server and shall have Dedicated=16 (File Server).
Antecedent		Mandatory	The hosting ComputerSystem. The hosting ComputerSystem may be the top level NAS ComputerSystem or an Multiple Computer System (non-top level) system.

#### 13.6.5 CIM\_ImplementationCapabilities (ImplementationCapabilities)

Experimental. The capabilities (features) of the profile implementation.

Created By: Static  
Modified By: Static  
Deleted By: Static  
Requirement: Optional

Table 210 describes class CIM\_ImplementationCapabilities (ImplementationCapabilities).

**Table 210 - SMI Referenced Properties/Methods for CIM\_ImplementationCapabilities (ImplementationCapabilities)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the implementation capability of an implementation.
ElementName		Optional	A provider supplied user-friendly name for this CIM_ImplementationCapabilities element.
SupportedCapacityOptimizations		Mandatory	This array of strings lists the capacity optimization techniques that are supported by the implementation. Valid string values are "none"   "SNIA:Thin Provisioning"   "SNIA:Data Compression"   "SNIA:Data Deduplication".

#### 13.6.6 CIM\_LogicalDisk (Disk for FS)

Created By: Extrinsic\_or\_External  
Modified By: Extrinsic\_or\_External  
Deleted By: Extrinsic\_or\_External

Requirement: Mandatory

Table 211 describes class CIM\_LogicalDisk (Disk for FS).

**Table 211 - SMI Referenced Properties/Methods for CIM\_LogicalDisk (Disk for FS)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	CIM Class of the Self-contained NAS System Computer System that is the host of this LogicalDisk.
SystemName		Mandatory	Name of the Self-contained NAS System Computer System that hosts this LogicalDisk.
CreationClassName		Mandatory	CIM Class of this instance of LogicalDisk.
DeviceID		Mandatory	Opaque identifier for the LogicalDisk.
OperationalStatus		Mandatory	A subset of operational status that is applicable for LogicalDisks in a Self-contained NAS System. The standard values for this are 2 (OK), 3 (Degraded), 6 (Error), 8 (Starting) or 15 (Dormant).
ExtentStatus		Mandatory	This LogicalDisk is neither imported (16) nor exported (17). The standard values for this are 0 (Other), 1 (Unknown), 2 (None/Not Applicable), 3 (Broken), 4 (Data Lost), 5 (Dynamic Reconfig), 6 (Exposed), 7 (Fractionally Exposed), 8 (Partially Exposed), 9 (Protection Disabled), 10 (Readying), 11 (Rebuild), 12 (Recalculate), 13 (Spare in Use), 14 (Verify In Progress) or 15 (In-Band Access Granted).
Primordial		Mandatory	This represents a Concrete Logical Disk that is not primordial.
NameFormat		Mandatory	The format of the Name appropriate for LogicalDisks in the Self-contained NAS System. This should be coded as "1" ("other").
Name		Mandatory	Identifier for a local LogicalDisk that will be used for a filesystem; since this storage extent will be referenced by a client, it needs to have a unique name. We cannot constrain the format here, but the OS-specific format described in the Block Services specification is not appropriate, so "Other" is used.

### 13.6.7 CIM\_SystemDevice (Logical Disks)

Created By: Extrinsic\_or\_External\_or\_Static

Modified By: Extrinsic\_or\_External

Deleted By: Extrinsic\_or\_External

Requirement: Mandatory

Table 212 describes class CIM\_SystemDevice (Logical Disks).

**Table 212 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Logical Disks)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this device. This shall be either the top level NAS system or a multiple computer system non-top level system.
PartComponent		Mandatory	The LogicalDisk that is a part of a computer system.

**STABLE**



---

---

**STABLE****14 NAS Network Port Profile****14.1 Description****14.1.1 Synopsis**

**Profile Name:** NAS Network Port (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** ProtocolEndpoint

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 213 describes the related profiles for NAS Network Port.

**Table 213 - Related Profiles for NAS Network Port**

Profile Name	Organization	Version	Requirement	Description
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

**14.1.2 Overview**

The NAS Network Port Profile models ProtocolEndpoints for file access (CIFS and NFS), TCP, IP and LAN endpoints. It also models the Network port supported by the protocol endpoints. The methods for manipulating these elements are covered by other profiles. This profile provides basic information in the NAS models for addressing paths for accessing the NAS implementations for the purpose of data access (front-end ports).

**14.2 Implementation**

Figure 24: "NAS Support for Front-end Network Ports" illustrates the classes for modeling of front end NetworkPorts for the NAS profiles.

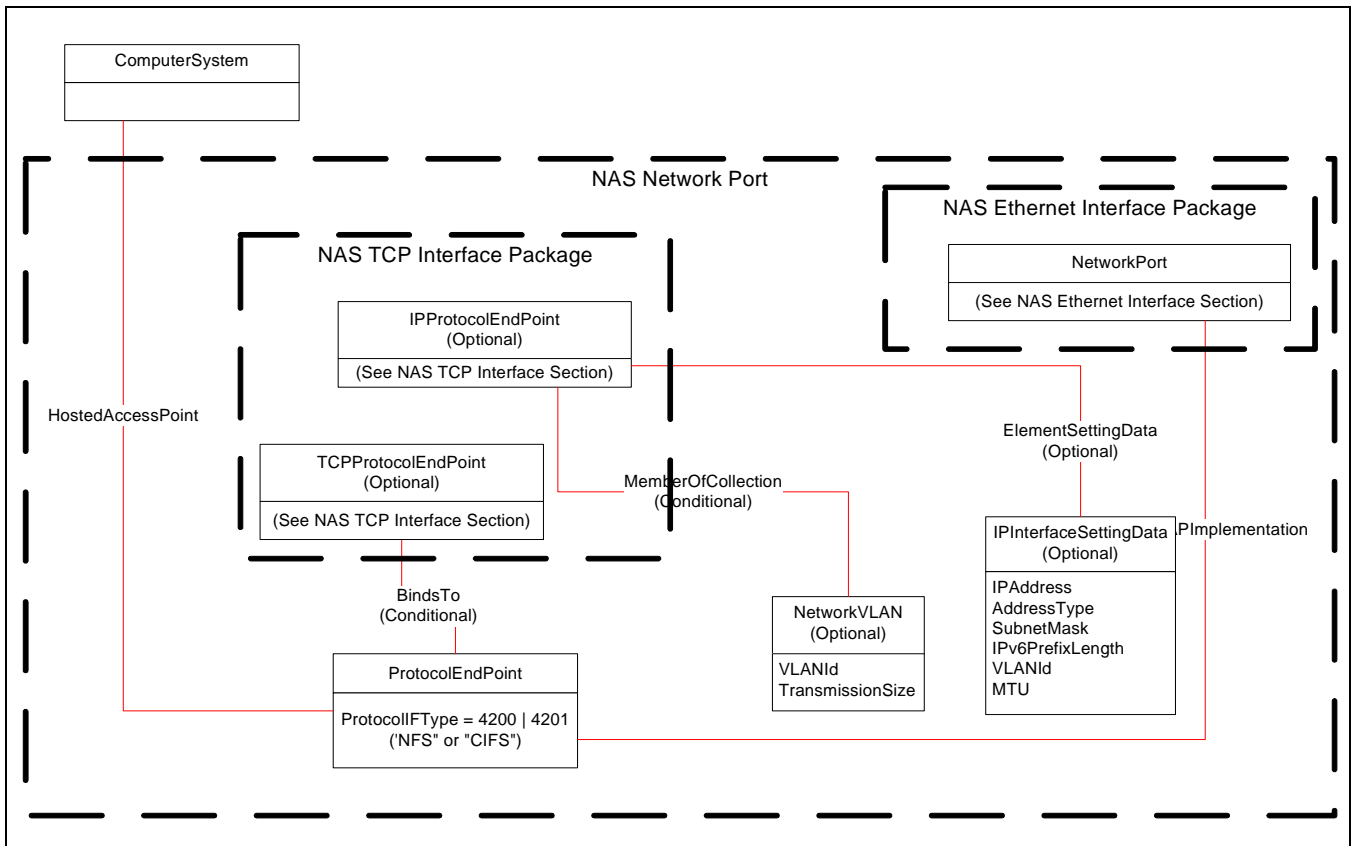


Figure 24 - NAS Support for Front-end Network Ports

The ProtocolEndPoint for NFS or CIFS shall be present and shall be associated to a ComputerSystem via a HostedAccessPoint association. It shall also be associated to one or more NetworkPort(s) via the DeviceSAPImplementation.

---

---

## EXPERIMENTAL

The NetworkVLAN and the CIM\_IPInterfaceSettingData classes are optional. And their associations to the IPProtocolEndPoint are conditional on the existence of these optional classes.

---

---

## EXPERIMENTAL

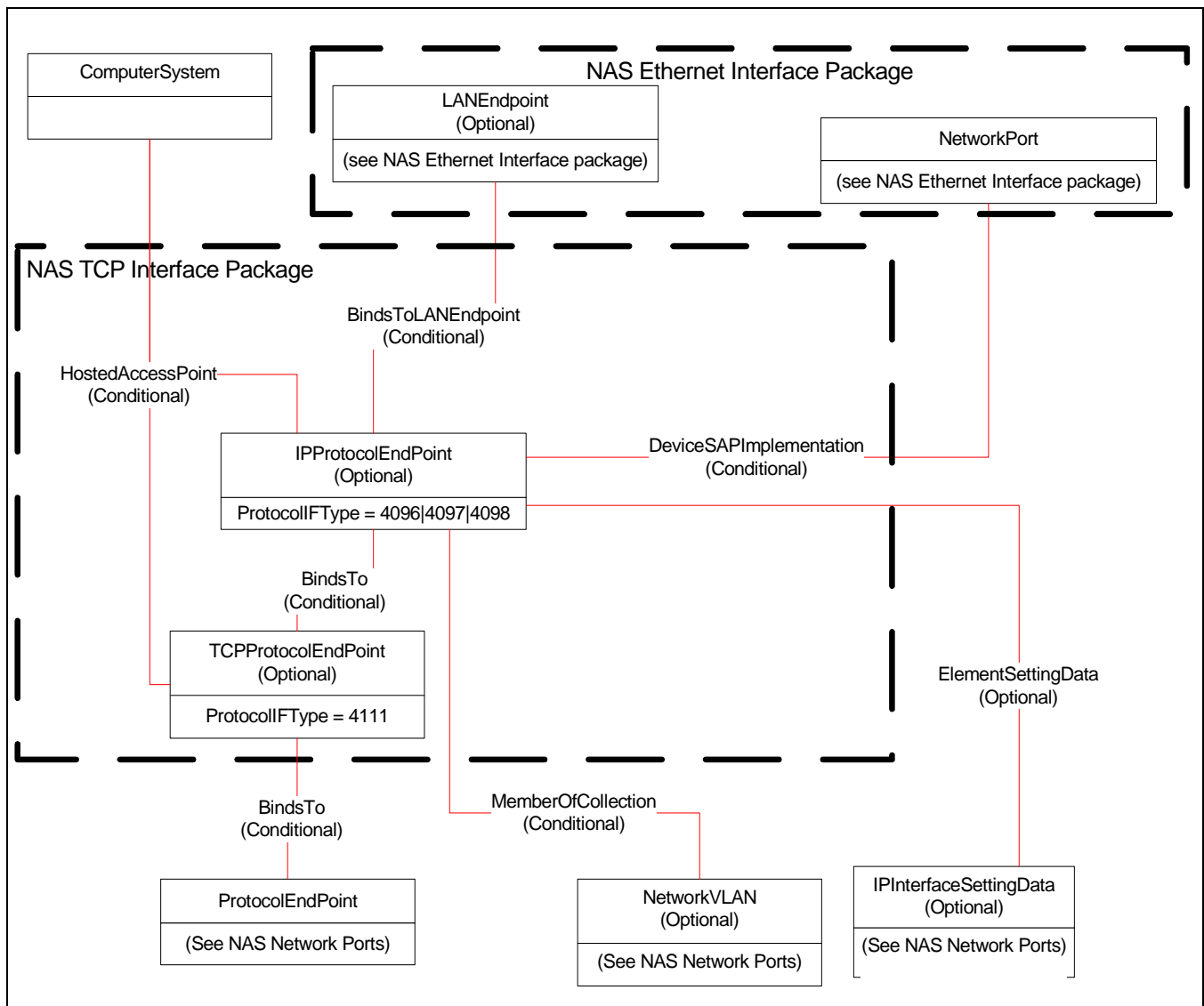
For TCP/IP Interface modeling (which is optional) see section 14.2.1

For modeling of Network ports for NAS (which is mandatory) see section 14.2.2

### 14.2.1 The NAS TCP Interface

Figure 25: "Optional NAS TCP Interface Modeling" illustrates the classes for the optional modeling of TCP/IP protocol stack for the NAS profiles.



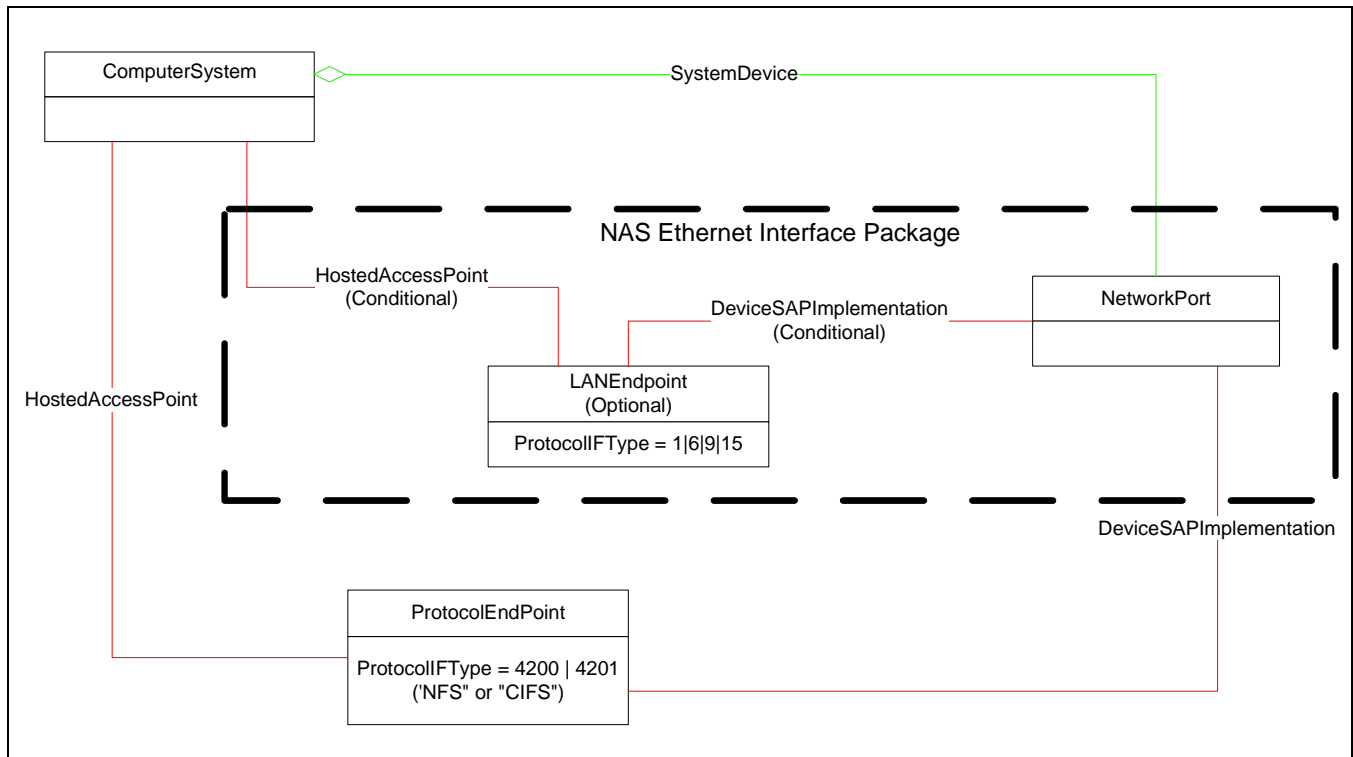


**Figure 25 - Optional NAS TCP Interface Modeling**

The modeling of the **TCPProtocolEndPoint** and **IPProtocolEndPoint** are optional. The associations from (to) those classes are conditional on the existence of the classes. Like the **NFS** or **CIFS** **ProtocolEndPoint**, the **TCPProtocolEndPoint** and **IPProtocolEndPoint** shall have **HostedAccessPoint** associations to some **ComputerSystem**. Typically, this would be the same **ComputerSystem** that hosts the **NetworkPort**. However this is not a requirement.

#### 14.2.2 The NAS Ethernet Interface

Figure 26: "Mandatory NAS Ethernet Port Modeling" illustrates the classes for the mandatory modeling of (front end) Network port for the NAS profiles.



**Figure 26 - Mandatory NAS Ethernet Port Modeling**

The **NetworkPort** shall be modeled and shall have an **SystemDevice** association to a **ComputerSystem**. The **ComputerSystem** in the diagram may be the top level system for the self-contained NAS or any of its component computer systems. The **ComputerSystem** that hosts the NFS or CIFS **ProtocolEndPoint** need not be the same **ComputerSystem** associated to the **NetworkPort** via its **SystemDevice** association.

The modeling of the **LANEndpoint** is optional. The associations from (to) this class are conditional on the existence of the **LANEndpoint**. Like the NFS or CIFS **ProtocolEndPoint**, the **LANEndpoint** shall have **HostedAccessPoint** associations to some **ComputerSystem**. Typically, this would be the same **ComputerSystem** that hosts the **NetworkPort**. However this is not a requirement.

### 14.2.3 Indication Events

#### 14.2.3.1 InstModification of NetworkPort

---



---

#### EXPERIMENTAL

Table 214 identifies the standard OperationalStatus values and the events that are being indicated.

**Table 214 - InstModification Events for NetworkPort**

New OperationalStatus	Event / Correlated Indications
OK	An Error in the port was corrected and the Port is now online
	The Port was enabled (and is online)
	Self test is complete and the port is back online
Error	The port has been unplugged
	The port is plugged in, but failed a self test
	The port is dependent on another element that has failed (e.g., a controller). CORRELATED INDICATION: InstModification of ComputerSystem
	The port is not able to establish connections to remote system
Stopped	The port is implicitly disabled due to a physical condition in the port
	The port is implicitly disabled due to a logical errors encountered on the port
	The port was explicitly disabled by user action
	The port was stopped due to a "parent" element (e.g., Controller) being stopped. CORRELATED INDICATION: InstModification of ComputerSystem
In Service	The port is in self test by explicit user request
	The port is in self test, due to errors encountered

---



---

#### EXPERIMENTAL

#### 14.2.3.2 InstModification of ProtocolEndpoint

---



---

#### EXPERIMENTAL

Table 215 identifies the standard OperationalStatus values and the events that are being indicated.

**Table 215 - InstModification Events for ProtocolEndpoint**

New OperationalStatus	Event / Correlated Indications
OK	ProtocolEndpoint is online
Error	ProtocolEndpoint has a failure
Stopped	ProtocolEndpoint is disabled
Unknown	

---



---

## EXPERIMENTAL

---



---

## EXPERIMENTAL

### 14.2.4 Bellwether Indications

#### 14.2.4.1 AlertIndication for NetworkPort Bellwether

This AlertIndication signals the change in status (OperationalStatus) of a NetworkPort as a bellwether event. It is supported by a standard message (MessageID=FSM002). Table 216 shows the OperationalStatus values that may signal that changes may have occurred in related elements (Implied Indications Inhibited).

**Table 216 - Bellwether AlertIndication Events for NetworkPort**

New OperationalStatus	Implied Indications Inhibited
OK, Error, Stopped	OperationalStatus changes to ProtocolEndpoints with DeviceSAPImplementation associations to this NetworkPort.
	OperationalStatus changes to FileShares with SAPAvailableForFileShare associations to a ProtocolEndpoint with a DeviceSAPImplementation association to this NetworkPort.
InService, Unknown	None

---



---

## EXPERIMENTAL

---



---

### 14.3 Health and Fault Management Considerations

The NAS Network Port Profile supports state information (e.g., OperationalStatus) on the following elements of the model:

- Network Ports (See 14.3.1 OperationalStatus for Network Ports)
- ProtocolEndpoints (See 14.3.2 OperationalStatus for ProtocolEndpoints)

### 14.3.1 OperationalStatus for Network Ports

Table 217 defines the network port OperationalStatus values supported by this standard.

**Table 217 - NetworkPort OperationalStatus**

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

### 14.3.2 OperationalStatus for ProtocolEndpoints

Table 217 defines the ProtocolEndpoint OperationalStatus values supported by this standard

**Table 218 - ProtocolEndpoint OperationalStatus**

OperationalStatus	Description
OK	ProtocolEndpoint is online
Error	ProtocolEndpoint has a failure
Stopped	ProtocolEndpoint is disabled
Unknown	

---

## EXPERIMENTAL

### 14.3.3 Standard Messages used by this Profile

The standard messages specific to this profile are listed in Table 219.

**Table 219 - Standard Messages used by NAS Head**

Message ID	Message Name
FSM002	NetworkPort bellwether alert

---

## EXPERIMENTAL

### 14.4 Cascading Considerations

Not Applicable.

### 14.5 Methods

#### 14.5.1 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators

- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

#### 14.5.2 Extrinsic Methods of the Profile

None. For creation of ProtocolEndpoints, see 5 File Export Manipulation Profile and 6 File Server Manipulation Profile.

#### 14.6 Use Cases

Not defined in this standard.

#### 14.7 CIM Elements

Table 220 describes the CIM elements for NAS Network Port.

**Table 220 - CIM Elements for NAS Network Port**

Element Name	Requirement	Description
14.7.1 CIM_BindsTo (CIFS or NFS)	Conditional	Conditional requirement: This is required if a TCPProtocolEndpoint exists. Associates a CIFS or NFS ProtocolEndpoint to an underlying TCPProtocolEndpoint. This is used in the NAS profiles to support the TCP/IP Network protocol stack.
14.7.2 CIM_BindsTo (TCP)	Conditional	Conditional requirement: This is required if an IPProtocolEndpoint exists. (See the TCP Interface Section) Associates a TCPProtocolEndpoint to an underlying IPProtocolEndpoint. This is used in the NAS Profiles to support the TCP/IP Network protocol stack.
14.7.3 CIM_BindsToLANEndpoint	Conditional	Conditional requirement: This is required if a LANEndpoint exists. (See the TCP Interface Section) Associates an IPProtocolEndpoint to an underlying LANEndpoint in the NAS Profiles (to support the TCP/IP Network protocol stack).
14.7.4 CIM_DeviceSAPImplementation (CIFS or NFS to NetworkPort)	Mandatory	Represents the association between a CIFS or NFS ProtocolEndpoint and the NetworkPort that it supports.
14.7.5 CIM_DeviceSAPImplementation (LANEndpoint to NetworkPort)	Conditional	Conditional requirement: This is required if a LANEndpoint exists. (See the Ethernet Interface Section) Associates a logical front end Port (a NetworkPort) to the LANEndpoint that uses that device to connect to a LAN.
14.7.6 CIM_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint)	Optional	The IPProtocolEndpoint associated with the IPInterfaceSettingData.
14.7.7 CIM_HostedAccessPoint (CIFS or NFS)	Mandatory	Represents the association between a CIFS or NFS front end ProtocolEndpoint and the Computer System that hosts it.

**Table 220 - CIM Elements for NAS Network Port**

Element Name	Requirement	Description
14.7.8 CIM_HostedAccessPoint (IP)	Conditional	Conditional requirement: This is required if an IPProtocolEndpoint exists. (See the TCP Interface Section) Represents the association between a front end IPProtocolEndpoint and the Computer System that hosts it.
14.7.9 CIM_HostedAccessPoint (LAN)	Conditional	Conditional requirement: This is required if a LANEndpoint exists. (See the Ethernet Interface Section) Represents the association between a front end LANEndpoint and the Computer System that hosts it.
14.7.10 CIM_HostedAccessPoint (TCP)	Conditional	Conditional requirement: This is required if a TCPProtocolEndpoint exists. (See the TCP Interface Section) Represents the association between a front end TCPProtocolEndpoint and the Computer System that hosts it.
14.7.12 CIM_IPProtocolEndpoint	Optional	(See the TCP Interface Section) Represents the front-end ProtocolEndpoint used to support the IP protocol services.
14.7.13 CIM_LANEndpoint	Optional	(See the Ethernet Interface Section) Represents the front-end ProtocolEndpoint used to support a Local Area Network and its services.
14.7.14 CIM_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.)	Conditional	Conditional requirement: The NetworkVLAN has been defined. Associates an IPProtocolEndpoint to NetworkVLAN.
14.7.15 CIM_NetworkPort	Mandatory	(See the Ethernet Interface Section) Represents the front end logical port that supports access to a local area network.
14.7.16 CIM_NetworkVLAN	Optional	This element represents the virtual LAN (VLAN) tag settings for an IP interface. In the context of a file server, it represents the VLAN information.
14.7.17 CIM_ProtocolEndpoint (CIFS or NFS)	Mandatory	Represents the front-end ProtocolEndpoint used to support NFS and CIFS services.
14.7.18 CIM_SystemDevice (Network Ports)	Mandatory	(See the Ethernet Interface section) This association links all NetworkPorts to the scoping system. This is used to represent both front end and back end ports.
14.7.19 CIM_TCPProtocolEndpoint	Optional	(See the TCP Interface Section) Represents the front-end ProtocolEndpoint used to support TCP services.
14.7.11 CIM_IPInterfaceSettingData	Optional	This class contains the settings for single IP interface.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_NetworkPort AND SourceInstance.CIM_NetworkPort::OperationalStatus <> PreviousInstance.CIM_NetworkPort::OperationalStatus	Mandatory	CQL -Change of Status of a Port.  PreviousInstance is optional, but may be supplied by an implementation of the Profile.  See 14.2.3.1 InstModification of NetworkPort.

**Table 220 - CIM Elements for NAS Network Port**

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity="SNIA" and MessageID="FSM2"	Optional	CQL -This is a bellwether indication of a change of Status of a Port and related classes (ProtocolEndpoints and FileShares).  See 14.2.4.1 AlertIndication for NetworkPort Bellwether.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ProtocolEndpoint AND SourceInstance.CIM_ProtocolEndpoint::OperationalStatus <> PreviousInstance.CIM_ProtocolEndpoint::OperationalStatus	Mandatory	CQL -Change of Status of a ProtocolEndpoint  PreviousInstance is optional, but may be supplied by an implementation of the Profile.  See 14.2.3.2 InstModification of ProtocolEndpoint.

**14.7.1 CIM\_BindsTo (CIFS or NFS)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: This is required if a TCPProtocolEndpoint exists.

Table 221 describes class CIM\_BindsTo (CIFS or NFS).

**Table 221 - SMI Referenced Properties/Methods for CIM\_BindsTo (CIFS or NFS)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The ProtocolEndpoint that uses a lower level ProtocolEndpoint for connectivity. The ProtocolIFType shall be 4200 (NFS) or 4201 (CIFS) in the referenced ProtocolEndpoint.
Antecedent		Mandatory	The TCPProtocolEndpoint that supports a CIFS or NFS ProtocolEndpoint.

**14.7.2 CIM\_BindsTo (TCP)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: This is required if an IPProtocolEndpoint exists.

Table 222 describes class CIM\_BindsTo (TCP).

**Table 222 - SMI Referenced Properties/Methods for CIM\_BindsTo (TCP)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The TCPProtocolEndpoint that uses an IPProtocolEndpoint for connectivity.
Antecedent		Mandatory	The IPProtocolEndpoint that supports a TCPProtocolEndpoint.

**14.7.3 CIM\_BindsToLANEndpoint**

Created By: External



Modified By: External

Deleted By: External

Requirement: This is required if a LANEndpoint exists.

Table 223 describes class CIM\_BindsToLANEndpoint.

**Table 223 - SMI Referenced Properties/Methods for CIM\_BindsToLANEndpoint**

Properties	Flags	Requirement	Description & Notes
FrameType		Mandatory	Only supports 1="Ethernet" at this point.
Dependent		Mandatory	A IPProtocolEndpoint.
Antecedent		Mandatory	A LANEndpoint.

#### 14.7.4 CIM\_DeviceSAPImplementation (CIFS or NFS to NetworkPort)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Mandatory

Table 224 describes class CIM\_DeviceSAPImplementation (CIFS or NFS to NetworkPort).

**Table 224 - SMI Referenced Properties/Methods for CIM\_DeviceSAPImplementation (CIFS or NFS to NetworkPort)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The ProtocolEndpoint that supports on the NetworkPort. The ProtocolIFType shall be 4200 (NFS) or 4201 (CIFS) in the referenced ProtocolEndpoint.
Antecedent		Mandatory	The NetworkPort supported by the Access Point.

#### 14.7.5 CIM\_DeviceSAPImplementation (LANEndpoint to NetworkPort)

Created By: External

Modified By: Static

Deleted By: External

Requirement: This is required if a LANEndpoint exists.

Table 225 describes class CIM\_DeviceSAPImplementation (LANEndpoint to NetworkPort).

**Table 225 - SMI Referenced Properties/Methods for CIM\_DeviceSAPImplementation (LANEndpoint to NetworkPort)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	A LANEndpoint that depends on a NetworkPort for connecting to its LAN segment.
Antecedent		Mandatory	The Logical network adapter device that connects to a LAN.

**14.7.6 CIM\_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: Optional

Table 226 describes class CIM\_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint).

**Table 226 - SMI Referenced Properties/Methods for CIM\_ElementSettingData (IPInterfaceSettingData to IPProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The IPProtocolEndpoint.
SettingData		Mandatory	The IPInterfaceSettingData.

**14.7.7 CIM\_HostedAccessPoint (CIFS or NFS)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: Mandatory

Table 227 describes class CIM\_HostedAccessPoint (CIFS or NFS).

**Table 227 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint (CIFS or NFS)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The ServiceAccessPoint hosted on the file server. The ProtocolIFType shall be 4200 (NFS) or 4201 (CIFS) in the referenced ProtocolEndpoint.
Antecedent		Mandatory	The Computer System hosting this Access Point. In the context of the NAS Profiles, these are always file servers (Dedicated=16).

**14.7.8 CIM\_HostedAccessPoint (IP)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: This is required if an IPProtocolEndpoint exists.

Table 228 describes class CIM\_HostedAccessPoint (IP).

**Table 228 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint (IP)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The IPProtocolEndpoint hosted on the file server.
Antecedent		Mandatory	The Computer System hosting this Access Point.

**14.7.9 CIM\_HostedAccessPoint (LAN)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: This is required if a LANEndpoint exists.

Table 229 describes class CIM\_HostedAccessPoint (LAN).

**Table 229 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint (LAN)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The LANEndpoint hosted on the file server.
Antecedent		Mandatory	The Computer System hosting this Access Point.

**14.7.10 CIM\_HostedAccessPoint (TCP)**

Created By: External

Modified By: Static

Deleted By: External

Requirement: This is required if a TCPProtocolEndpoint exists.

Table 230 describes class CIM\_HostedAccessPoint (TCP).

**Table 230 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint (TCP)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The TCPProtocolEndpoint hosted on the file server.
Antecedent		Mandatory	The Computer System hosting this Access Point.

**14.7.11 CIM\_IPInterfaceSettingData**

Created By: Extrinsic: CreateFileServer | AddIPInterface

Modified By: Extrinsic: ModifyIPInterface

Deleted By: Extrinsic: DeleteFileServer | DeleteIPInterface

Requirement: Optional

Table 231 describes class CIM\_IPInterfaceSettingData.

**Table 231 - SMI Referenced Properties/Methods for CIM\_IPInterfaceSettingData**

Properties	Flags	Requirement	Description & Notes
IPAddress		Mandatory	The IPAddress that will be used by the File Server. This can be either an IPv4 or IPv6 address.
AddressType		Mandatory	The IPAddress format. This can be either "IPv4" or "IPv6".
SubnetMask		Conditional	Conditional requirement: This is required if an ProtocolIFType = 4096 or 4098. The subnet mask that will be used by the File Server.

**Table 231 - SMI Referenced Properties/Methods for CIM\_IPInterfaceSettingData**

Properties	Flags	Requirement	Description & Notes
IPv6PrefixLength		Conditional	Conditional requirement: This is required if an ProtocolIFType = 4097 or 4098. If AddressType specifies IPv6, then this specifies the prefix length for the IPv6 address in IPAddress.
VLANId		Optional	If present contains the ID of the VLAN that this IP setting will be associated with.
MTU		Optional	If present contains the maximum transmission unit to be used for this IP setting. If not present, then the default of 1500 will be used.

**14.7.12CIM\_IPProtocolEndpoint**

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 232 describes class CIM\_IPProtocolEndpoint.

**Table 232 - SMI Referenced Properties/Methods for CIM\_IPProtocolEndpoint**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the Protocol Endpoint.
SystemName		Mandatory	The name of the Computer System hosting the Protocol Endpoint.
CreationClassName		Mandatory	The CIM Class name of the Protocol Endpoint.
Name		Mandatory	The unique name of the Protocol Endpoint.
NameFormat		Mandatory	(POSSIBLE NAS CONSTRAINT) The Format of the Name.
RequestedState		Optional	(DMTF Core/IP Interface).
OperationalStatus		Conditional	Conditional requirement: Required if parent profile is NAS Head. The operational status of the PEP.
EnabledState		Optional	(DMTF Core/IP Interface).
OtherEnabledState		Optional	
TimeOfLastStateChange		Optional	
Description		Conditional	Conditional requirement: Required if parent profile is NAS Head. or Required if parent profile is a Self-contained NAS System. This shall be the IP protocol endpoints supported by the NAS Profiles.
ProtocolIFType		Mandatory	4096="IP v4", 4097="IP v6", and 4098 is both. (Note that 1="Other" is not supported).
IPv4Address		Conditional	Conditional requirement: This is required if an ProtocolIFType = 4096 or 4098. An IP v4 address in the format "A.B.C.D".
IPv6Address		Conditional	Conditional requirement: This is required if an ProtocolIFType = 4097 or 4098. An IP v6 address.

**Table 232 - SMI Referenced Properties/Methods for CIM\_IPProtocolEndpoint**

Properties	Flags	Requirement	Description & Notes
SubnetMask		Conditional	Conditional requirement: This is required if an ProtocolIFType = 4096 or 4098. An IP v4 subnet mask in the format "A.B.C.D".
PrefixLength		Conditional	Conditional requirement: This is required if an ProtocolIFType = 4097 or 4098. For an IPv6 address.

**14.7.13CIM\_LANEndpoint**

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 233 describes class CIM\_LANEndpoint.

**Table 233 - SMI Referenced Properties/Methods for CIM\_LANEndpoint**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the Protocol Endpoint.
SystemName		Mandatory	The name of the Computer System hosting the Protocol Endpoint.
CreationClassName		Mandatory	The CIM Class name of the Protocol Endpoint.
Name		Conditional	Conditional requirement: Required if parent profile is a Self-contained NAS System. or Required if parent profile is NAS Head. The unique name of the Protocol Endpoint.
NameFormat		Conditional	Conditional requirement: Required if parent profile is a Self-contained NAS System. or Required if parent profile is NAS Head. The Format of the Name.
RequestedState		Optional	A NAS Head option.
OperationalStatus		Conditional	Conditional requirement: Required if parent profile is NAS Head. The operational status of the PEP.
EnabledState		Optional	A NAS Head option.
OtherEnabledState		Optional	A NAS Head option.
TimeOfLastStateChange		Optional	A NAS Head option.
Description		Conditional	Conditional requirement: Required if parent profile is NAS Head. This shall be the LAN protocol endpoints supported by the NAS Head.
ProtocolIFType		Conditional	Conditional requirement: Required if parent profile is a Self-contained NAS System. or Required if parent profile is NAS Head. LAN endpoints supported are: 1="Other",6="Ethernet CSMA/CD", 9="ISO 802.5 Token Ring", 15="FDDI".
OtherTypeDescription		Optional	If the LAN endpoint is a vendor-extension specified by "Other" and a description.
LANID	N	Optional	A unique id for the LAN segment to which this device is connected. The value will be NULL if the LAN is not connected.
MACAddress		Mandatory	(POSSIBLE NAS CONSTRAINT) Primary Unicast address for this LAN device.

**Table 233 - SMI Referenced Properties/Methods for CIM\_LANEndpoint**

Properties	Flags	Requirement	Description & Notes
AliasAddresses		Mandatory	(POSSIBLE NAS CONSTRAINT) Other unicast addresses supported by this device.
GroupAddresses		Mandatory	(POSSIBLE NAS CONSTRAINT) Multicast addresses supported by this device.
MaxDataSize		Mandatory	(POSSIBLE NAS CONSTRAINT) The max size of packet supported by this LAN device.

**14.7.14 CIM\_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: The NetworkVLAN has been defined.

Table 234 describes class CIM\_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.).

**Table 234 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (The IPProtocolEndpoint to NetworkVLAN.)**

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	The IPProtocolEndpoint.
Collection		Mandatory	The NetworkVLAN.

**14.7.15 CIM\_NetworkPort**

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 235 describes class CIM\_NetworkPort.

**Table 235 - SMI Referenced Properties/Methods for CIM\_NetworkPort**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	A user-friendly name for this Network adapter that provides a network port.
OperationalStatus		Mandatory	The operational status of the adapter.
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the Network Port.
SystemName		Mandatory	The name of the Computer System hosting the Network Port.
CreationClassName		Mandatory	The CIM Class name of the Network Port.
DeviceID		Mandatory	A unique ID for the device (in the context of the hosting System).
Speed		Optional	(Fabric/Extender).
MaxSpeed		Optional	(Fabric/Extender).
RequestedSpeed		Optional	

**Table 235 - SMI Referenced Properties/Methods for CIM\_NetworkPort**

Properties	Flags	Requirement	Description & Notes
UsageRestriction		Optional	
PortType		Optional	(Fabric/Extender).
PortNumber		Optional	(Fabric/Extender) A unique number for the adapter in the context of the hosting System.
PermanentAddress	C	Mandatory	The hard-coded address of this port.
NetworkAddresses		Conditional	Conditional requirement: Required if parent profile is NAS Head. An array of network addresses for this port.
LinkTechnology		Optional	(Fabric/Extender) 1="Other", 2="Ethernet", 3="IB", 4="FC", 5="FDDI", 6="ATM", 7="Token Ring", 8="Frame Relay", 9="Infrared", 10="BlueTooth", 11="Wireless LAN. The link technology supported by this adapter.
OtherLinkTechnology		Optional	The vendor-specific "Other" link technology supported by this adapter.
FullDuplex		Optional	
AutoSense		Optional	
SupportedMaximumTransmissionUnit		Optional	
ActiveMaximumTransmissionUnit		Optional	

**14.7.16 CIM\_NetworkVLAN**

Created By: Extrinsic

Modified By: Extrinsic

Deleted By: Extrinsic

Requirement: Optional

Table 236 describes class CIM\_NetworkVLAN.

**Table 236 - SMI Referenced Properties/Methods for CIM\_NetworkVLAN**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the NetworkVLAN.
VLANId		Mandatory	The VLAN id that is to be associated with an IP interface. The id shall be included in all IP packets being sent through an IP interface.
TransmissionSize		Mandatory	The maximum transmission unit size that is associated with an IP Interface.

**14.7.17 CIM\_ProtocolEndpoint (CIFS or NFS)**

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 237 describes class CIM\_ProtocolEndpoint (CIFS or NFS).

**Table 237 - SMI Referenced Properties/Methods for CIM\_ProtocolEndpoint (CIFS or NFS)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the Protocol Endpoint.
SystemName		Mandatory	The name of the Computer System hosting the Protocol Endpoint.
CreationClassName		Mandatory	The CIM Class name of the Protocol Endpoint.
Name		Mandatory	The unique name of the Protocol Endpoint.
NameFormat		Mandatory	The Format of the Name.
RequestedState		Optional	
OperationalStatus		Mandatory	The operational status of the PEP.
EnabledState		Optional	
OtherEnabledState		Optional	
TimeOfLastStateChange		Optional	
Description		Mandatory	This shall be one of the NFS or CIFS protocol endpoints supported by the NAS Profiles.
ProtocolIFTType		Mandatory	This represents either NFS=4200 or CIFS=4201. Other protocol types are specified in subclasses of ProtocolEndpoint.

#### 14.7.18 CIM\_SystemDevice (Network Ports)

Created By: Extrinsic\_or\_External\_or\_Static

Modified By: Extrinsic\_or\_External

Deleted By: Extrinsic\_or\_External

Requirement: Mandatory

Table 238 describes class CIM\_SystemDevice (Network Ports).

**Table 238 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Network Ports)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this device. This shall be either the top level NAS system or a multiple computer system non-top level system.
PartComponent		Mandatory	The NetworkPort that is a part of a computer system.

#### 14.7.19 CIM\_TCPProtocolEndpoint

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional



Table 239 describes class CIM\_TCPProtocolEndpoint.

**Table 239 - SMI Referenced Properties/Methods for CIM\_TCPProtocolEndpoint**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the Protocol Endpoint.
SystemName		Mandatory	The name of the Computer System hosting the Protocol Endpoint.
CreationClassName		Mandatory	The CIM Class name of the Protocol Endpoint.
Name		Mandatory	The unique name of the Protocol Endpoint.
NameFormat		Conditional	Conditional requirement: Required if parent profile is a Self-contained NAS System. or Required if parent profile is NAS Head. The Format of the Name.
RequestedState		Optional	A NAS Head option.
OperationalStatus		Conditional	Conditional requirement: Required if parent profile is NAS Head. The operational status of the PEP.
EnabledState		Optional	A NAS Head option.
OtherEnabledState		Optional	A NAS Head option.
TimeOfLastStateChange		Optional	A NAS Head option.
Description		Conditional	Conditional requirement: Required if parent profile is NAS Head. This shall be the TCP protocol endpoints supported by the NAS Head.
ProtocolIFType		Mandatory	4111="TCP". Note that no other protocol type is supported by this endpoint.
PortNumber		Mandatory	The number of the TCP Port that this element represents.

---



---

**STABLE**



---



---

## EXPERIMENTAL

### 15 Host Filesystem Profile

#### 15.1 Description

##### 15.1.1 Synopsis

**Profile Name:** Host Filesystem (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** FileSystemConfigurationService

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 240 describes the related profiles for Host Filesystem.

**Table 240 - Related Profiles for Host Filesystem**

Profile Name	Organization	Version	Requirement	Description
Filesystem	SNIA	1.7.0	Mandatory	
File Export	SNIA	1.7.0	Optional	
Access Points	SNIA	1.3.0	Optional	
Software	SNIA	1.4.0	Optional	
Filesystem Manipulation	SNIA	1.7.0	Optional	
File Export Manipulation	SNIA	1.7.0	Optional	
Filesystem Performance	SNIA	1.7.0	Optional	Experimental.
FileSystem Quotas	SNIA	1.7.0	Optional	Experimental.
Filesystem Replication Services	SNIA	1.7.0	Optional	Experimental.
Job Control	SNIA	1.5.0	Optional	
Device Credentials	SNIA	1.3.0	Optional	
Health	SNIA	1.2.0	Mandatory	
Launch In Context	DMTF	1.0.0	Optional	Experimental. See DSP1102, version 1.0.0
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

##### 15.1.2 Overview

The Host Filesystem Profile is a component profile of the Base Server (host system) Profile (see 30 Base Server Profile in the *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*). All references to ComputerSystem in the Host Filesystem Profile implies a single instance for a customer server or storage system as defined in the Base Server Profile. See Annex B (Informative) Host Profile Deployment Guidelines in the *Storage Management Technical Specification, Part 7 Host Elements, 1.7.0 Rev 5* for information on the use of host profiles with Base Server profile.

A host filesystem is a filesystem that runs on an application host and gets its storage from a volume manager or host operating systems (Host Discovered Resources) storage. The storage obtained is visible

to external management tools and they may share their storage with other host applications. For example, host filesystem might go to a volume manager for its storage. The volume manager provides storage to the host filesystem, but may also supply storage to other host applications (e.g., a DataBase manager) or to other host filesystems.

This profile defines how to model the host filesystem constructs, and how it reflects connections to and storage from the volume manager or system below it.

The Host Filesystem Profile reuses many profiles and packages used by the NAS profiles. This is illustrated in Figure 27: "Host Filesystem Profiles and Package".

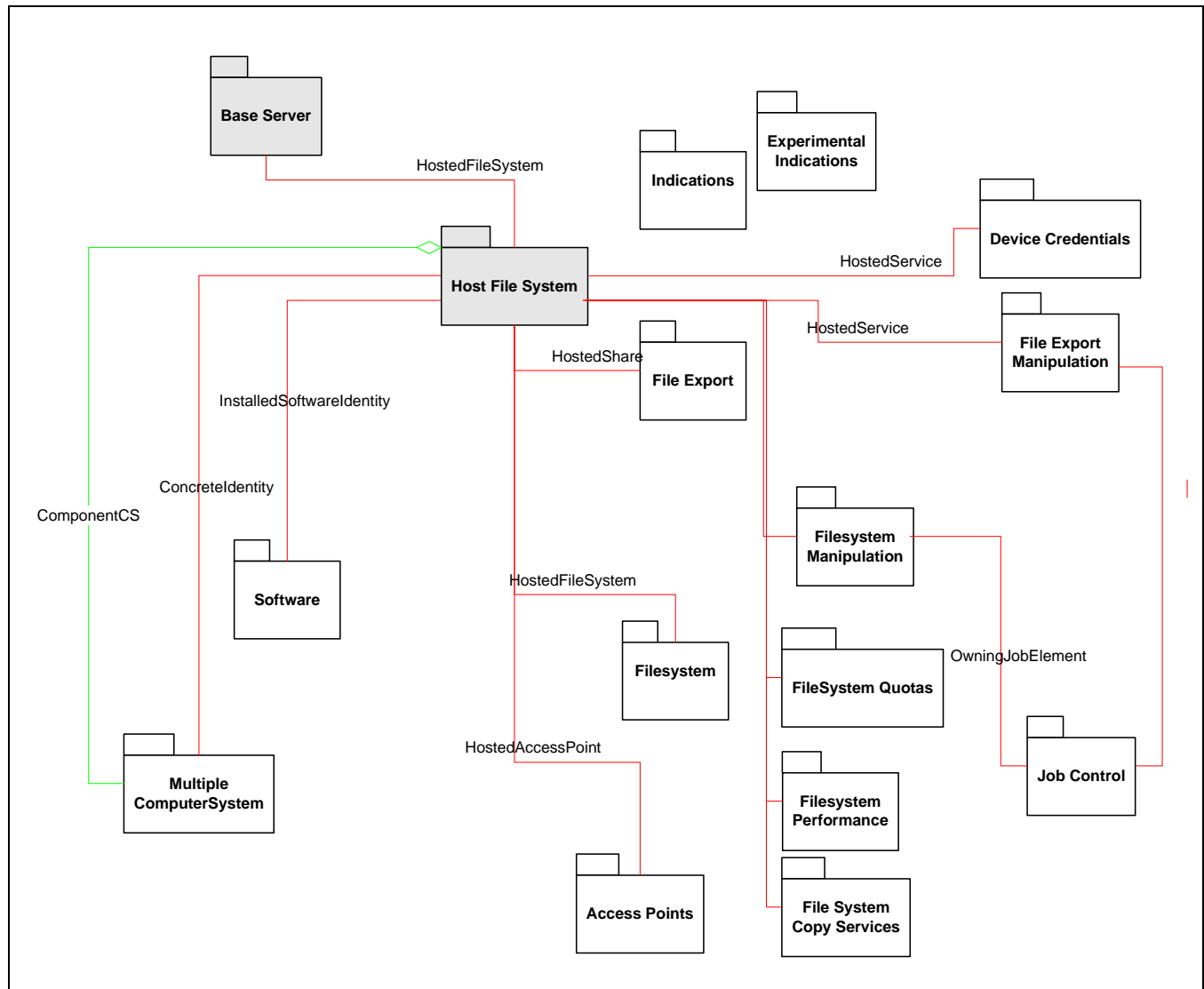
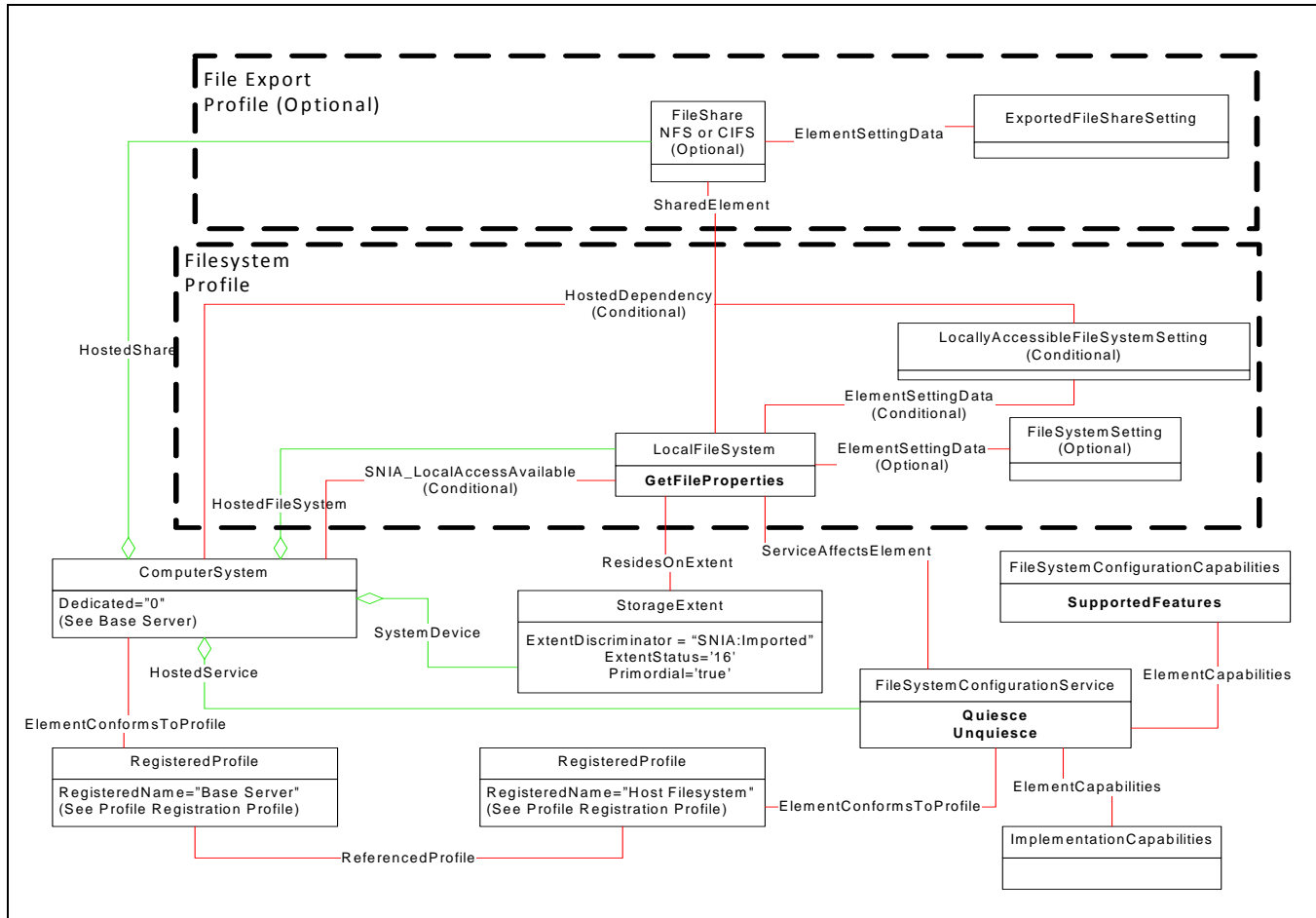


Figure 27 - Host Filesystem Profiles and Package

## 15.2 Implementation

### 15.2.1 Summary Instance Diagram

Figure 28: "Host Filesystem Instance Diagram" illustrates the mandatory classes for the host filesystem (and the Base Server). This figure shows all the classes that are mandatory and some of the optional classes (identified) for the Host Filesystem Profile.



**Figure 28 - Host Filesystem Instance Diagram**

The Host Filesystem Profile draws its storage from LogicalDisks provided by a volume manager or HDR profile. The profile models the LogicalDisks that it gets from the underlying volume manager or HDR as StorageExtents. The association between a LocalFileSystem and the StorageExtents it resides on is ResidesOnExtent.

The Base Server ComputerSystem may not be a real ComputerSystem. It is merely the ManagedElement upon which all aspects of the host filesystem offering are scoped.

LocalFileSystems are created on the StorageExtents and files within those LocalFileSystems may be shared (FileShare) with remote users. The Filesystem Profile is a required profile.

The host filesystem augments the definition of the LocalFileSystem defined in the Filesystem Profile by adding a method (GetFileProperties). If this method is supported, the support shall be indicated in the FileSystemCapabilities.SupportedFeatures property.

The host filesystem also includes a `FileSystemConfigurationService` and a `FileSystemConfigurationCapabilities`. These are the augmented instances of those defined by the Filesystem Manipulation Profile. The Host Filesystem extends these with methods (`Quiesce` and `Unquiesce`) and a property (`SupportedFeatures`). The Filesystem Manipulation Profile is optional, but the `FileSystemConfigurationService` and the `FileSystemConfigurationCapabilities` are required by the Host Filesystem Profile. If the Filesystem Manipulation Profile is not implemented, the Host Filesystem Profile shall implement these two classes as defined by this profile.

In addition to the `FileSystemConfigurationCapabilities`, an instance of `ImplementationCapabilities` may be associated to the `FileSystemConfigurationService`. This `Capabilities` instance identifies the capacity optimization techniques supported by the implementation. An implementation may advertise that it supports "None", "SNIA:Thin Provisioning", "SNIA:Data Compression" or "SNIA:Data Deduplication".

The classes and associations in the dashed boxes are from the component profiles (as indicated by the labels on the dashed boxes).

The `SharedElement` association between the `FileShare` and the `LocalFilesystem` is required if `FileShares` are implemented (the File Export Profile).

Also note that `FileSystemSetting` (and the corresponding `ElementSettingData`) are also optional. They are only shown here to illustrate where they would show up in the model should they be implemented.

In the base Host Filesystem Profile, the model is automatically populated based on how the host filesystem is configured. Client modification of the configuration (including configuring storage, creating extents, local filesystems and file shares) are functions found in component profiles of the Host Filesystem Profile.

### **15.2.2 Host Filesystem Use of Filesystem Profile (Mandatory)**

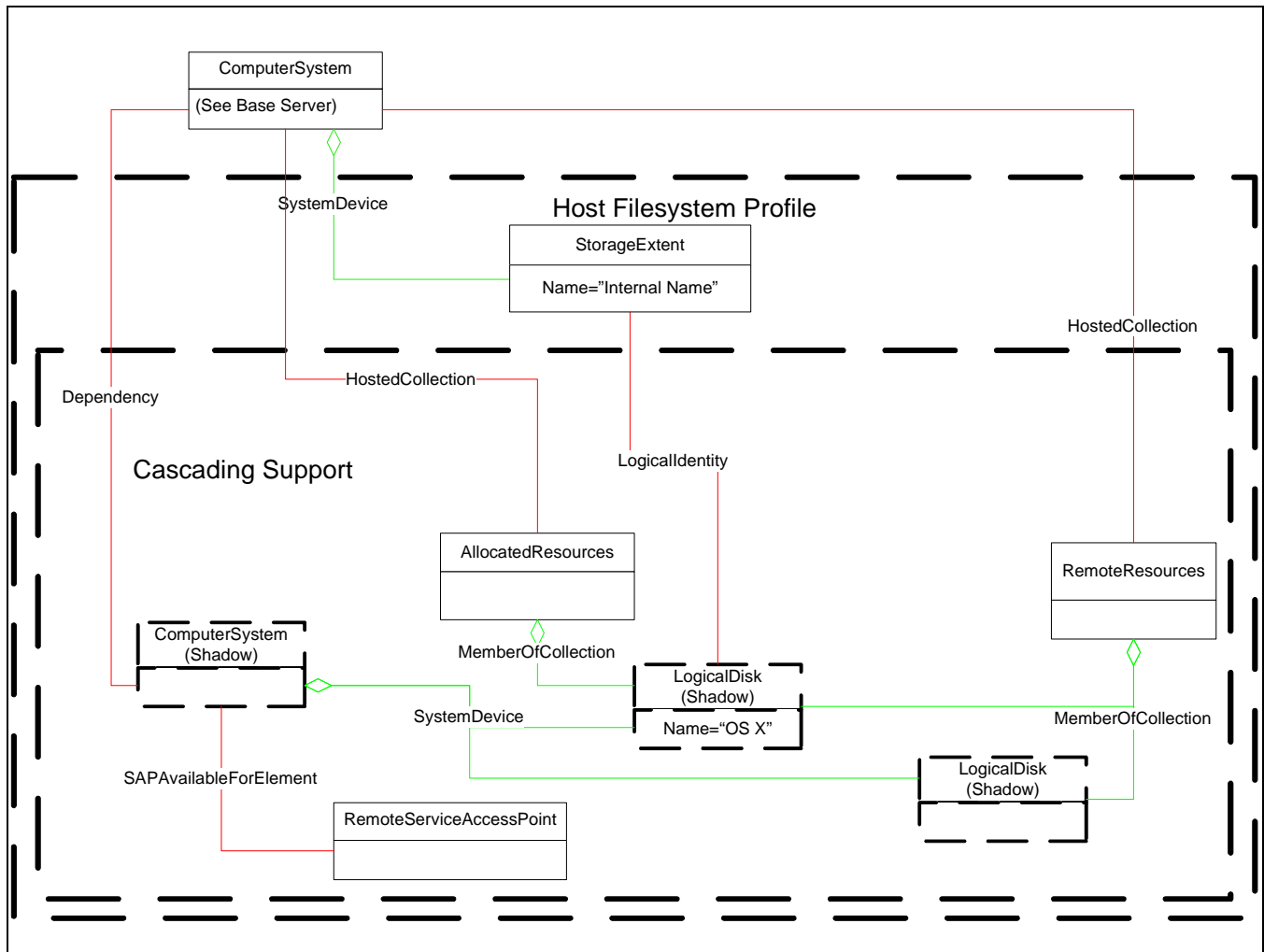
The Host Filesystem Profile uses the Filesystem Profile for modeling of its filesystem constructs. For the host filesystem, implementation of the Filesystem Profile is mandatory. See 8 Filesystem Profile for details on this modeling.

### **15.2.3 Host Filesystem Use of File Export Profile (Optional)**

The Host Filesystem Profile uses the File Export Profile for modeling of its file export constructs. For the host filesystem, implementation of the File Export Profile is optional. See 4 File Export Profile for details on this modeling.

#### 15.2.4 Host Filesystem Support of Cascading

Figure 29: "Host Filesystem support for Cascading" illustrates the host filesystem support for cascading. Support for some of the elements for cascading are mandatory. The figure illustrates stitching between the Host Filesystem and the Volume Management or HDR Profiles.



### Figure 29 - Host Filesystem support for Cascading

A host filesystem gets its storage from the operating system (HDR Profile) or a volume manager. As such, there is a cascading relationship between the Host Filesystem Profile and the profiles (e.g., Volume Management Profiles) that provide the storage for the host filesystem. Figure 29, “Host Filesystem support for Cascading” illustrates the constructs to be used to model this cascading relationship.

- The Host Filesystem cascaded resources are StorageExtents (used to house filesystem data)
- The Host Filesystem obtains the storage for these from LogicalDisks in Volume Management or HDR Profiles.
- Each StorageExtent used by the Host Filesystem maps (via LogicalIdentity) to a LogicalDisk (from the Volume Management or HDR Profile).

The embedded dashed box in the figure illustrates the classes and associations of the cascading support. The dashed classes are shadow instances (copies cached from the Volume Management or HDR Profile). The other classes of the cascading support represent Host Filesystem usage of those classes. For

example, the collection `AllocatedResources` collects all the volume manager or HDR volumes that are used by the Host Filesystem. The `RemoteResources` collection collects all `LogicalDisks` that the Host Filesystem has discovered (whether used or not).

The Dependency between the Base Server `ComputerSystem` and the shadow `ComputerSystem` may exist, even when there are no resources that are imported. This signifies that the Host Filesystem has discovered the Volume Management or HDR Profile, but has no access to any of their `LogicalDisks`.

NOTE The Base Server and Shadow `ComputerSystems` may represent the same system.

The `RemoteServiceAccessPoint` is the URL of the management interface that the Host Filesystem uses for managing the volume manager or HDR support. This may or may not be an SMI-S Server URL.

### 15.2.5 Health and Fault Management Consideration

The Host Filesystem Profile supports state information (e.g., `OperationalStatus` and `HealthState`) on the following elements of the model:

- `ComputerSystems` (See 22 Health Package in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*)
- `FileShares` that are exported (See 4 File Export Profile)
- `LocalFileSystems` (See 8 Filesystem Profile)

## 15.3 Methods of the Profile

### 15.3.1 Extrinsic Methods of the Profile

Not defined in this standard

### 15.3.2 Extrinsic Methods in the Filesystem Profile

#### 15.3.2.1 GetFileProperties

uint32 GetFileProperties(

```
    IN string DirectoryName,
    IN, OUT string Handle,
    IN, OUT uint64 NumberOfFiles,
    IN (false), OUT EmbeddedInstance("CIM_LogicalFile") string FileRecs[];
);
```

This method gets a set of file records from a filesystem. As there may be millions of records in this report, a chunking mechanism is provided so that the client does not become overwhelmed by the quantity of data furnished by the server.

The `DirectoryName` is an optional pathname for a directory to restrict the data returned. If this parameter is NULL, then files are returned for all files in the filesystem.

The initial call to `GetFileProperties` shall pass in NULL as a `Handle`. Subsequent calls shall pass back the `Handle` exactly as received from the server, without modification, as an indication of where to continue the report from.

The `NumberOfFiles` is the number of files returned in a block of `FileRecs`. If NULL the provider will supply a default number (and put that number in the parameter as output).



### 15.3.3 Extrinsic Methods in the Filesystem Manipulation Profile

#### 15.3.3.1 QuiesceFilesystem

```
uint32 QuiesceFilesystem(
    IN, Required CIM_Filesystem REF TheElement,
    IN, OUT datetime Timeout,
    IN (false), OUT CIM_Job REF Job;
);
```

This method temporarily suspends write operations to the underlying storage extents of a filesystem specified by TheElement.

The Timeout parameter identifies how long the system is to hold the filesystem in a quiesced state. The default is 30 seconds. The purpose of the timeout is to prevent a filesystem from staying in a quiesced state due to an application failure. That is, if the application does not do an unquiesce in the timeout period, the provider may automatically do the unquiesce.

#### 15.3.3.2 Unquiesce a Filesystem

```
uint32 UnquiesceFilesystem(
    IN, Required CIM_Filesystem REF TheElement;
);
```

This method resumes write activity to the underlying storage extents of a filesystem specified by TheElement.

### 15.3.4 Intrinsic Methods of the Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

## 15.4 Use Cases

### 15.4.1 Discovery of the Filesystem Volumes

Table 241 identifies the elements of the use case to discover the volumes on which a filesystem resides.

**Table 241 - Discovery of the Filesystem Volumes**

Use Case Element	Description
Summary	Given a Host Filesystem Profile, find the volumes (by their external name) on which a filesystem resides

**Table 241 - Discovery of the Filesystem Volumes**

Use Case Element	Description
Basic Course of Events	1. Find the filesystem (using its name) 2. Find the related Volumes (that the filesystem is on) 3. Locate the external names of the volumes
Alternative Paths	None
Exception Paths	None
Triggers	Discovery or rebuild of the filesystem configuration
Assumptions	None
Preconditions	The Base Server system of the profile has been discovered from profile registration and ElementConformsToProfile.
Postconditions	A list of volumes on which the filesystems exist.

#### 15.4.2 Expansion of a Filesystem

Table 242 identifies the elements of the use case to increase the size of a filesystem.

**Table 242 - Expansion of a Filesystem**

Use Case Element	Description
Summary	Increase the size of the filesystem by a certain amount.
Basic Course of Events	1. Administrator identifies the filesystem and size to increase. 2. System responds operation is complete.
Alternative Paths	None
Exception Paths	Failure 2a. System responds that the filesystem cannot be extended. The filesystem is left unchanged. Invalid state: Filesystem state does not allow expansion. 2b. System cannot support the size increase requested (size too large) 2c. System cannot support expansion of a mounted filesystem 2d. System cannot support expansion given the current configuration of partitions 2e. Filesystem is in a transient state that does not allow expansion
Triggers	Business need to increase the size of the filesystem.
Assumptions	None
Preconditions	Administrator has permission and access for the operation.
Postconditions	The filesystem size is at least the original size plus the requested increment.

#### 15.4.3 Replication of a Filesystem

Table 243 identifies the elements of the use case to create a point in time copy of a filesystem.

**Table 243 - Replication of a Filesystem**

Use Case Element	Description
Summary	Given a filesystem, create a point in time copy of the filesystem.
Basic Course of Events	1. Administrator identifies filesystem to copy. 2. Administrator signals that the copy should be created. 3. System responds that the copy is ready.

**Table 243 - Replication of a Filesystem**

Use Case Element	Description
Alternative Paths	Specify Storage 1a. Administrator identifies where (e.g., what storage extent) and maximum space that may be used for the copy.
Exception Paths	Failed 3. System responds that the copy could not be created.
Triggers	Business need.
Assumptions	There is no requirement that the copy increases the fault tolerance of the filesystem.
Preconditions	Filesystem available.
Postconditions	The copy is available. The copy is self-consistent.

**15.4.4 Quiesce a Filesystem**

Table 244 identifies the elements of the use case to quiesce a filesystem.

**Table 244 - Quiesce a Filesystem**

Use Case Element	Description
Summary	Temporarily suspends write operations to the underlying storage extents of a filesystem.
Basic Course of Events	1. Administrator identifies the filesystem. 2. System responds operation is complete.
Alternative Paths	Provide timeout 1a. Administrator provides a maximum quiesce timeout.
Exception Paths	Failure 2a. System responds that the write operations to the filesystem cannot be suspended. The filesystem has the same operational state as before. Filesystem already quiesced. 2b. System responds that the filesystem is already suspended. The filesystem has the same operational state as before (and the request is ignored - timeout not extended)
Triggers	Business need for the image of the filesystem on the underlying storage extents to be complete and correct as of a known point in time.
Assumptions	Any application that needs quiescing has been completed. (NOTE: because provider cannot tell)
Preconditions	Administrator has permission and access for the operation.
Postconditions	The data residing on the underlying storage extents reflects the state of the filesystem at some point in time between steps 1 and 2. No write activity to the filesystem shall transfer to the underlying storage extents. All future write activity should be blocked.

**15.4.5 Unquiesce a Filesystem**

Table 245 identifies the elements of the use case to unquiesce a filesystem.

**Table 245 - Unquiesce a Filesystem**

Use Case Element	Description
Summary	Resume write activity to the underlying storage extents of a filesystem.

**Table 245 - Unquiesce a Filesystem**

Use Case Element	Description
Basic Course of Events	1. Administrator identifies the filesystem 2. System responds operation is complete
Alternative Paths	None
Exception Paths	None
Triggers	Business need for the quiesce operation has completed.
Assumptions	None
Preconditions	Administrator has permission and access for the operation.
Postconditions	The selected filesystem is no longer quiesced.

#### 15.4.6 Filesystem quiesce timeout

Table 246 identifies the elements of the use case when a quiesced filesystem times out.

**Table 246 - Filesystem quiesce timeout**

Use Case Element	Description
Summary	Resume write activity to the underlying storage extents of a filesystem after a timeout
Basic Course of Events	1. System responds operation is complete.
Alternative Paths	None
Exception Paths	None
Triggers	The timeout has expired.
Assumptions	None
Preconditions	The filesystem is in a quiesced state.
Postconditions	The selected filesystem is no longer quiesced.

#### 15.4.7 Retrieve File Information

Table 247 identifies the elements of the use case retrieving file information from a filesystem.

**Table 247 - Retrieve File Information**

Use Case Element	Description
Summary	Get available information on files in a filesystem directory.
Basic Course of Events	1. Administrator identifies the filesystem and directory. 2. System responds
Alternative Paths	None
Exception Paths	Failure 2a. System responds that the directory identified was not found. 2b. System responds that the operation is not supported at this time
Triggers	Business need for the information to support filesystem "information lifecycle management" functions.

**Table 247 - Retrieve File Information**

Use Case Element	Description
Assumptions	The underlying filesystem supports the information being requested
Preconditions	Administrator has permission and access for the operation and the operation is supported.
Postconditions	Once all data has been returned, a new operation must initiated to get information about the files.

## 15.5 CIM Elements

Table 248 describes the CIM elements for Host Filesystem.

**Table 248 - CIM Elements for Host Filesystem**

Element Name	Requirement	Description
15.5.1 CIM_AllocatedResources	Mandatory	This is a SystemSpecificCollection for collecting LogicalDisks that are being used by the Host Filesystem profile (e.g., LogicalDisks that the filesystem is using).
15.5.2 CIM_ComputerSystem (Shadow)	Mandatory	'Top level' system that represents a Volume Manager or Host Discovered Resources.
15.5.3 CIM_Dependency (Systems)	Mandatory	This associates the Volume Manager or Host Discovered Resources System to the Host Filesystem System.
15.5.4 CIM_ElementCapabilities (FS Configuration Capabilities)	Mandatory	Associates the Filesystem Configuration Service to the Capabilities element that represents the capabilities that it supports.
15.5.5 CIM_ElementCapabilities (ImplementationCapabilities to Service)	Optional	Experimental. Associates the Host Filesystem configuration service to the CIM_ImplementationCapabilities supported by the implementation.
15.5.6 CIM_ElementConformsToProfile (FilesystemConfigurationService to Host Filesystem RegisteredProfile)	Mandatory	Ties the FileSystemConfigurationService to the registered profile for Host Filesystem.
15.5.7 CIM_FileSystemConfigurationCapabilities	Mandatory	An extension of the FileSystemConfigurationCapabilities defined in the Filesystem Manipulation Profile.
15.5.8 CIM_FileSystemConfigurationService	Mandatory	An extension of the Filesystem Configuration Service that adds filesystem methods.
16.5.8 CIM_HostedCollection (Allocated Resources)	Mandatory	This would associate the AllocatedResources collection to the Base Server system for the Host Filesystem.
15.5.9 CIM_HostedCollection (Remote Resources)	Conditional	Conditional requirement: This is required if CIM_RemoteResources is modeled. This would associate the RemoteResources collection to the Base Server system for the Host Filesystem.
15.5.10 CIM_HostedService	Mandatory	Associates the Filesystem Configuration Service to the Base Server ComputerSystem.
15.5.11 CIM_ImplementationCapabilities (ImplementationCapabilities)	Optional	Experimental. The capabilities of the profile implementation.

**Table 248 - CIM Elements for Host Filesystem**

Element Name	Requirement	Description
15.5.13 CIM_LogicalDisk (Shadow)	Mandatory	A shadow instance of a LogicalDisk that is imported to the Host Filesystem Profile.
15.5.12 CIM_LocalFileSystem	Mandatory	Represents an extension of the LocalFileSystem defined in the Filesystem Profile.
15.5.14 CIM_LogicalFile	Optional	This is an output of the GetFileProperties method on CIM_LocalFileSystem. It is never instantiated, but the output follows this format.
15.5.15 CIM_LogicalIdentity (LogicalDisk)	Mandatory	Associates a Host Filesystem StorageExtent to a shadow instance of an (imported) LogicalDisk.
15.5.16 CIM_MemberOfCollection (Allocated Resources)	Mandatory	This supports collecting LogicalDisks. This is required to support the AllocatedResources collection.
15.5.17 CIM_MemberOfCollection (Remote Resources)	Optional	This supports collecting all Shadow instances of LogicalDisk that the Host Filesystem has available to use. This is optional when used to support the RemoteResources collection (the RemoteResources collection is optional).
15.5.19 CIM_RemoteResources	Optional	This is a SystemSpecificCollection for collecting Logical Disks that may be allocated by the Host Filesystem Profile (e.g., LogicalDisks that may be allocated to support a filesystem).
15.5.18 CIM_RemoteServiceAccessPoint (Shadow)	Optional	CIM_RemoteServiceAccessPoint represents the management interface to a Shadow system.
15.5.20 CIM_ResidesOnExtent	Mandatory	Represents the association between a local FileSystem and the underlying storage extent(s) that it is built on.
15.5.21 CIM_SAPAvailableForElement	Conditional	Conditional requirement: This is required if CIM_RemoteServiceAccessPoint is modeled. Represents the association between a RemoteServiceAccessPoint and the Shadow (Volume Manager or Host Discovered Resources) System to which it provides access.
15.5.22 CIM_ServiceAffectsElement	Mandatory	Associates the Filesystem Configuration Service to the filesystems that the service manages.
15.5.23 CIM_StorageExtent (Primordial Imported Extent)	Mandatory	Used to represent the storage imported from the OS (Host Discovered Resources) or Volume Managers.
15.5.24 CIM_SystemDevice (LogicalDisks)	Mandatory	This association links shadow <b>LogicalDisks</b> to the scoping (Shadow) system (of the Volume Manager or Host Discovered Resources). This is used to associate the shadow LogicalDisks with the System that manages them.

**Table 248 - CIM Elements for Host Filesystem**

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::Operational Status <> PreviousInstance.CIM_ComputerSystem::OperationalStatus	Mandatory	CQL -Change of Status of a ComputerSystem. PreviousInstance is optional, but may be supplied by an implementation of the Profile.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageExtent AND SourceInstance.CIM_StorageExtent::OperationalStatus <> PreviousInstance.CIM_StorageExtent::OperationalStatus	Mandatory	CQL -Change of status of a StorageExtent. PreviousInstance is optional, but may be supplied by an implementation of the Profile.

**15.5.1 CIM\_AllocatedResources**

An instance of a default CIM\_AllocatedResources defines the set of LogicalDisks that are allocated and in use by the Host Filesystem Profile.

CIM\_AllocatedResources is subclassed from CIM\_SystemSpecificCollection.

At least one instance of the CIM\_AllocatedResources shall exist for a Host Filesystem Profile and shall be hosted by one of its ComputerSystems (typically the top level ComputerSystem).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 249 describes class CIM\_AllocatedResources.

**Table 249 - SMI Referenced Properties/Methods for CIM\_AllocatedResources**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the AllocatedResources collection (e.g., Allocated LogicalDisks).
ElementType		Mandatory	The type of remote resources collected by the AllocatedResources collection.  For this version of SMI-S, the only value supported is '7' (LogicalDisk).
CollectionDiscriminator		Mandatory	An array of strings indicating the purposes of the collection of elements. This shall contain 'SNIA:Imported Volumes'.

**15.5.2 CIM\_ComputerSystem (Shadow)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 250 describes class CIM\_ComputerSystem (Shadow).

**Table 250 - SMI Referenced Properties/Methods for CIM\_ComputerSystem (Shadow)**

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the shadow system. E.g., IP address.
ElementName		Mandatory	User friendly name.
OtherIdentifyingInfo	C	Mandatory	
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	Overall status of the shadow system, as seen by the Host Filesystem.
NameFormat		Mandatory	Format for Name property.
Dedicated		Mandatory	Indicates that this computer system is dedicated to operation as a shadow system.
PrimaryOwnerContact	M	Optional	Contact a details for owner.
PrimaryOwnerName	M	Optional	Owner of the shadow system.

### 15.5.3 CIM\_Dependency (Systems)

CIM\_Dependency is an association between a shadow System (Volume Manager or Host Discovered Resources) and the Host Filesystem System (ComputerSystem). The specific nature of the dependency is determined by associations between resources (StorageExtents) of the Host Filesystem system and resources (LogicalDisks) of the shadow system.

CIM\_Dependency is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 251 describes class CIM\_Dependency (Systems).

**Table 251 - SMI Referenced Properties/Methods for CIM\_Dependency (Systems)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Base Server System.
Dependent		Mandatory	The shadow System (system of the Volume Manager or Host Discovered Resources).

### 15.5.4 CIM\_ElementCapabilities (FS Configuration Capabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory



Table 252 describes class CIM\_ElementCapabilities (FS Configuration Capabilities).

**Table 252 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (FS Configuration Capabilities)**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The Filesystem Configuration Capabilities element.
ManagedElement		Mandatory	The Filesystem Configuration Service.

#### 15.5.5 CIM\_ElementCapabilities (ImplementationCapabilities to Service)

Experimental. Associates the Host Filesystem configuration service to the CIM\_ImplementationCapabilities supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 253 describes class CIM\_ElementCapabilities (ImplementationCapabilities to Service).

**Table 253 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (ImplementationCapabilities to Service)**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The ImplementationCapabilities.
ManagedElement		Mandatory	The host FileSystemConfigurationService that has ImplementationCapabilities.

#### 15.5.6 CIM\_ElementConformsToProfile (FileSystemConfigurationService to Host Filesystem RegisteredProfile)

The CIM\_ElementConformsToProfile ties FileSystemConfigurationService to the registered profile for Host Filesystem.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 254 describes class CIM\_ElementConformsToProfile (FileSystemConfigurationService to Host Filesystem RegisteredProfile).

**Table 254 - SMI Referenced Properties/Methods for CIM\_ElementConformsToProfile (FileSystemConfigurationService to Host Filesystem RegisteredProfile)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A FileSystemConfigurationService instance that represents the Host Filesystem.
ConformantStandard		Mandatory	RegisteredProfile instance describing the Host Filesystem profile.

### 15.5.7 CIM\_FileSystemConfigurationCapabilities

An extension of the FileSystemConfigurationCapabilities defined in the Filesystem Manipulation Profile. For the base definition of this class, see 9.5.9 CIM\_FileSystemConfigurationCapabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 255 describes class CIM\_FileSystemConfigurationCapabilities.

**Table 255 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
ElementName		Mandatory	See the ElementName property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
SupportedActualFileSystemTypes		Mandatory	See the SupportedActualFileSystemTypes property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
SupportedSynchronousMethods	N	Mandatory	See the SupportedSynchronousMethods property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
SupportedAsynchronousMethods	N	Mandatory	See the SupportedAsynchronousMethods property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
InitialAvailability		Mandatory	See the InitialAvailability property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
LocalAccessibilitySupport		Optional	See the LocalAccessibilitySupport property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
BlockStorageCreationSupport		Optional	See the BlockStorageCreationSupport property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
DirectoryServerParameterSupported		Optional	See the DirectoryServerParameterSupported property in 9.5.9 CIM_FileSystemConfigurationCapabilities.
SupportedFeatures		Mandatory	This may be 'None', 'GetFileProperties' or 'Quiesce/Unquiesce'.

### 15.5.8 CIM\_FileSystemConfigurationService

An extension of the Filesystem Configuration Service that adds filesystem methods. For the base definition of the FileSystemConfigurationService see 9.5.10 CIM\_FileSystemConfigurationService.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 256 describes class CIM\_FileSystemConfigurationService.

**Table 256 - SMI Referenced Properties/Methods for CIM\_FileSystemConfigurationService**

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	See the ElementName property in 9.5.10 CIM_FileSystemConfigurationService.
SystemCreationClassName		Mandatory	See the SystemCreationClassName property in 9.5.10 CIM_FileSystemConfigurationService.
SystemName		Mandatory	See the SystemName property in 9.5.10 CIM_FileSystemConfigurationService.
CreationClassName		Mandatory	See the CreationClassName property in 9.5.10 CIM_FileSystemConfigurationService.
Name		Mandatory	See the Name property in 9.5.10 CIM_FileSystemConfigurationService.
Quiesce()		Conditional	Conditional requirement: This is required if SupportedFeatures includes \Quiesce/Unquiesce\.' See the method description in 15.3.3.1 QuiesceFilesystem.
Unquiesce()		Conditional	Conditional requirement: This is required if SupportedFeatures includes \Quiesce/Unquiesce\.' See the method description in 15.3.3.2 Unquiesce a Filesystem.

#### 15.5.9 CIM\_HostedCollection (Remote Resources)

CIM\_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Host Filesystem Profile, it is used to associate the Remote Resources to the Base Server Computer System.

CIM\_HostedCollection is subclassed from CIM\_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if CIM\_RemoteResources is modeled.

Table 257 describes class CIM\_HostedCollection (Remote Resources).

**Table 257 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Remote Resources)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

#### 15.5.10 CIM\_HostedService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 258 describes class CIM\_HostedService.

**Table 258 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The Filesystem Configuration Service.
Antecedent		Mandatory	The Base Server ComputerSystem.

#### 15.5.11 CIM\_ImplementationCapabilities (ImplementationCapabilities)

Experimental. The capabilities (features) of the profile implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 259 describes class CIM\_ImplementationCapabilities (ImplementationCapabilities).

**Table 259 - SMI Referenced Properties/Methods for CIM\_ImplementationCapabilities (ImplementationCapabilities)**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the implementation capability of an implementation.
ElementName		Optional	A provider supplied user-friendly name for this CIM_ImplementationCapabilities element.
SupportedCapacityOptimizations		Mandatory	This array of strings lists the capacity optimization techniques that are supported by the implementation. Valid string values are "none"   "SNIA:Thin Provisioning"   "SNIA:Data Compression"   "SNIA:Data Deduplication".

#### 15.5.12 CIM\_LocalFileSystem

Represents an extension of the LocalFileSystem defined in the Filesystem Profile. See 8.5.12 CIM\_LocalFileSystem.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 260 describes class CIM\_LocalFileSystem.

**Table 260 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
CSCreationClassName		Mandatory	See the CSCreationClassName property in 8.5.12 CIM_LocalFileSystem.
CSName		Mandatory	See the CSName property in 8.5.12 CIM_LocalFileSystem.

**Table 260 - SMI Referenced Properties/Methods for CIM\_LocalFileSystem**

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	See the CreationClassName property in 8.5.12 CIM_LocalFileSystem.
Name		Mandatory	See the Name property in 8.5.12 CIM_LocalFileSystem.
OperationalStatus		Mandatory	See the OperationalStatus property in 8.5.12 CIM_LocalFileSystem.
Root		Optional	See the Root property in 8.5.12 CIM_LocalFileSystem.
BlockSize		Optional	See the BlockSize property in 8.5.12 CIM_LocalFileSystem.
FileSystemSize		Optional	See the FileSystemSize property in 8.5.12 CIM_LocalFileSystem.
AvailableSpace		Optional	See the AvailableSpace property in 8.5.12 CIM_LocalFileSystem.
ReadOnly		Optional	See the ReadOnly property in 8.5.12 CIM_LocalFileSystem.
EncryptionMethod		Optional	See the EncryptionMethod property in 8.5.12 CIM_LocalFileSystem.
CompressionMethod		Optional	See the CompressionMethod property in 8.5.12 CIM_LocalFileSystem.
CaseSensitive		Mandatory	See the CaseSensitive property in 8.5.12 CIM_LocalFileSystem.
CasePreserved		Mandatory	See the CasePreserved property in 8.5.12 CIM_LocalFileSystem.
CodeSet		Optional	See the CodeSet property in 8.5.12 CIM_LocalFileSystem.
MaxFileNameLength		Mandatory	See the MaxFileNameLength property in 8.5.12 CIM_LocalFileSystem.
FileSystemType		Mandatory	See the FileSystemType property in 8.5.12 CIM_LocalFileSystem.
NumberOfFiles		Optional	See the NumberOfFiles property in 8.5.12 CIM_LocalFileSystem.
GetFileProperties()		Conditional	Conditional requirement: This is required if SupportedFeatures includes \GetFileProperties\'. See the method description in 15.3.2.1 GetFileProperties.

**15.5.13 CIM\_LogicalDisk (Shadow)**

A shadow instance of a remote LogicalDisk that is imported to the Host Filesystem profile. If the Host Filesystem has access to the Volume Management or Host Discovered Resources profile, the data in this class should reflect what the Host Filesystem obtains from that profile. If the Host Filesystem does not have access to the Volume Management or Host Discovered Resources profile, then this should be filled out as best can be done.

The properties in this class table are the properties as defined by either Volume Management, Block Services or Host Discovered Resources. If a property is optional in any of the three profiles, then it is defined as optional in the Shadow LogicalDisk. The only exception to this rule is the ExtentDiscriminator, which is used by the Host Filesystem profile to distinguish the LogicalDisk from other StorageExtents.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 261 describes class CIM\_LogicalDisk (Shadow).

**Table 261 - SMI Referenced Properties/Methods for CIM\_LogicalDisk (Shadow)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.
NameFormat		Mandatory	Format for name.
ExtentStatus		Optional	
OperationalStatus		Mandatory	Value shall be 0 2 3 6 8 15 (Unknown or OK or Degraded or Error or Starting or Dormant).
BlockSize		Optional	
NumberOfBlocks		Optional	The number of blocks of capacity consumed from the parent StoragePool.
ConsumableBlocks		Optional	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Optional	
NoSinglePointOfFailure		Optional	
DataRedundancy		Optional	
PackageRedundancy		Optional	
DeltaReservation		Optional	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Shadow'.

#### 15.5.14 CIM\_LogicalFile

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 262 describes class CIM\_LogicalFile.

**Table 262 - SMI Referenced Properties/Methods for CIM\_LogicalFile**

Properties	Flags	Requirement	Description & Notes
CSCreationClassName		Mandatory	Class Name of the ComputerSystem that hosts the filesystem containing this file.
CSName		Mandatory	The Name property of the ComputerSystem that hosts the filesystem containing this file.
FSCreationClassName		Mandatory	Class Name of the LocalFileSystem that represents the filesystem containing this file.
FSName		Mandatory	The Name property of the LocalFileSystem that represents the filesystem containing this file.
CreationClassName		Mandatory	Class Name of this instance of LogicalFile that represents the file.
Name		Mandatory	The Name property of the LogicalFile that represents the file.
ElementName		Mandatory	The pathname from the root of the containing LocalFileSystem to this LogicalFile. The root of the LocalFileSystem is indicated if this is NULL or the empty string. The format of the pathname is specific to the LocalFileSystem's FileSystemType. If it is a sequence of directories from the root, the separator string is specified by the CIM_LocalFileSystem.PathNameSeparatorString property.
FileSize		Optional	Size of the File in bytes.
CreationDate		Optional	File's creation date.
LastModified		Optional	Time that the File was last modified.
LastAccessed		Optional	Time that the File was last accessed.
Readable		Optional	Boolean indicating that the File can be read.
Writeable		Optional	Boolean indicating that the File can be written.
Executable		Optional	Indicates the file is executable.
CompressionMethod		Optional	A free form string indicating the algorithm or tool used to compress the LogicalFile.
EncryptionMethod		Optional	A free form string indicating the algorithm or tool used to encrypt the LogicalFile.
InUseCount		Optional	The number of 'file opens' that are currently active against the File.

#### 15.5.15CIM\_LogicalIdentity (LogicalDisk)

Associates local StorageExtent to a shadow instance of an (imported) LogicalDisk.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 263 describes class CIM\_LogicalIdentity (LogicalDisk).

**Table 263 - SMI Referenced Properties/Methods for CIM\_LogicalIdentity (LogicalDisk)**

Properties	Flags	Requirement	Description & Notes
SystemElement		Mandatory	This is a reference to the shadow (imported) LogicalDisk.
SameElement		Mandatory	This is a reference to the Host Filesystem StorageExtent that maps to the shadow (imported) LogicalDisk.

#### 15.5.16 CIM\_MemberOfCollection (Allocated Resources)

This use of MemberOfCollection is to collect all allocated shadow LogicalDisk instances (in the AllocatedResources collection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 264 describes class CIM\_MemberOfCollection (Allocated Resources).

**Table 264 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Allocated Resources)**

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

#### 15.5.17 CIM\_MemberOfCollection (Remote Resources)

This use of MemberOfCollection is to collect all shadow LogicalDisk instances (in the RemoteResources collection). Each association (and the RemoteResources collection, itself) is created through external means.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 265 describes class CIM\_MemberOfCollection (Remote Resources).

**Table 265 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Remote Resources)**

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

#### 15.5.18 CIM\_RemoteServiceAccessPoint (Shadow)

CIM\_RemoteServiceAccessPoint is an instance that provides access information for accessing the actual Shadow (Volume Manager or Host Discovered Resources) via a management interface.



CIM\_RemoteServiceAccessPoint is not subclassed from CIM\_ServiceAccessPoint.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 266 describes class CIM\_RemoteServiceAccessPoint (Shadow).

**Table 266 - SMI Referenced Properties/Methods for CIM\_RemoteServiceAccessPoint (Shadow)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the management interface.
SystemName		Mandatory	The name of the Computer System hosting the management interface.
CreationClassName		Mandatory	The CIM Class name of the management interface.
Name		Mandatory	The unique name of the management interface.

#### 15.5.19 CIM\_RemoteResources

An instance of a default CIM\_RemoteResources defines the set of shadow LogicalDisks that are available to be used by the Host Filesystem Profile.

CIM\_RemoteResources is subclassed from CIM\_SystemSpecificCollection.

One instance of the CIM\_RemoteResources would exist and shall be hosted by the top level ComputerSystems of the Host Filesystem Profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 267 describes class CIM\_RemoteResources.

**Table 267 - SMI Referenced Properties/Methods for CIM\_RemoteResources**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the RemoteResources collection (e.g., Remote Logical Disks).
ElementType		Mandatory	The type of remote resources collected by the RemoteResources collection. This shall be '7' (LogicalDisk).
CollectionDiscriminator		Mandatory	An array of strings indicating the purposes of the collection of elements. This shall contain 'SNIA:Imported Volumes'.

#### 15.5.20 CIM\_ResidesOnExtent

Created By: External

Modified By: Static

Deleted By: External

Requirement: Mandatory

Table 268 describes class CIM\_ResidesOnExtent.

**Table 268 - SMI Referenced Properties/Methods for CIM\_ResidesOnExtent**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The LocalFileSystem that is built on top of a storage extent.
Antecedent		Mandatory	A StorageExtent that underlies a LocalFileSystem.

#### 15.5.21 CIM\_SAPAvailableForElement

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if CIM\_RemoteServiceAccessPoint is modeled.

Table 269 describes class CIM\_SAPAvailableForElement.

**Table 269 - SMI Referenced Properties/Methods for CIM\_SAPAvailableForElement**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Shadow System.
AvailableSAP		Mandatory	

#### 15.5.22 CIM\_ServiceAffectsElement

Created By: Extrinsic: CreateFileSystem

Modified By: Extrinsic: ModifyFileSystem

Deleted By: Extrinsic: DeleteFileSystem

Requirement: Mandatory

Table 270 describes class CIM\_ServiceAffectsElement.

**Table 270 - SMI Referenced Properties/Methods for CIM\_ServiceAffectsElement**

Properties	Flags	Requirement	Description & Notes
ElementEffects		Mandatory	In this profile, the service provides management for the element. The standard allows Other to support vendor extensions. The standard values are 1 (Other) and 5 (Manages).
OtherElementEffectsDescriptions		Optional	A description of other element effects that this association might be exposing.
AffectedElement		Mandatory	The LocalFileSystem.
AffectingElement		Mandatory	The FileSystemConfigurationService.

#### 15.5.23 CIM\_StorageExtent (Primordial Imported Extent)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 271 describes class CIM\_StorageExtent (Primordial Imported Extent).

**Table 271 - SMI Referenced Properties/Methods for CIM\_StorageExtent (Primordial Imported Extent)**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	
ExtentStatus		Mandatory	This shall contain the value '16' ('Imported').
Primordial		Mandatory	This shall be 'true'.
OperationalStatus		Mandatory	Value shall be 2 3 6 8 15 (OK or Degraded or Error or Starting or Dormant).
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Pool Component' and 'SNIA:Imported'.

**15.5.24CIM\_SystemDevice (LogicalDisks)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 272 describes class CIM\_SystemDevice (LogicalDisks).

**Table 272 - SMI Referenced Properties/Methods for CIM\_SystemDevice (LogicalDisks)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Shadow Computer System that contains this LogicalDisk.
PartComponent		Mandatory	The logical disk that is managed by a computer system.

**EXPERIMENTAL**



## EXPERIMENTAL

### 16 FileSystem Replication Services Profile

#### 16.1 Description

##### 16.1.1 Synopsis

**Profile Name:** Filesystem Replication Services (Component Profile)

**Version:** 1.7.0

**Organization:** SNIA

**Central Class:** ReplicationService

**Scoping Class:** ComputerSystem

**Related Profiles:** Table 273 describes the related profiles for Filesystem Replication Services.

**Table 273 - Related Profiles for Filesystem Replication Services**

Profile Name	Organization	Version	Requirement	Description
Filesystem	SNIA	1.7.0	Mandatory	
File Export	SNIA	1.7.0	Optional	
Multiple Computer System	SNIA	1.2.0	Optional	
Job Control	SNIA	1.5.0	Optional	
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

##### 16.1.2 Overview

The Filesystem Replication Services Profile, a component profile, specifies the attributes and methods to create and manage replica of storage elements, for instance, file system. The target replica of storage element may be from the same storage system or across a connection to a different storage system.

Two types of synchronization views are supported. A replica may be synchronized to the current view of storage element or may be synchronized to a point-in-time view. Snapshots and clones always represent a point-in-time view, and a mirror represents a current view.

Two copy operation modes are supported -- synchronous and asynchronous. In the synchronous mode, the write operation to the source element is reflected to the target element before signaling the host that a write operation is complete. In the asynchronous mode, the host is signaled as soon as the write operation to the source element is complete; however, the write to the target element may take place at a later time.

Filesystem Replication Services Profile supports local and remote replication. Local replication specifies that both the source and the target element are contained in a single managed storage system, such as a NAS array platform. Remote replication specifies the source and the target element are contained in separate storage systems. For remote replication, the client may interact with both the source and target storage systems, however, the client only invokes the replication methods to single ReplicationService.

Filesystem Replication Services Profile supports "copying" thinly provisioned elements. Unlike fully provisioned elements, a thinly provisioned element has fewer actual allocated space than the advertised capacity of the element.

FileSystem Replication Services Profile supports copy operation from and to ReplicationEntity which represents an addressable entity without a known object model.

Throughout this profile, there are specific references to class, properties and methods pertaining to each section. Refer to 16.5: CIM Elements for a complete list of all properties and methods, including the description.

#### 16.1.2.1 Key Feature

The following is a brief list of key feature of the Filesystem Replication Services:

- The ability to efficiently retrieve replication relationships
- The ability to support the different *Copy Methodologies*, for example, mirror, snapshot and clone.
- The ability to support the different mode, for example, synchronous and asynchronous.
- The ability to handle local and remote replication seamlessly
- The ability to specify *Individual* or *Groups* of elements to manage replication
- The ability to copy from and to undiscovered resources
- The ability to support *Consistency Management*
- The ability to replicate Thinly Provisioned element

#### 16.1.2.2 Key Components

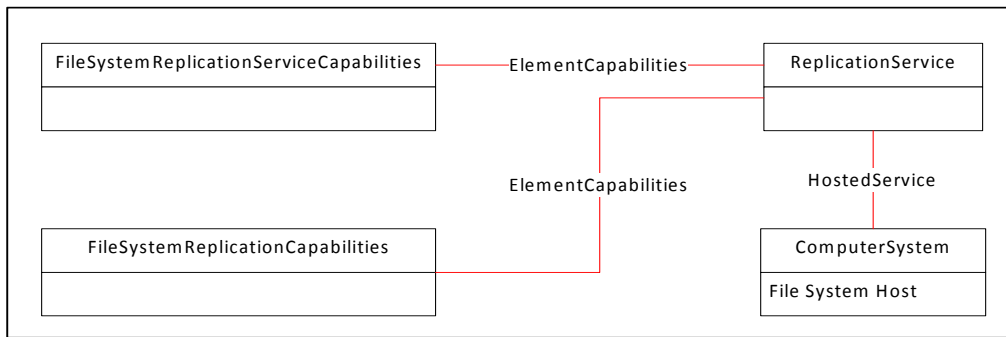
Table 274 shows a list of key classes used by Filesystem Replication Services. Refer to 16.3: Methods and 16.5: CIM Elements for addition details on methods and properties of these classes.

**Table 274 - Key Components**

Class Name	Notes
ReplicationService	The main class for Replication Services. It contains methods for replication and group management, for example, CreateGroup, CreateElementReplica, CreateGroupReplica, ModifyReplicaSynchronization
FileSystemReplicationServiceCapabilities	Contains a set of properties and methods that describe the capabilities of the service, for example, SupportedReplicationTypes, GetSupportedFeatures.
FileSystemReplicationCapabilities	Contains a set of properties and methods that describe the capabilities of each supported SupportedReplicationType.
ReplicationGroup	Represents a group of elements participating in replication activities.
ReplicationSettingData	Contains options to customize replication operations, for example, pairing of group elements, TargetElementSupplier, CopyMethodology, ThinProvisioningPolicy.
ReplicationEntity	Represents information about an addressable entity without a known object model.
FileSystemGroupSynchronized	Associates source and target groups
FileSystemSynchronized	Associates source and target elements.

#### 16.1.3 Filesystem Replication Services Discovery

Figure 30 shows a diagram of Filesystem Replication Services Discovery.



**Figure 30 - Replication Service Discovery**

The ComputerSystem has a HostedService association with ReplicationService. The single instance of the class ReplicationService and its methods provide the mechanism for the creating and managing the replicas.

The single instance of the class FileSystemReplicationServiceCapabilities and its methods describe the various capabilities of the service. Clients should examine the FileSystemReplicationServiceCapabilities instance and invoke its methods to determine the specific capabilities of a replication service implementation.

The instances of FileSystemReplicationCapabilities may be associated with ReplicationService using ElementCapabilities. Each instance of FileSystemReplicationCapabilities should be for each supported FileSystemReplicationServiceCapabilities.SupportedReplicationTypes.

#### 16.1.3.1 SyncTypes

SyncTypes describe the replication policy support by the profile. The following SyncTypes are defined:

**Mirror:** Creates and maintains a synchronized mirror copy of the source. Write done to the source element are reflected to the target element. The target element remains dependent on the source element.

**Snapshot<sup>1</sup>:** Creates a point-in-time, virtual image of the source element. The target element remains dependent on the source element. Identical information in the source and target elements are shared via implementation-dependent means, to achieve space savings compared to full copies. Snapshots are commonly known as delta replicas.

**Clone:** Create a point-in-time, independent, copy of the source element.

Synchronized replication indicates that updates to a source element are reflected to the target element. The mode determines whether the target element updated immediately, in the case of synchronous mode, or some time later, in the case of asynchronous mode.

1. Industry usage of the term 'snapshot' varies widely. In this specification, it is used to mean a 'delta snapshot' as defined in the *SNIA Dictionary*.

Table 275 compares the SyncTypes and the relationships between the source and target elements. It is a quick reference for the clients to determine the appropriate SyncType for the intended target results.

**Table 275 - Comparing SyncTypes**

SyncType	Relation of Target to Source	Updates to Source Reflected to Target	Target is Point-in-Time Copy	Target is self-contained	Target is Virtual copy of Source	Target's space consumption
Mirror	Dependent	Yes	No	Yes after Split/Detach	NO	Same as Source
Snapshot	Dependent	No	Yes	No	Yes	Less than Source
Clone	Independent	No	Yes	Yes	No	Same as Source

With respect to “Relation of Target to Source”, **Dependent** indicates the target element must remain associated with the source element; **Independent** indicates the target element can exist without the source element.

With respect to “Target is Virtual copy of Source”, the target element is not a “physical” copy of the source element, instead the system holds a collection of mapping information that map the target element data to the source element data.

#### 16.1.3.2 Mode

The mode controls when the write operations are performed. The following modes are defined:

**Synchronous:** The writer waits until the write operations are committed to both the source and target elements; or to both the source element and a target related entity, such as pointer tables.

**Asynchronous:** The writer waits until the write operations are committed to the source elements only. In this mode, there can be a delay before the write operations are committed to the target elements.

#### 16.1.4 Locality of Target Elements

Locality specifies the relationship between the source and target element. Replication Services defines the following localities:

**Local:** It indicates the source and target elements are contained in a single managed system.

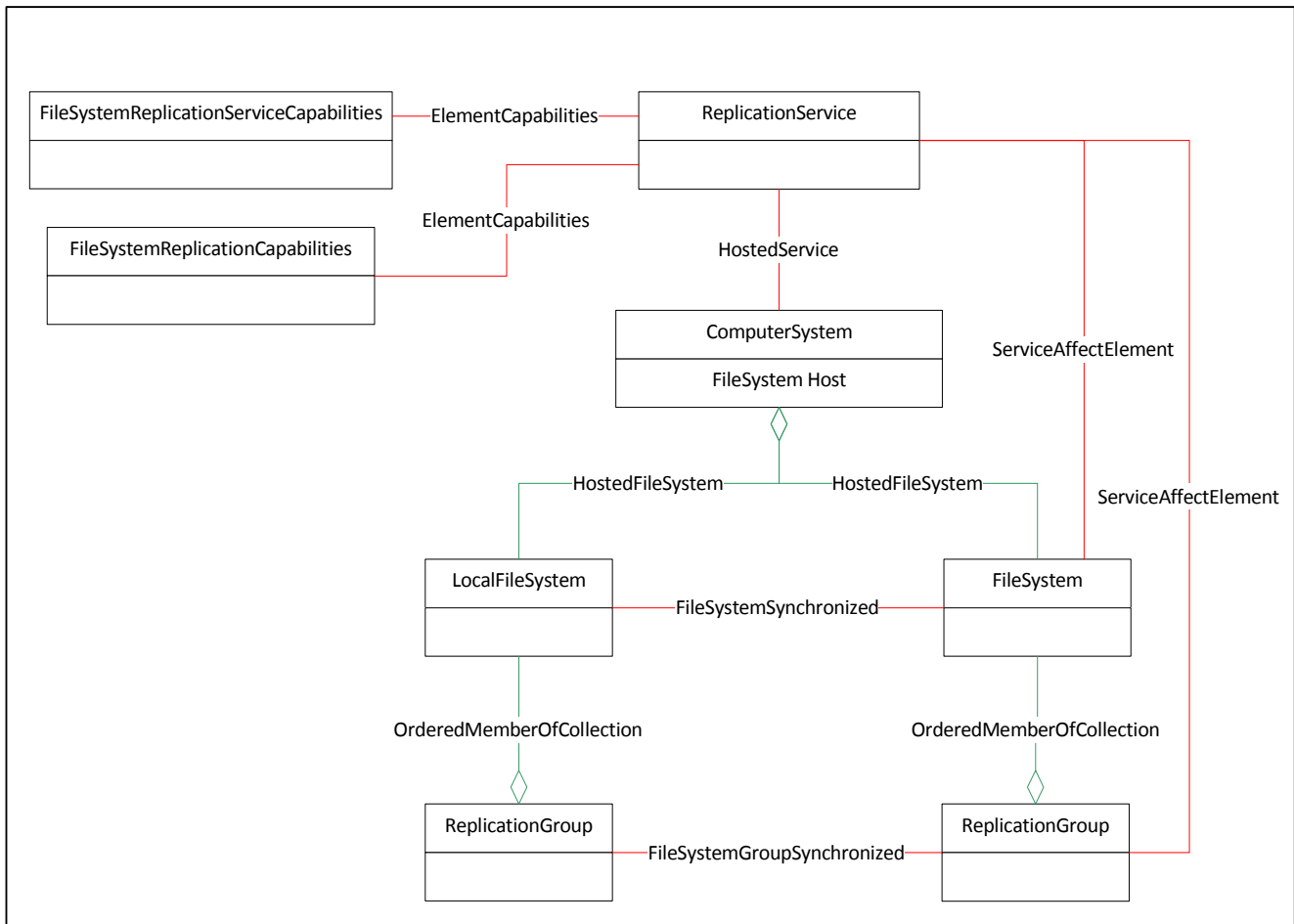
**Remote:** it indicates the source and target element are contained in separate managed systems. In this case, the service must rely on a networking protocol for the copy operations.

The networking protocols are modeled using ProtocolEndpoint, which enables a replication service to reach a remote element. The property ProtocolEndpoint.ProtocolType specifies the protocol type, for example, TCP, Fibre Channel, Other, etc.

Locality is important because it advertises the capability of replication service. For example, the property FileSystemReplicationServiceCapabilities.SupportedReplicationType may have values such as “Synchronous Mirror Local” and “Synchronous Mirror Remote”.

Figure 31 illustrates the mandatory, optional and conditional classes for the modeling of local replication instance diagram.



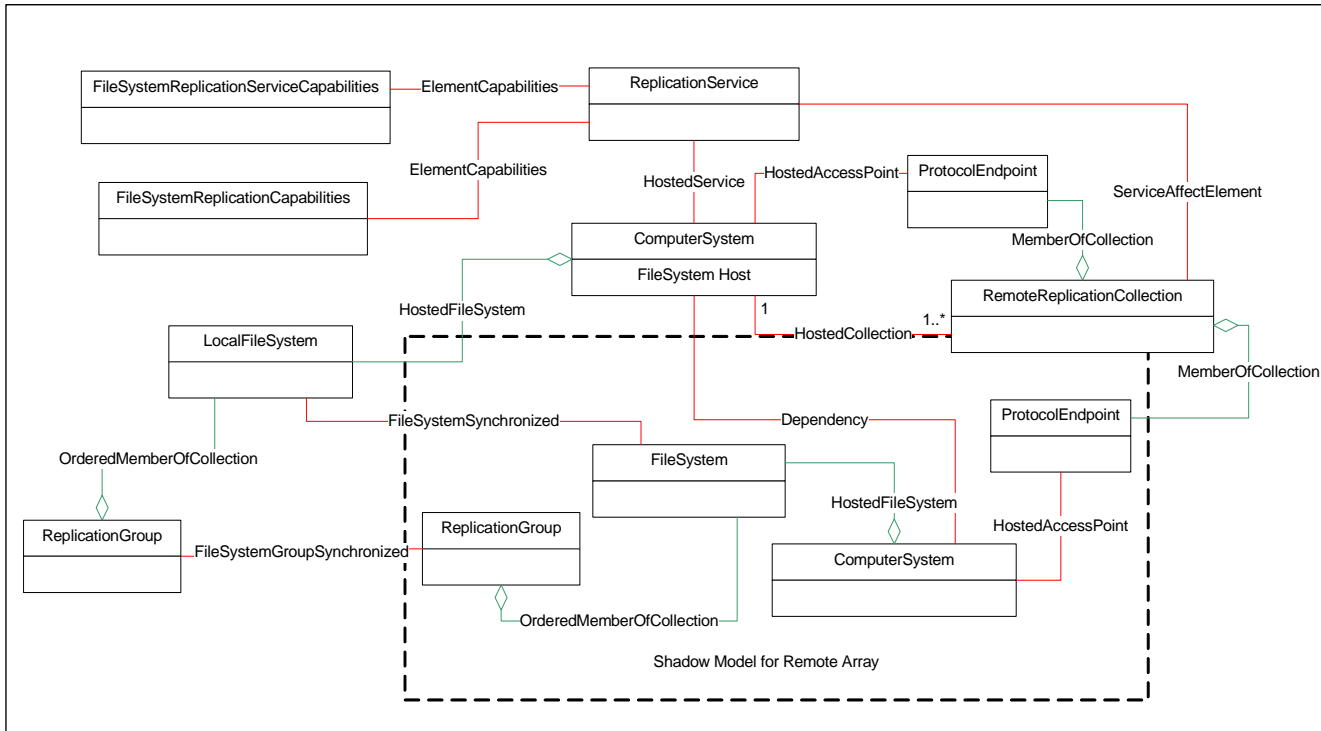


**Figure 31 - I Local File System Replication**

The source element shall be represented in the model as an instance of LocalFileSystem and have a FileSystemSynchronized association to the target element, that is, FileSystem. The association's property CopyState indicates the current state of the association. Meanwhile, the ReplicationService instance shall have a ServiceAffectElement association to target replica and/or target group.

Both source element and target replica shall have a HostedFileSystem association to a ComputerSystem. Normally it will be the top-level ComputerSystem of the parent profile (typically one of the filesystem-related profiles such as the NAS Head or the Self-Contained NAS Profile). However, if the Multiple Computer System Profile is implemented, the HostedFileSystem may be associated to a component ComputerSystem. See 16: FileSystem Replication Services Profile in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*.

Figure 32 illustrates the mandatory, optional and conditional classes for the modeling of remote replication instance diagram.



### Figure 32 - Remote File System Replication

The `RemoteReplicationCollection` abstracts the details of network connections to a remote system to allow clients to focus on whether a remote system is reachable or not.

Instance of `RemoteReplicationCollection` may statically be created by the implementation, or clients may be required to create such instance by invoking the extrinsic method `CreateRemoteReplicationCollection`. Client subsequently can manipulate instances of `RemoteReplicationCollection` by invoke the intrinsic method `ModifyInstance` and/or the extrinsic method `AddToRemoteReplicationCollection` and `RemoveFromRemoteReplicationCollection`.

Each instance of `RemoteReplicationCollection` can have one or more paths to the remote system. As long as one of these path to the remote system is up, the property `RemoteReplicationCollection.ConnectivityStatus` indicates “UP”. As long as one connection is functioning, there are replication operations between the local and the remote system.

Remote replication may require access information such as an RemoteServiceAccessPoint instance for the remote resource. See 16.2.2: Cascading Considerations for addition information.

### 16.1.5 Group

FileSystem Replication Services utilizes group of element to manage replication activities that include more than one source or target element in a copy operation. A major advantage of using groups is that an operation, such as *fracture* may be performed on the group as a whole, instead of fracturing individual element pairs one by one. The optional ReplicationGroup class represents a collection of ordered storage elements. An implementation may allow the target group to have more (or fewer) elements than the source group.

Key feature of replication groups are:

- A group can be the source and/or the target of a copy operation.

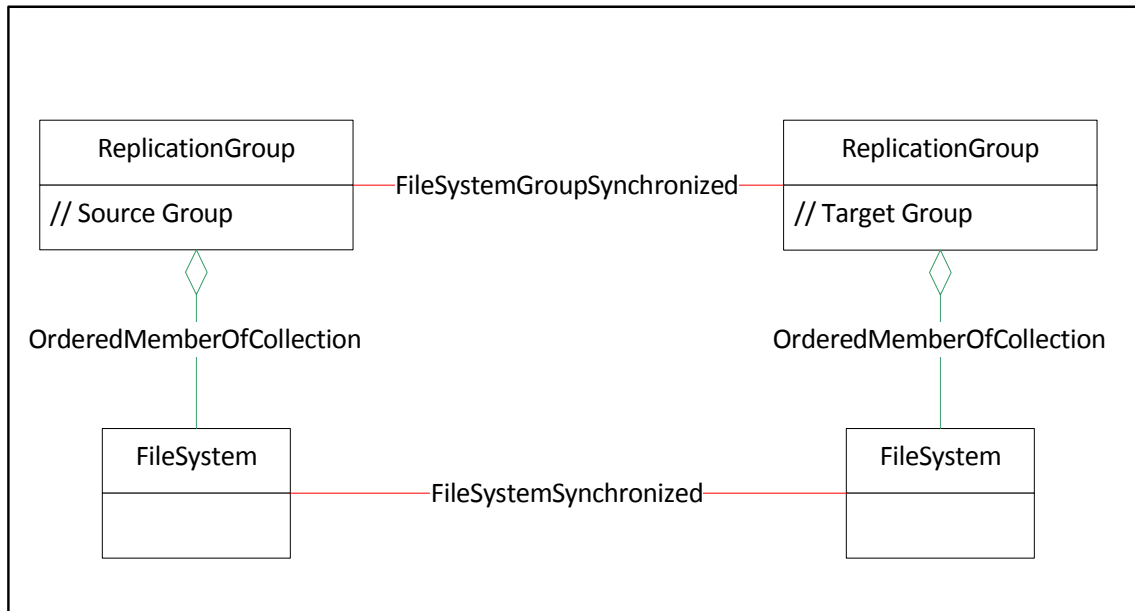
- Elements of a group may be optionally declared *Consistent*
- A group may optionally be declared as temporary (Persistent = false).
- A group may contain zero elements (an empty group)

FileSystem Replication Services includes the methods to create and delete a group, and the methods to add the elements or pair of elements to an existing group(s) or to remove elements from a group.

Certain copy operations such as copy one source to many target elements (one-to-many) may result in the service creating a temporary group to keep track of all the target elements. The service may delete the temporary group that is no longer associated with a copy operation. Deleting a temporary group does not affect the elements associated with the group.

Figure 33 shows a diagram of the group instance. The method `ReplicationService.CreateGroupReplica()` is used to copy a group of elements. The property `ReplicationSettingData.Pairing` determines the pairing of the source and the target elements. Possible values are *Exact Order* and *Optimum*. *Exact Order* means the first element of the source group is copied to the first element of the target group, the second element of the source group is copied to the second element of the target group, and so on. *Optimum* means in order to minimize any resource and data flow contentions, if possible, pair the source and target elements in such a way that they are on different data paths.

See the `FileSystemReplicationServiceCapabilities.GetSupportReplicationSettingData()` method for `Pairing` and for `UnequalGroupAction` Capabilities.



**Figure 33 - Group Instance Diagram**

The association between `ReplicationGroup` and its storage elements (e.g., `FileSystem`) is `OrderedMemberOfCollection` to maintain the order of the storage elements to facilitate pairing of the source and the target group elements.

#### 16.1.5.1 Composite Group

A Composite Group is a group that includes storage elements from multiple storage systems.

### 16.1.5.2 Consistency Group

A Consistency Group is a set of elements that have an “Application Consistent view”. Application Consistent View is a set of the elements that collectively represent some resource in a known state.

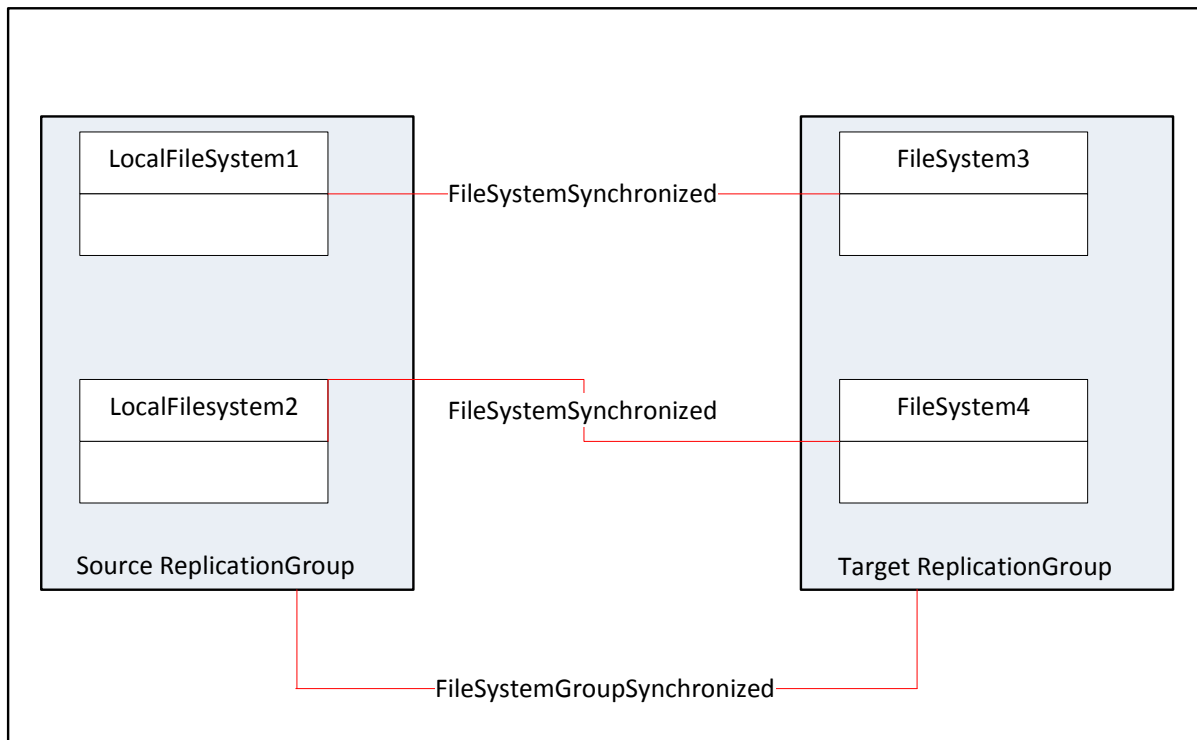
#### 16.1.5.2.1 Sequentially consistent

A group of target element is considered to be “sequentially consistent” if each element is updated in the same order as the application updates the corresponding source elements. Sequentially Consistency is also known as Dependent Write Consistency.

### 16.1.5.3 FileSystemGroupSynchronized Association

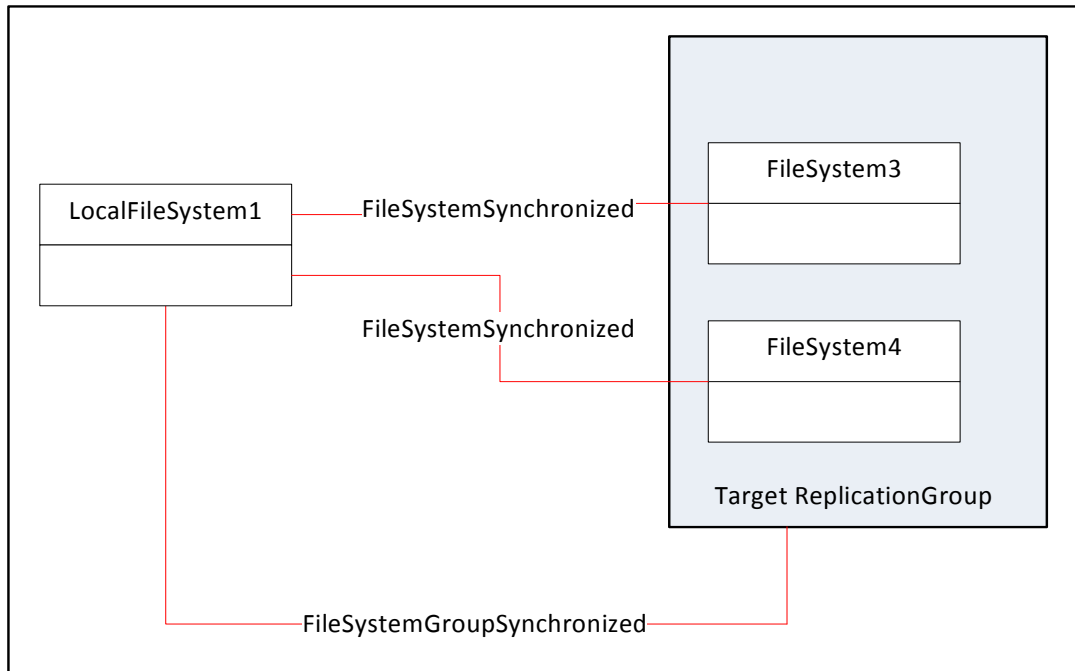
.FileSystem Replication Services utilizes FileSystemGroupSynchronized to associate one pair of source and target groups or a source element to a target group for a one-to-many relationship. Within a group, the SyncType and Mode properties of all subordinate FileSystemSynchronized associations between the resource and the target elements shall be the same. The SyncType and Mode properties of the FileSystemGroupSynchronized association shall also be the same as the ones of subordinate FileSystemSynchronized associations.

Figure 34 shows the associated groups with equal number of source and target elements.



**Figure 34 - Associated Group and Elements**

As shown in Figure 35, one source element is associated to more than one target element.



**Figure 35 - One-to-Many Association**

If the property `ConsistencyEnabled` set to true, the target elements have a sequentially consistent view at all time. Within a group, once the connection between the individual source and target element is broken, all subsequent copy operations to the target elements stop, therefore maintaining the consistency of the target element.

#### 16.1.6 State Management For Associated Replicas

Both mirror and snapshot replicas maintain stateful associational with source elements. In the case of clone replica, the replication associations to the source elements exist while the copy operation is in progress.

The `CopyState` property of the replication association identifies the state, while the `ProgressStatus` property of the same association indicates the “status” of the copy operation to reach the requested `CopyState`, which is indicated in the property `RequestedCopyState`. For example, `CopyState` might have a value of “UnSynchronized”, while `ProgressStatus` might have a value of “Synchronizing”, also known as “sync-in-progress”. In all cases, when creating a replica element, the desired `CopyState`, as reflected in the property `RequestedCopyState`, is `Synchronized`, which indicates the replica element has the same data as the source element. The `RequestedCopyState` property will contain “Not Applicable” once the requested `CopyState` is achieved.

The `FileSystemGroupSynchronized` association between the source and target groups also includes the `CopyState` property. If all values of `FileSystemSynchronized.CopyState` of the source and target association are the same (i.e., `Synchronized`), `FileSystemGroupSynchronized.CopyState` will also have the same value. On the other hand, any mismatch in the `FileSystemSynchronized.CopyState` values, will render the `FileSystemGroupSynchronized.CopyState` property to have a value of *Mixed*.

Synchronized state for the Mirror and Clone `SyncType` indicates all data has been copied from the source element to the target element. For the Snapshot `SyncType`, because the target element is a virtual point-in-time view of the source element, the Synchronized `CopyState` indicates all the metadata (pointer or mapping information) for the snapshot have been created. Synchronization for the snapshot is achieved rapidly in comparison to synchronization of Mirror and Clone.

Unsynchronized CopyState indicates the target element is not an exact copy of the source element (or the source's point-in-time). The copy operation automatically continues until the synchronization between the source element (or its point-in-time) and the target element is reached.

The Skewed CopyState is similar to the Unsynchronized CopyState except that the synchronized relationship remains in the Skewed state until a client issues the Resync operation (the extrinsic methods ModifyReplicaSynchronization or ModifyListSynchronization). As an example: Committing write operations to a Snapshot target element causes the source and the target elements to become Skewed.

Unplanned states, such as Broken, Aborted or Partitioned can be entered from any other state and generally indicate an unusual circumstance. Recovery from the Broken or Partitioned state may be automatic once the error condition is resolved, or it may require a client to intervene with a "Resync" operation (See 16.3.3.3: GetSupportedFeatures) or a "Resume" operation. Continuing from an Aborted state requires a client to intervene with a "Resync" operation. In this situation, the implementation may indicate a "Resync" operation is required by the setting the ProgressStatus to "Waiting for resync". Additionally, the copy operation may be temporarily stopped due to system or connection bandwidth. In this case the ProgressStatus will be set to "Pending". See 16.3.3.3: GetSupportedFeatures

If after the error condition is resolved, the CopyState indicates "Suspended" State, in order to resume the copy operation it is necessary for the client to issue a "Resume" operation.

If the CopyState indicates "Invalid", generally, it means system is unable to determine the state of the copy operation. In this situation, the client needs to "detach" and "reestablish" the replication relationship.

Use the method FileSystemReplicationServiceCapabilities.GetSupportedCopyState to determine the possible CopyStates. The CopyStates have been normalized in such a way that they may apply to all SyncTypes.

Table 276 describes the supported CopyStates.

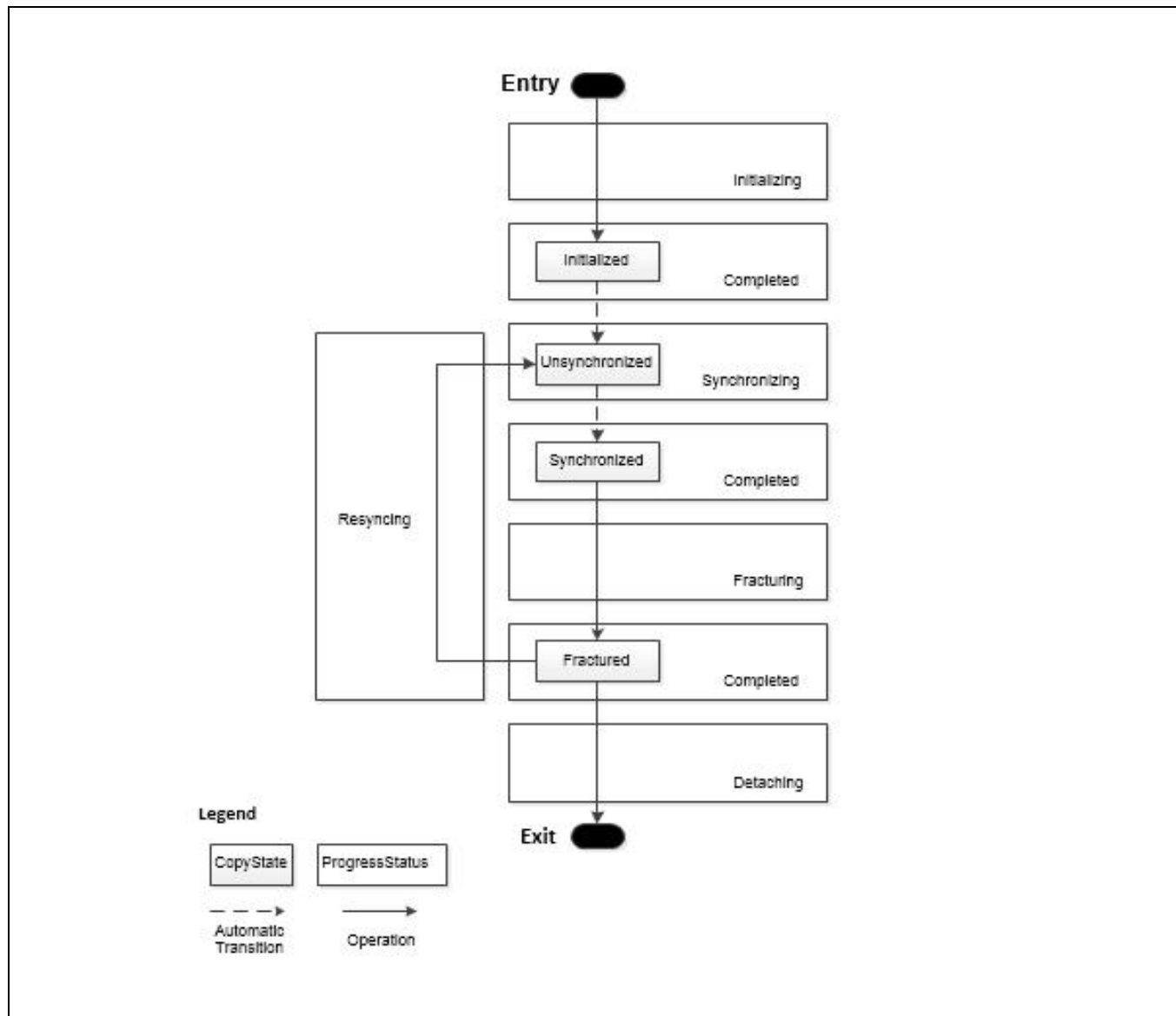
**Table 276 - CopyStatus Values**

CopyState value	Description
Initialized	The source and target elements are associated. The copy operation has not start - no data flow.
Unsynchronized	Not all the source element data has been "copy" to the target element.
Synchronized	The copy operation is complete. The target element is an "extra replica" of the source element.
Broken	Replica is not valid view of the source element. OperationalStatus of replica may indicate an Error condition. This state generally indicates an error condition such as broken connection.
Fractured	The target element was abruptly split from its source element - consistency is not guaranteed
Split	The target element was gracefully (or systematically) split from its source element - consistency is guaranteed.
Inactive	Copy operation has stopped, writes to source element will not be sent to the target element.
Suspended	Data flow between the source and target element has stopped. Writes to source element are held until a resume operation is completed.
Failedover	Reads and writes to/from the target element. Source element is not "reachable"

**Table 276 - CopyStatus Values**

CopyState value	Description
Prepared	Initialization is completed, the copy operation has started, however, the data flow has not started.
Aborted	The copy operation is aborted with the Aborted operation. Use the Resync Replica operation to restart the copy operation
Skewed	The target has been modified and is no longer synchronized with the source element or the point-in-time view. Use the Resync Replica operation to resynchronized the source and target element.
Mixed	Applies to the CopyStatus of FileSystemGroupSynchronized. It indicates the FileSystemSynchronized associations of the elements in the groups have different CopyState values.
Partitioned	The state of replication relationship can not be determined, for example, due to a connection problem.
Invalid	The array is unable to determine the state of the replication relationship, for example, after the connection is restored; however, either source or target element has an unknown status.
Restored	It indicates the source element was restored from the target element.

Figure 36 shows a sample of the CopyState transitions and corresponding ProgressStatus changes. In a steady state condition, for example, the CopyState has a value of “Synchronized” and at the same time the ProgressStatus has a value of “Completed”.



**Figure 36 - Sample CopyState and ProgressStatus Transitions**

Depending on implementation, the clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation. See the method `FileSystemReplicationServiceCapabilities.GetSupportedFeatures` in 16.3.3.3.

#### 16.1.7 Undiscovered Resource

An undiscovered resource is any addressable entity without a known object model. General, clients identify an undiscovered resource using one or more of the following:

- WWN (World Wide Name)
- URI (Uniform Resourced Identifier)

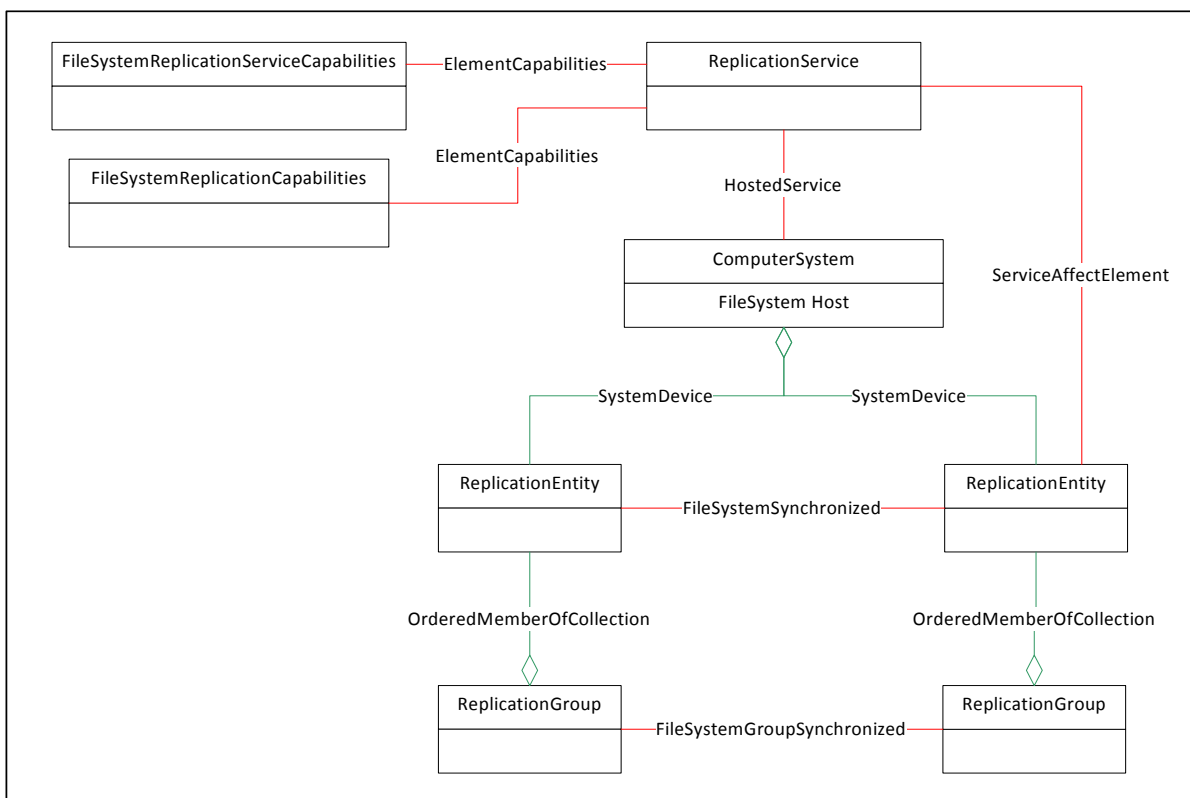


- IP Address
- Remote ComputerSystem Objectpath

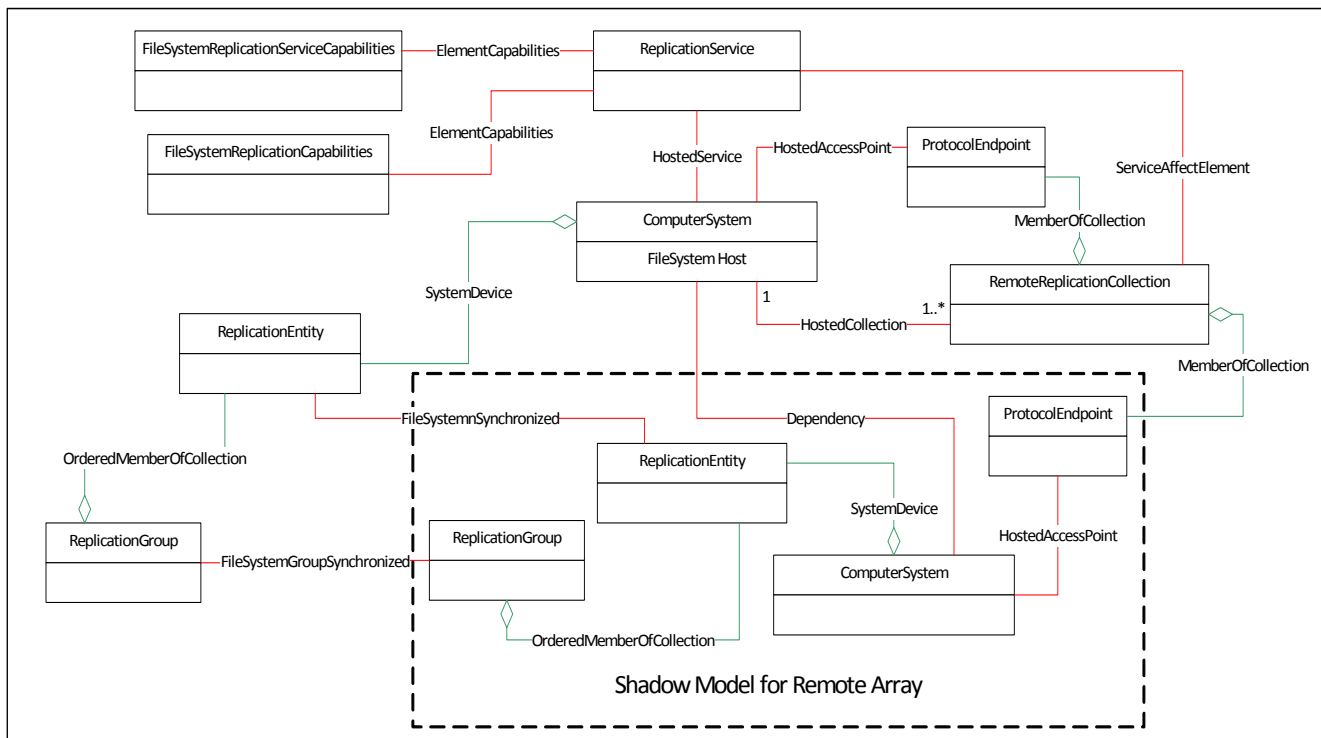
In all cases, the assumption is that the underlying implementation “knows” how to perform the copy operation.

the FileSystem Replication Service includes the necessary methods to create and manage the instances representing undiscovered resource. See the class ReplicationEntity (in 16.5: CIM Elements and the method AddReplicationEntity (16.3.2.15). Also, in the replication service capabilities the absence of “Requires full discovery of target ComputerSystem” in SupportedFeatures property indicates the service support undiscovered resources.

Figure 37 and Figure 38 show entire instance diagram with ReplicationEntity for local and remote replication.



**Figure 37 - Local Replication with ReplicationEntity**



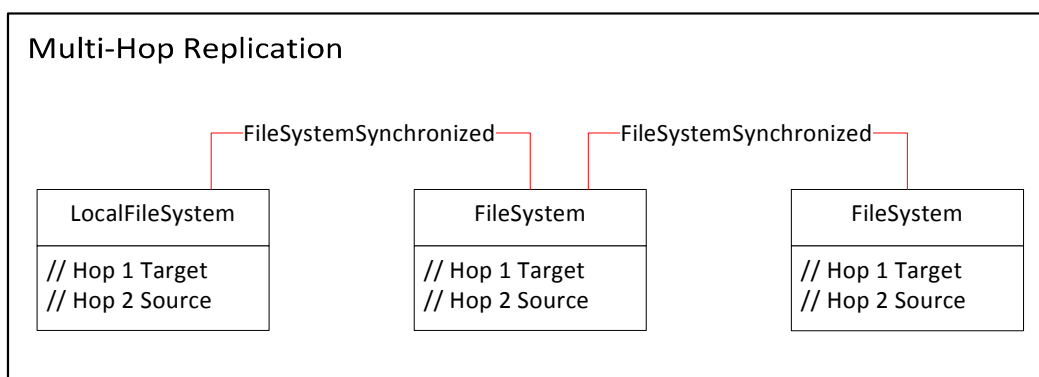
### Figure 38 - Remote replication with ReplicationEntity

An instance of `FileSystemSynchronized` association identifies the source and target element of a copy operation. Additionally, the `FileSystemSynchronized.UndiscoveredElement` property may indicate which elements in the copy operation are “undiscovered”. The possible values are:

- SystemElement - the source element
- SyncedElement - the target element
- Both - both the source and target elements

### 16.1.8 Multiple-Hop Replication

In multi-hop replication, the target element of one copy operation can simultaneously be the source for another copy operation. As shown in Figure 39, multi-hop replication involves at least three elements.



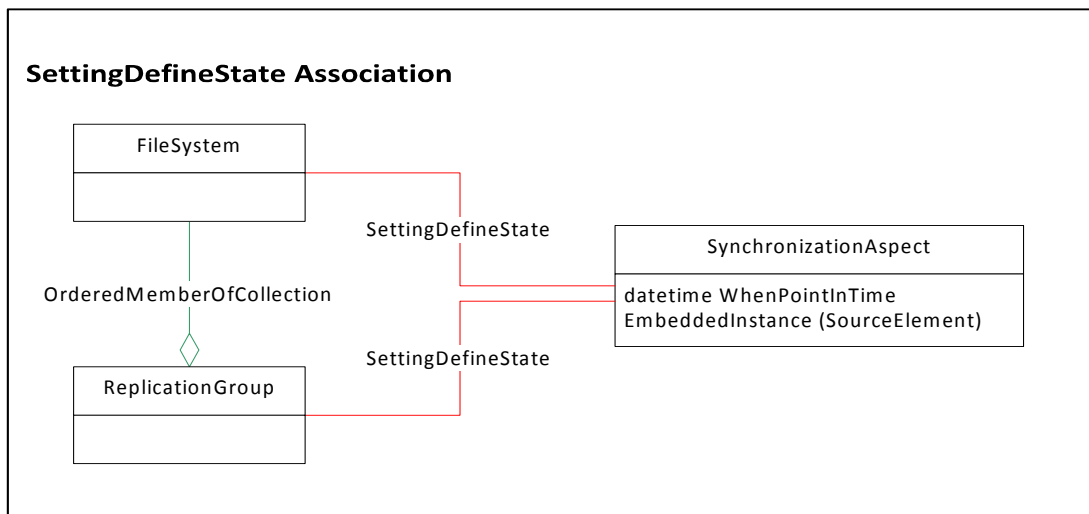
**Figure 39 - Multi-Hop Replication**

If an implementation supports multi-hop replication, the supported features capabilities will indicate “Multi-Hop element replication”. Furthermore, the implementation may need to know that the client is planning to add additional hops in the subsequent operations. In this case, the replication capabilities would indicate “Multi-hop requires advance notice”. In response to this capability, the client in creating the first replica, must set the property `ReplicationSettingData.Multihop` appropriately (see 16.5: CIM Elements for details on Multi-hop specification). The capabilities method `GetSupportedMaximum` indicates the maximum number of hops supported by the implementation.

### 16.1.9 SettingDefineState Association and SynchronizationAspect Instance

The `SettingDefineState` associates an element (e.g. a `FileSystem`) or a group of elements (e.g. a `ReplicationGroup`) to a `SynchronizationAspect`. An instance of `SynchronizationAspect` includes properties for the data and time of the point-in-time copy and a reference to a source element (see Figure 40). The association is particularly useful for Clones (targets) and Snapshot (source) that do not have a `FileSystemSynchronized` association to another storage element. In the case of Clone, the `FileSystemSynchronized` association is removed (generally, following the provider’s restart) after the copy operation completes. As for Snapshot, it is possible to create a point-in-time snapshot copy of an element or a group of elements, without having a target element (using the method `CreateSynchronizationAspect`). In this mode, the target elements are added at a later time (using the method `ModifySettingDefineState`). Creating a `SynchronizationAspect` of a Snapshot is particularly useful when a client wants to capture a point-in-time copy at a given time; however, the client wants to create an actual target element at a later time, perhaps when it is more convenient.

If an instance of a `SynchronizationAspect` is associated to a group of elements, the property “WhenPointInTime” applies to all elements of the groups, indicating the point-in-time copy of all elements is created at the same exact time.

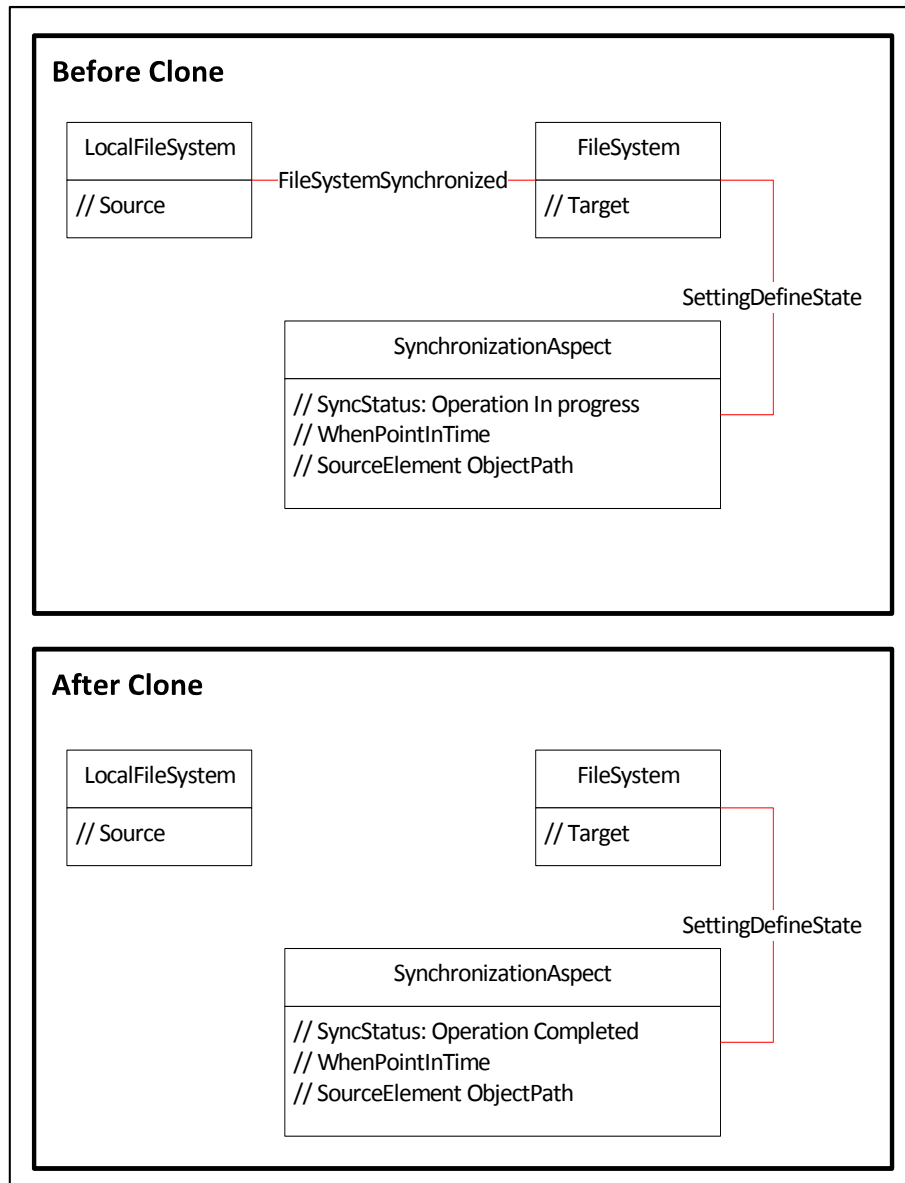


**Figure 40 - SettingDefineState**

`SettingDefineState` may also be applied to Mirror targets; as much the property `SynchronizationAspect.WhenPointInTime` would have the date and time of when the mirror relationship was fractured (or split).

In all cases, the `SettingDefineState` association may not persist across the provider’s restart. Furthermore, an instance of a `SynchronizationAspect` shall be removed if the `SourceElement` is deleted.

Figure 41 shows an instance diagram for a clone target element and its associated SynchronizationAspect instance. Once the clone target element becomes synchronized, the FileSystemSynchronized association is removed and the property SynchronizationAspect.CopyState has a value of “Operation Completed”



**Figure 41 - SynchronizationAspect Instance Diagram**

#### 16.1.10Indication

Depending on the implementation, the FileSystem Replication Services Profile generates a number of different alert and lift cycle indications, as shown in Table 277. clients decide what indications they wish to receive by subscribing to the appropriate indications.

Because on the large system with many copy operation in progress simultaneously, there is a potential to receive many unwanted indications. Therefore, it is recommended for the client to subscribe to

indications that have a query that is constrained to a specific replication association. See 16.5: CIM Elements for the indication queries.

For the file system and job indications, refer to Section 8: Filesystem Profile, Section 9: Filesystem Manipulation Profile and *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* Section 23: Job Control Profile.

**Table 277 - Indications**

Indication	Source Of
CIM_InstCreation	<ul style="list-style-type: none"> <li>• New Job Creation</li> <li>• New Target Element Creation</li> <li>• New FileSystemSynchronized Association Creation</li> <li>• New FileSystemGroupSynchronized Association Creation</li> </ul>
CIM_InstDeletion	<ul style="list-style-type: none"> <li>• Job Deletion</li> <li>• Target Element Deletion (e.g. Snapshot)</li> <li>• FileSystemSynchronized Association Deletion</li> <li>• FileSystemGroupSynchronized Association Deletion</li> </ul>
CIM_InstModification	<ul style="list-style-type: none"> <li>• Job Progress and Status Change</li> <li>• Source and Target Element Status Changes</li> <li>• CopyState Changes</li> <li>• ProgressStatus Changes</li> <li>• ProtocolEndpoint and RemoteReplicationCollection Status Changes</li> </ul>
CIM_AlertIndication	<ul style="list-style-type: none"> <li>• Error conditions, such as                             <ul style="list-style-type: none"> <li>• FileSystemSynchronized and FileSystemGroupSynchronized State set to <i>Broken</i></li> <li>• ProtocolEndpoints.OperationalStatus set to Error</li> <li>• RemoteReplicationCollection.ConnectivityStatus set to “down”</li> </ul> </li> </ul>

## 16.2 Implementation

### 16.2.1 Health and Fault Management Consideration

The profile uses indications to report health and fault management. In general, instance modification indications are sent when changes in OperationalStatus and HealthState values of the following instances indicate a fault condition:

- Source and Replica elements
- ProtocolEndpoints
- RemoteReplicationCollections

In response to a fault indication, clients can follow the `RelatedElementCausingError` association between instance reporting the error and the faulted component.

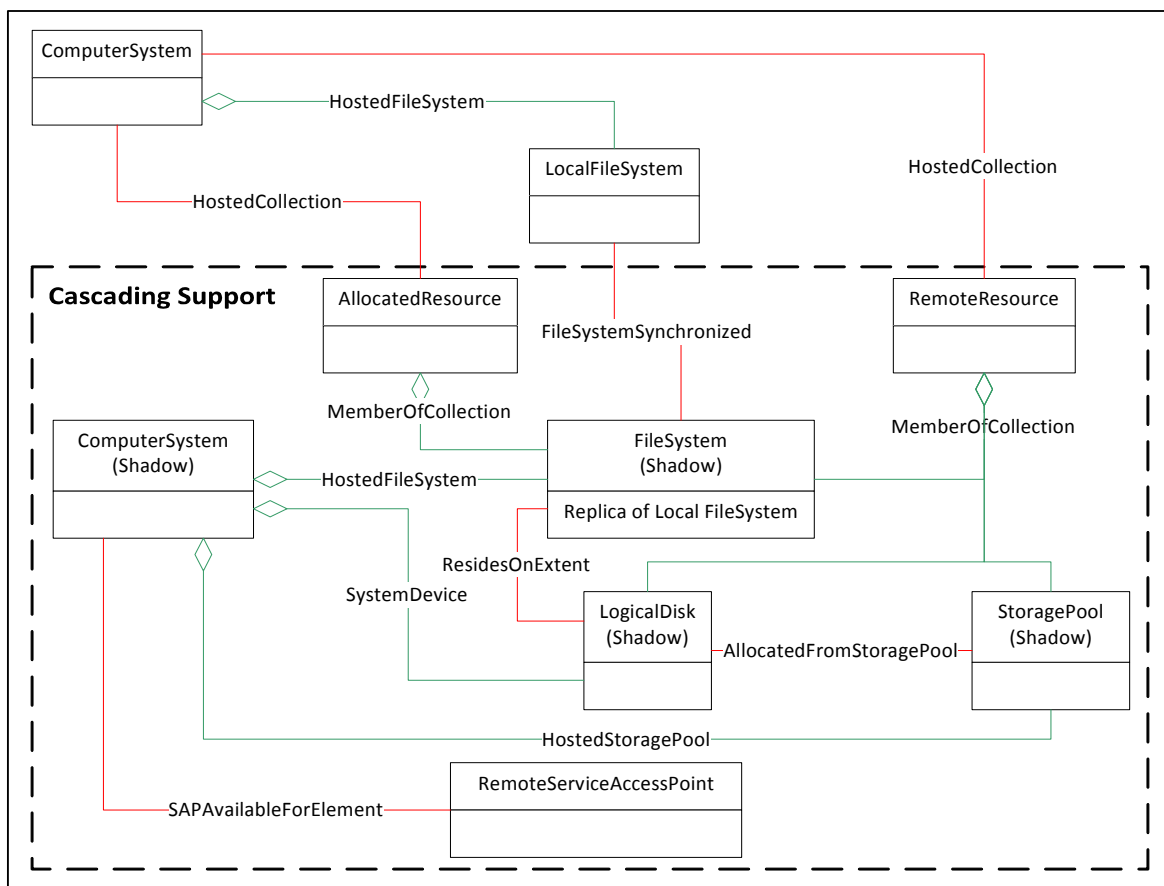
The profile also generates alert indications when the CopyState of a replication association transitions to the *Broken* state.

### 16.2.2 Cascading Considerations

For remote replication, the FileSystem Replication Services Profile requires a cascading provider to perform the “stitching” of resources between cascading profile (FileSystem Replication Services Profile) and a leaf profile (for example, NAS Head Profile), where the remote resource are contained. The cascading provider ensures that the leaf resource represent real instances of ComputerSystem, ProtocolEndpoint and storage objects such as FileSystem in cascading profile. Furthermore, the cascading provider shall ensure that the state and status properties such as OperationalStatus and CopyState have consistent values between the leaf and real resource.

The replication service relies on other profile to facilitate access to the leaf resource. For example, the RemoteServiceAccessPoint instance identifies the necessary information to establish access to the leaf system's resources.

Figure 42 illustrates the FileSystem Replication Services support for cascading.

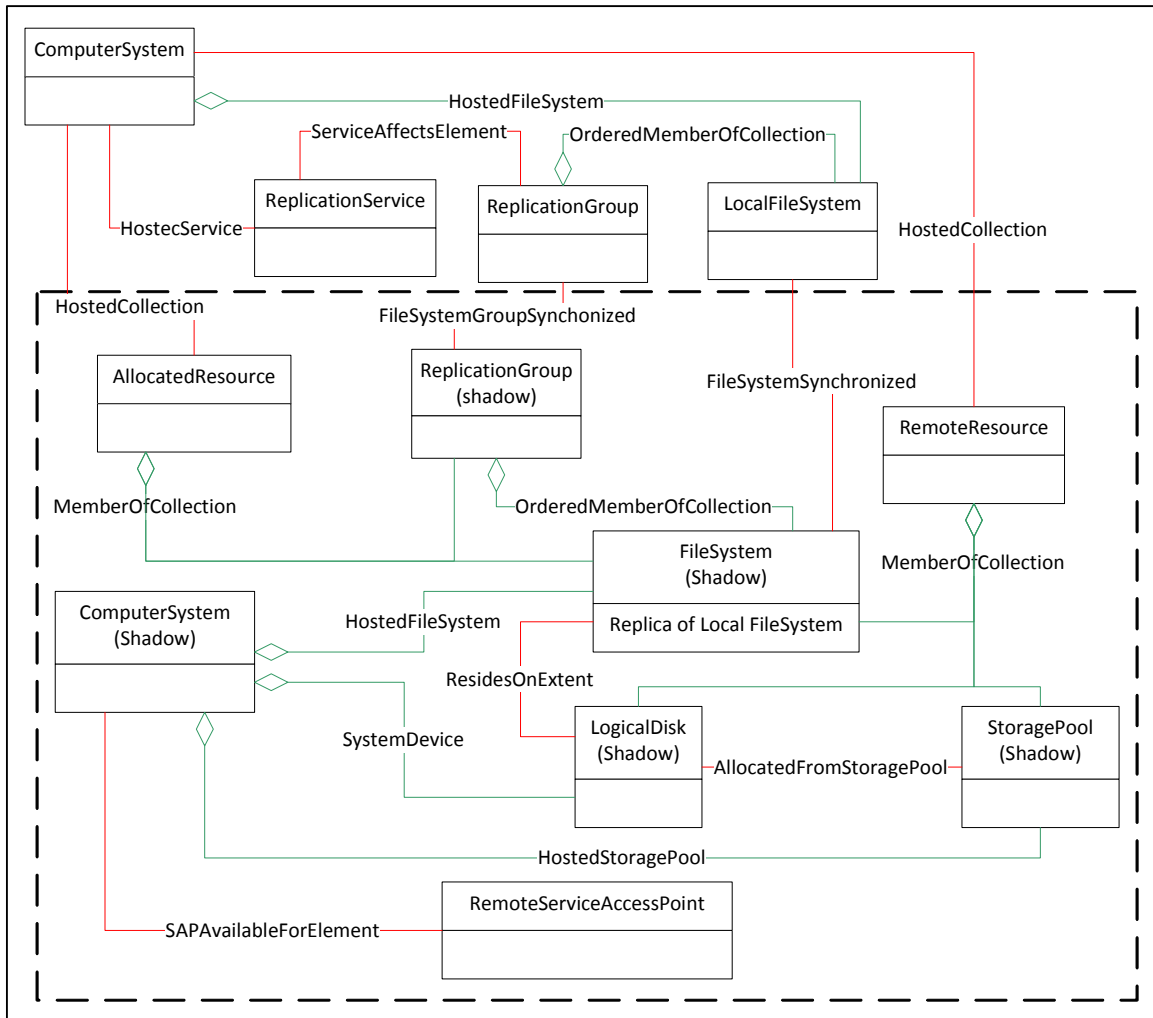


### Figure 42 - FileSystem Replication Service support for Cascading

The dashed classes are shadow of instances provided by the remote system. The collection CIM\_AllocatedResources collects all the components in use by the replication service. the collection

CIM\_RemoteResources collects all component (FileSystem, FileShare, etc.) accessible to the replication service whether used or not.

Figure 43 shows cascading support utilizing replication groups.



### Figure 43 - Cascading and Replication Groups

## 16.3 Methods

The FileSystem Replication Services Profile has an umber of the extrinsic methods for group management and replication management. Additionally, there are a number of the extrinsic methods in the FileSystemReplicationServiceCapabilities that advertise the implemented replication service capabilities. Also, the FileSystem Replication Services Profile is dependent on the other extrinsic methods provided by the Filesystem Manipulation Profile for file system manipulations. Furthermore, it relies on a number of intrinsic methods such as ModifyInstance, DeleteInstance for certain optional capabilities.

All of the FileSystem Replication Service Profile extrinsic methods return one of the following status codes. Depending on the error condition, a method may return additional error codes and/or throw an appropriate exception to indicate the error encountered.

- 0: (Job) Completed with no error

- 1: Method not supported
- 4: Failed
- 5: Invalid Parameter
- 4096: Method Parameters Checked - Job Started

For the input/output parameter values, refer to the appropriate MOF files and the value maps.

Table 278 summarizes the extrinsic methods for group management (class ReplicationService)

**Table 278 - Extrinsic Method for Group Management**

Method	Described in
CreateGroup()	See 16.3.1.1
DeleteGroup()	See 16.3.1.2
AddMembers()	See 16.3.1.3
RemoveMembers()	See 16.3.1.4

Table summarizes the extrinsic methods for replication management (class ReplicationService)

**Table 279 - Extrinsic Method for Replication Management**

Method	Described in
CreateElementReplica()	See 16.3.2.1
CreateGroupReplica()	See 16.3.2.2
CreateListReplica()	See 16.3.2.3
CreateGroupReplicaFromElements()	See 16.3.2.4
CreateSynchronizationAspect()	See 16.3.2.5
ModifyReplicaSynchronization()	See 16.3.2.6
ModifyListSynchronization()	See 16.3.2.7
ModifySettingsDefineState()	See 16.3.2.8
ModifyListSettingsDefineState()	See 16.3.2.9
GetAvailableTargetElements()	See 16.3.2.10
GetPeerSystems()	See 16.3.2.11
GetServiceAccessPoints()	See 16.3.2.12
GetReplicationRelationships()	See 16.3.2.13
GetReplicationRelationshipInstances()	See 16.3.2.14
AddReplicationEntity()	See 16.3.2.15
AddServiceAccessPoint()	See 16.3.2.16
AddShareSecret()	See 16.3.2.17
CreateRemoteReplicationCollection()	See 16.3.2.18
AddToRemoteReplicationCollection()	See 16.3.2.19
RemoveFromRemoteReplicationCollection()	See 16.3.2.20



Table 280 summarizes the extrinsic methods for examining the implemented capabilities (class `FileSystemReplicationServiceCapabilities`). The majority of these methods accept the `ReplicationType` as an input parameter. The supplied `ReplicationType` must be a supported replication type corresponding to the property `FileSystemReplicationServicesCapabilities.SupportedReplicationTypes`; otherwise the method returns "Not Supported" or throw a "Not Supported" exception.

**Table 280 - Extrinsic Method for Getting Supported Capabilities**

Method	Described in
<code>ConvertSyncTypeToReplicationType()</code>	See 16.3.3.1
<code>ConvertReplicationTypeToSyncType()</code>	See 16.3.3.2
<code>GetSupportedFeatures()</code>	See 16.3.3.3
<code>GetSupportedGroupFeatures()</code>	See 16.3.3.4
<code>GetSupportedCopyStates()</code>	See 16.3.3.5
<code>GetSupportedGroupCopyStates()</code>	See 16.3.3.6
<code>GetSupportedWaitForCopyStates()</code>	See 16.3.3.7
<code>GetSupportedConsistency()</code>	See 16.3.3.8
<code>GetSupportedOperations()</code>	See 16.3.3.9
<code>GetSupportedGroupOperations()</code>	See 16.3.3.10
<code>GetSupportedListOperations()</code>	See 16.3.3.11
<code>GetSupportedSettingsDefineStateOperations()</code>	See 16.3.3.12
<code>GetSupportedThinProvisioningFeatures()</code>	See 16.3.3.13
<code>GetSupportedMaximum()</code>	See 16.3.3.14
<code>GetDefaultConsistency()</code>	See 16.3.3.15
<code>GetDefaultGroupPersistency()</code>	See 16.3.3.16
<code>GetSupportedReplicationSettingData()</code>	See 16.3.3.17
<code>GetDefaultReplicationSettingData()</code>	See 16.3.3.18
<code>GetSupportedConnectionFeatures()</code>	See 16.3.3.19
<code>GetSupportedStorageCompressionFeatures()</code>	See 16.3.3.20

## 16.3.1 Group Management Methods

### 16.3.1.1 CreateGroup

```
uint32 ReplicationService.CreateGroup(
    [IN] string GroupName,
    [IN] CIM_LogicalElement REF Members[],
    [IN] boolean Persistent,
    [IN] boolean DeleteOnEmptyElement,
    [IN] boolean DeleteOnUnassociated,
    [IN] CIM_ServiceAccessPoint REF ServiceAccessPoint,
    [OUT] CIM_ReplicationGroup REF ReplicationGroup,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
    string ReplicationSettingData );
```

This Method allows a client to create a new replication group. Any required associations (such as HostedCollection) are created in addition to the instance of the group. The parameters are as follows:

- **GroupName:** If nameable, an end user relevant name for the group being created. If NULL or not nameable, then system assigns a name.
- **Members[]:** List of elements to add to the group -- order is maintained. If NULL, the group will be empty -- if empty groups are supported.
- **Persistent:** If false, the group, not the elements associated with the group, may be deleted at the completion of a copy operation. Use the intrinsic method `ModifyInstance` to change `Persistent` of a group.
- **DeleteOnEmptyElement:** If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty. If this parameter is not NULL, its value will be used to set the group's `DeleteOnEmptyElement` property. Use the intrinsic method `ModifyInstance` to change this property after the group is created.
- **DeleteOnUnassociated:** If true, the group will be deleted when the group is no longer associated with another group. This can happen if all synchronization associations to the individual elements of the group are dissolved. If this parameter is not NULL, its value will be used to set the group's `DeleteOnUnassociated` property. Use the intrinsic method `ModifyInstance` to change this property after the group is created.
- **ServiceAccessPoint:** Reference to access point information to allow the service to create a group on a remote system. If NULL, the group is created on the local system.
- **ReplicationGroup:** Reference to the created group.
- **ReplicationSettingData:** If supplied, it provides additional replication settings for the method. For example, to supply the "Description" for the created group.

This method returns the following additional values/statuses:

- If a groups are not nameable and a name is supplied, the method return 7 ("Groups are not nameable") or throws an appropriate exception.

#### 16.3.1.2 DeleteGroup

```
uint32 ReplicationService.DeleteGroup(
    [IN, Required] CIM_ReplicationGroup REF ReplicationGroup,
    [IN] CIM_ServiceAccessPoint REF ServiceAccessPoint,
    [IN] boolean RemoveElements,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" ) ]
    string ReplicationSettingData);
```

This method allows a client to delete a replication group. All associations to the deleted group are also removed as part of the action. The parameters are as follows:

- **ReplicationGroup:** Reference to a replication group that the client want s to delete.
- **ServiceAccessPoint:** Reference to access point information to allow the service to delete the group on a remote system. If null, the group is on the local system.
- **RemoveElements:** Delete the group even if it is not empty. If one or more elements in the group are in a replication relationship, `RemoveElements` has no effect.
- **ReplicationSettingData:** If supplied, it provides additional replication settings for the method. For example, what should happen `OnGroupOrListError`.

This method returns the following addition values/statuses:

- If an element in the group is in a replication association, the method return 7 ("One or more elements in a replication relationship") or throws an appropriate exception.

#### 16.3.1.3 AddMembers

```
uint32 ReplicationService.AddMembers(
    [IN] CIM_LogicalElement REF Members[],
    [IN, Required] CIM_ReplicationGroup REF ReplicationGroup,
    [IN] CIM_ServiceAccessPoint REF ServiceAccessPoint,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData);
```

This method allows a client to add members to an existing replication group. The parameters are as follows:

- Members[]: An array of strings containing object references to the new elements to add to the replication group. The new elements are added at the end of current members of the replication group. Duplicate members are not allowed.
- ReplicationGroup: A reference to an existing replication group.
- ServiceAccessPoint: Reference to access point information to allow the service to access the group on a remote system. If null, the group is on the local system.
- ReplicationSettingData: If supplied, it provides additional replication settings for the method. For example, what should happen OnGroupOrListError.

#### 16.3.1.4 RemoveMembers

```
uint32 ReplicationService.RemoveMembers(
    [IN] CIM_LogicalElement REF Members[],
    [IN] boolean DeleteOnEmptyElement,
    [IN, Required] CIM_ReplicationGroup REF ReplicationGroup,
    [IN] CIM_ServiceAccessPoint REF ServiceAccessPoint,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData );
```

This method allows a client to remove members from an existing replication group. If empty replication groups are not supported by the implementation, deleting all members will delete the group. The parameters are as follows:

- Members[]: An array of strings containing object references to the elements to remove from the replication group. Attempting to remove a member that is not in the replication group, returns an error.
- DeleteOnEmptyElement: If true and removal of the members causes the group to become empty, the group will be deleted. Note, if empty groups are not allowed, the group will be deleted automatically when the group becomes empty. If this parameter is not null, it overrides the group's property DeleteOnEmptyElement.
- ReplicationGroup: A reference to an existing replication group.
- ServiceAccessPoint: Reference to access point information to allow the service to access the group on a remote system. If null, the group is on the local system.
- ReplicationSettingData: If supplied, it provides additional replication settings for the method. For example, what should happen OnGroupOrListError.

This method returns the following additional values/statuses:

- Attempting to remove a group member that is in a replication association, returns 7 (“One or more element in a replication relationship”) or throws an appropriate exception.

### 16.3.2 Replication Management

#### 16.3.2.1 CreateElementReplica

```
uint32 ReplicationService.CreateElementReplica(
    [IN] string ElementName,
    [IN, Required] uint16 SyncType,
    [IN] uint16 Mode,
    [IN, Required] CIM_LogicalElement REF SourceElement,
    [IN] CIM_ServiceAccessPoint REF SourceAccessPoint,
    [IN, OUT] CIM_LogicalElement REF TargetElement,
    [IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
    [IN, EmbeddedInstance("CIM_ReplicationSettingData")]
        string ReplicationSettingData,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_Synchronized REF Synchronization,
    [IN] CIM_SettingData REF TargetSettingGoal,
    [IN] CIM_ResourcePool REF TargetPool,
    [IN] uint16 WaitForCopyState);
```

This method allows a client to create (or start a job to create) a new storage object which is a replica of the specified source storage object (SourceElement). The parameters are as follows:

- ElementName: A end user relevant name for the element being created. If null, then a system supplied name is used. The value will be stored in the 'ElementName' property for the created element.
- SyncType: Describes the type of copy that will be made. For example, Mirror, Snapshot, and Clone.
- Mode: Describes whether the target elements will be updated synchronously or asynchronously.
- SourceElement: The source storage object which may be a FileSystem or storage object.
- SourceAccessPoint: Reference to source access point information. If null, service does not need access information to access the source element.
- TargetElement:
  - As an input, refers to a target element to use. If a target element is not supplied, the implementation may locate or create a suitable target element. See 16.3.3.17: GetSupportedReplicationSettingData
  - As an output, refers to the created target storage element (i.e., the replica). If a job is created, the target element may not be available immediately.
- TargetAccessPoint: Reference to target access point information. If null, service does not need access information to access the target element.
- ReplicationSettingData: If provided, it overrides the default replication setting data for the given SyncType. If not provided, the implementation uses the default replication setting data.
- Job: If a Job is created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter (may be null if job is completed).
- Synchronization: Refers to the created association between the source and the target element. If a job is created, this parameter may be null, unless the association is actually formed.

- **TargetSettingGoal:** The definition for the FileSystemSetting to be maintained by the target storage object (the replica). If a target element is supplied, this parameter shall be null.
- **TargetPool:** The underlying storage for the target element (the replica) will be drawn from TargetPool if specified, otherwise the allocation is implementation specific. If a target element is supplied, this parameter shall be null.
- **WaitForCopyState:** Before returning, the method shall wait until this CopyState is reached. For example, CopyState of Initialized means associations have been established, but there is no data flow. CopyState of Synchronized indicates the replica is an exact copy of the source element. CopyState of UnSynchronized means copy operation is in progress (see Figure 276 for the CopyStates).

**Method Notes:**

- Creates a storage element of the same type as the source element.
- If the TargetElement, the TargetPool, or the TargetAccessPoint are not specified, the TargetElement is created on the system hosting the replication service, via the HostedService association. Additionally, when required, the created TargetElement will have the applicable association to the top level ComputerSystem. For example, if the TargetElement is a FileSystem, the created TargetElement will have a HostedFileSystem association to the top level computer system.
- Creates a FileSystemSynchronized association.
- Creates HostedFileSystem, ResidesOnExtent, and ElementSettingData associations to the newly created target element.
- May create BasedOn and ReplicaPoolForStorage associations.

Table 281 shows selected optional parameters that can interact.

**Table 281 - Selected CreateElementReplica optional parameters**

TargetElement	TargetSettingGoal	TargetPool	Comment
Null	Null	Null	Implementation locates/creates target element*
Supplied	Null	Null	
Null	Supplied	Null	Goal is used to locate/create target element*
Null	Supplied	Supplied	Goal is used to locate/create target element* in the supplied Pool
Null	Null	Supplied	Pool is used to locate/create target element* in Pool. Implementation determines the Goal

NOTE \* See capabilities (Table 299, "Target Element Suppliers") for whether implementation locates/creates target elements.

### 16.3.2.2 CreateGroupReplica

```
uint32 ReplicationService.CreateGroupReplica(
    [IN] string RelationshipName,
    [IN, Required] uint16 SyncType,
    [IN] uint16 Mode,
    [IN] CIM_ReplicationGroup REF SourceGroup,
    [IN] CIM_LogicalElement REF SourceElement,
    [IN] CIM_ServiceAccessPoint REF SourceAccessPoint,
    [IN, OUT] CIM_ReplicationGroup REF TargetGroup,
    [IN] uint64 TargetElementCount,
```

```
[IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
[IN] uint16 Consistency,
[IN, EmbeddedInstance("CIM_ReplicationSettingData")]
    string ReplicationSettingData,
[OUT] CIM_ConcreteJob REF Job,
[OUT] CIM_Synchronized REF Synchronization,
[IN] CIM_SettingData REF TargetSettingGoal,
[IN] CIM_ResourcePool REF TargetPool,
[IN] uint16 WaitForCopyState);
```

This method allows a client to create (or start a job to create) a new group of storage objects which are replicas of the specified source storage or a group of source storage objects (SourceElements). The parameters are as follows:

- **RelationshipName:** A user relevant name for the relationship between the source and target groups or between a source element and a target group (i.e., one-to-many). If null, the implementation assigns a name. If the individual target elements require an ElementName, a name would be constructed using RelationshipName (or ReplicationSettingData.ElementName) as prefix followed by "\_n" sequence number, where n is a number beginning with 1.

If the method is expected to create the target group, and the parameter ReplicationSettingData is supplied, the property ReplicationSettingData.ElementName may be used as the group name.

- **SyncType:** See CreateElementReplica's parameters (16.3.2.1)
- **Mode:** See CreateElementReplica's parameters (16.3.2.1)
- **SourceGroup:** A group of source storage objects. If this parameter is not supplied, SourceElement is required. Both SourceGroup and SourceElement shall not be supplied.
- **SourceElement:** The source storage object. If this parameter is not supplied, SourceGroup is required. Both SourceGroup and SourceElement shall not be supplied.
- **SourceAccessPoint:** Reference to source access point information. If null, service does not need access information to access the source elements/group.
- **TargetGroup:**

As an input, refers to a target group to use.

As an output, refers to the created target group (i.e., the replica group). If a job is created, the target group may not be available immediately. If TargetGroup is supplied, TargetElementCount shall be null.

- **TargetElementCount:** This parameter applies to one-source-to-many-target elements. If TargetGroup is supplied, this parameter shall be null.
- **TargetAccessPoint:** Reference to target access point information. If null, service does not need access information to access the target elements/group.
- **Consistency:** This parameter overrides the default group consistency. For example, "No Consistency", "Sequential Consistency".
- **ReplicationSettingData:** See CreateElementReplica's parameters (16.3.2.1)
- **Job:** See CreateElementReplica's parameters (16.3.2.1)

- Synchronization: Refers to the created association between the source element (or source replication group) and the target replication group. If a job is created, this parameter may be null, unless the association is actually formed.
- TargetSettingGoal: See CreateElementReplica's parameters (16.3.2.1)
- TargetPool: See CreateElementReplica's parameters (16.3.2.1)
- WaitForCopyState: See CreateElementReplica's parameters (16.3.2.1)

Method Notes:

- Creates storage elements of the same type as the source element(s)
- If the TargetGroup or the TargetAccessPoint are not specified, the TargetGroup is created on the system hosting the replication service, via the HostedService association.
- Creates FileSystemSynchronized and FileSystemGroupSynchronized associations.
- Creates HostedFileSystem, ResidesOnExtent, and ElementSettingData associations to the newly created target elements.
- May create BasedOn and ReplicaPoolForStorage associations.

Table 282 shows selected optional parameters that can interact

**Table 282 - Selected CreateGroupReplica optional parameters**

TargetGroup	TargetElementCount	TargetSettingGoal	TargetPool	Comment
Null	Null	Null	Null	Implementation locates/ creates target elements*
Supplied	Null	Null	Null	
Supplied	Supplied	Null	Null	An illegal combination.
Null	Supplied	Null	Null	Implementation locates/ creates target elements*
Null	Supplied	Supplied	Null	Goal is used to locate/create target elements
Null	Supplied	Supplied	Supplied	Goal is used to locate/create target elements* in the supplied Pool
Null	Null	Supplied	Null	Goal is used to locate/create target elements
Null	Null	Supplied	Supplied	Goal is used to locate/create target elements in the supplied Pool
Null	Null	Null	Supplied	Pool is used to locate/create target elements* in Pool. Implementation determines the Goal

NOTE \* See capabilities (Table 299, "Target Element Suppliers") for whether implementation locates/creates target elements

### 16.3.2.3 CreateListReplica

```
uint32 ReplicationService.CreateListReplica(
```

```
[IN] string ElementNames[],
[IN, Required] uint16 SyncType,
[IN] uint16 Mode,
[IN, Required] CIM_LogicalElement REF SourceElements[],
[IN] CIM_ServiceAccessPoint REF SourceAccessPoint,
[IN, OUT] CIM_LogicalElement REF TargetElements[],
[IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
[IN, EmbeddedInstance("CIM_ReplicationSettingData")]
    string ReplicationSettingData,
[OUT] CIM_ConcreteJob REF Job,
[OUT] CIM_Synchronized REF Synchronizations[],
[IN] CIM_SettingData REF TargetSettingGoal,
[IN] CIM_ResourcePool REF TargetPool,
[IN] uint16 WaitForCopyState);
```

This method allows a client to create (or start a job to create) new storage objects which are a replica of the corresponding specified source storage object (an element of the SourceElements). The parameters are as follows:

- **ElementNames:** An array of end user relevant names for the elements being created. If null, then a system supplied name is used. The value will be stored in the 'ElementName' property for the created element. The first element of the array ElementNames is assigned to the first replica, the second element to the second replica and so on. If there are more SourceElements entries than ElementNames, the system supplied name is used.
- **SyncType:** See CreateElementReplica's parameters (16.3.2.1)
- **Mode:** See CreateElementReplica's parameters (16.3.2.1)
- **SourceElements:** An array of source storage objects. All the source elements shall be of the same type -- for example, all FileSystems.
- **SourceAccessPoint:** Reference to source access point information. If null, service does not need access information to access the source element. The same SourceAccessPoint applies to all SourceElements entries.
- **TargetElements:**

As an input, refers to an array of target elements to use. If specified, the elements will match one to one with SourceElements[]. If a target elements are not supplied, the implementation may locate or create a suitable target elements. See 16.3.3.17: GetSupportedReplicationSettingData.

As an output, refers to the created target storage elements (i.e., the replicas). If a job is created, the target elements may not be available immediately

- **TargetAccessPoint:** Reference to target access point information. If null, service does not need access information to access the target element. The same TargetAccessPoint applies to all TargetElements entries.
- **ReplicationSettingData:** If provided, it overrides the default replication setting data for the given SyncType. If not provided, the implementation uses the default replication setting data. The same ReplicationSettingData applies to SourceElements entries.
- **Job:** If a Job is created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter (may be null if job is completed).
- **Synchronizations:** Refers to an array of created associations between the source and the target elements. If a job is created, this parameter may be null, unless the associations are actually formed.



- **TargetSettingGoal:** The definition for the FileSystemSetting to be maintained by the target storage object (the replica). If a target element is supplied, this parameter shall be null. The same TargetSettingGoal applies to all TargetElements entries.
- **TargetPool:** The underlying storage for the target element (the replica) will be drawn from TargetPool if specified, otherwise the allocation is implementation specific. If a target element is supplied, this parameter shall be null. The same TargetPool applies to all TargetElement entries.
- **WaitForCopyState:** Before returning, the method shall wait until this CopyState is reached for all Synchronizations. For example, CopyState of Initialized means associations have been established, but there is no data flow. CopyState of Synchronized indicates the replicas are an exact copy of the corresponding source element. CopyState of UnSynchronized means copy operation is in progress (see Table 276 -: CopyStatus Values for the CopyStates).

**Method Notes:**

- Creates a storage elements of the same type as the source elements.
- If the TargetElements, the TargetPool, or the TargetAccessPoint are not specified, the TargetElements are created on the system hosting the replication service, via the HostedService association. Additionally, when required, the created TargetElements will have the applicable associations to the top level ComputerSystem. For example, if the TargetElements are FileSystem, the created TargetElements will have HostedFileSystem associations to the top level computer system.
- Creates the FileSystemSynchronized associations.
- Creates HostedFileSystem, ResidesOnExtent, and ElementSettingData associations to the newly created target elements.
- May create BasedOn and ReplicaPoolForStorage associations.

Table 283 shows selected optional parameters that can interact.

**Table 283 - Selected CreateListReplica optional parameters**

TargetElements	TargetSettingGoal	TargetPool	Comment
Null	Null	Null	Implementation locates/creates target elements*
Supplied	Null	Null	
Null	Supplied	Null	Goal is used to locate/create target elements*
Null	Supplied	Supplied	Goal is used to locate/create target elements* in the supplied Pool
Null	Null	Supplied	Pool is used to locate/create target elements* in Pool. Implementation determines the Goal

NOTE \* See capabilities (Table 299, "Target Element Suppliers") for whether implementation locates/creates target elements

#### 16.3.2.4 CreateGroupReplicaFromElements

```
uint32 ReplicationService.CreateGroupReplicaFromElements(
    [IN] string RelationshipName,
    [IN, Required] uint16 SyncType,
    [IN] uint16 Mode,
    [IN,OUT] CIM_ReplicationGroup REF SourceGroup,
    [IN] CIM_LogicalElement REF SourceElements[],
    [IN, OUT] string SourceGroupName,
```

```
[IN] CIM_ServiceAccessPoint REF SourceAccessPoint,
[IN, OUT] CIM_ReplicationGroup REF TargetGroup,
[IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
[IN] uint16 Consistency,
[IN, EmbeddedInstance("CIM_ReplicationSettingData")]
    string ReplicationSettingData,
[OUT] CIM_ConcreteJob REF Job,
[OUT] CIM_Synchronized REF Synchronization,
[IN] CIM_SettingData REF TargetSettingGoal,
[IN] CIM_ResourcePool REF TargetPool,
[IN] uint16 WaitForCopyState);
```

This method allows a client to create (or start a job to create) a new group of storage objects which are replicas of the specified source storage objects (SourceElements). This method combines the functionality of CreateGroup and CreateGroupReplica in that the methods accepts a list of source elements and creates the source group, and the target group, if not supplied.

The parameter SourceGroupName corresponds to the parameter GroupName as defined in the CreateGroup method.

For the explanation of the parameters, see the methods CreateGroup (16.3.1.1) and CreateGroupReplica (16.3.2.2).

#### 16.3.2.5 CreateSynchronizationAspect

```
uint32 ReplicationService.CreateSynchronizationAspect(
    [IN] string ElementName,
    [IN, Required] uint16 SyncType,
    [IN] uint16 Mode,
    [IN] CIM_ReplicationGroup REF SourceGroup,
    [IN] CIM_LogicalElement REF SourceElement,
    [IN] CIM_ServiceAccessPoint REF SourceAccessPoint,
    [IN] uint16 Consistency,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_SettingsDefineState REF SettingsState );
```

This method allows a client to create (or start a job to create) new instances of SynchronizationAspect that are associated to the source element (or a group of source elements) via the SettingsDefineState associations. This representation may be of a form of pointers or a series of checkpoints that keep track of the source element data for the created point-in-time.

This method does not include a target element, however, a target element can be added subsequently using the ModifySettingsDefineState method.

The method creates individual associations between the source elements and the instances of SynchronizationAspect.

The parameters are as follows:

- ElementName: A end user relevant name. If null, then a system supplied default name can be used. The value will be stored in the ElementName property of the created SynchronizationAspect.
- SyncType: See CreateElementReplica's parameters (16.3.2.1).

- Mode: See CreateElementReplica's parameters (16.3.2.1).
- SourceGroup: See CreateGroupReplica's parameters in (16.3.2.2)
- SourceElement: See CreateGroupReplica's parameters (16.3.2.2)
- SourceAccessPoint: Reference to source access point information. If null, service does not need access information to access the source element/group.
- Consistency: See CreateGroupReplica's parameters (16.3.2.2)
- ReplicationSettingData: See CreateElementReplica's parameters (16.3.2.1).
- Job: See CreateElementReplica's parameters (16.3.2.1).
- SettingsState: Refers to the created association between the source element or group and the instance of the SynchronizationAspect. If a job is created, this parameter may be null, unless the association is actually formed.

Method Notes:

- May create an instance of SynchronizationAspect if an appropriate one does not exist already.
- May create ReplicaPoolForStorage associations.

#### 16.3.2.6 ModifyReplicationSynchronization

```
uint32 ReplicationService.ModifyReplicaSynchronization(
    [IN, Required] uint16 Operation,
    [IN, Required] CIM_Synchronized REF Synchronization,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
    string ReplicationSettingData,
    [IN] CIM_Synchronized REF SyncPair[],
    [OUT] CIM_ConcreteJob REF Job,
    [IN] boolean Force,
    [OUT] CIM_SettingsDefineState REF SettingsState,
    [IN] uint16 WaitForCopyState);
```

This method allows a client to modify (or start a job to modify) the synchronization association between two storage objects or replication groups. The parameters are as follows:

- Operation: This parameter describes the type of modification to be made to the replica and/or to the related associations, for example, Split.
- Synchronization: The reference to the replication association describing the elements/groups relationship that is to be modified.
- ReplicationSettingData: See CreateElementReplica's parameters (16.3.2.1).
- SyncPair[]: This parameter applies to AddSyncPair/RemoveSyncPair Operations. It allows a client to form a Synchronized association between source and target elements and then add the association to existing source and target groups. Alternatively, a client can remove a Synchronized association from source and target groups.
- Job: See CreateElementReplica's parameters (16.3.2.1).
- SettingsState: Reference to the association between the source or group element and an instance of SynchronizationAspect. This parameters applies to operations such as Dissolve, which dissolves the

Synchronized relationship, but causes the SettingsDefineState association to be created. Depending on the implementation, Deactivate may also return a SettingsState.

- Force: Some operations may cause an inconsistency among the target elements. If true, the client is not warned and the operation is performed if possible.
- WaitForCopyState: See CreateElementReplica's parameters (16.3.2.1)

#### 16.3.2.7 ModifyListSynchronization

```
uint32 ReplicationService.ModifyListSynchronization(
    [IN, Required] uint16 Operation,
    [IN, Required] CIM_Synchronized REF Synchronization[],
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData,
    [OUT] CIM_ConcreteJob REF Job,
    [IN] boolean Force,
    [IN] uint16 WaitForCopyState);
```

This method allows a client to modify (or start a job to modify) a list of synchronization associations between two storage objects or replication groups. The parameters are as follows:

- Operation: This parameter describes the type of modification to be made to the replica and/or to the related associations, for example, Split.
- Synchronization: An array of references to the replication association describing the elements/groups relationship that is to be modified. All elements of the this array shall of the same concrete class, i.e., FileSystemSynchronized or FileSystemGroupSynchronized, and shall have the same SyncType, the same Mode, and the Operation must be valid for the ReplicationType -- SyncType, Mode, Local/Remote.
- ReplicationSettingData: See CreateElementReplica's parameters (16.3.2.1).
- Job: See CreateElementReplica's parameters (16.3.2.1).
- Force: Some operations may cause an inconsistency among the target elements. If true, the client is not warned and the operation is performed if possible.
- WaitForCopyState: See CreateElementReplica's parameters (16.3.2.1). All the supplied synchronization associations must reach at least the specified CopyState before the method returns.

#### 16.3.2.8 ModifySettingsDefineState

```
uint32 ReplicationService.ModifySettingsDefineState(
    [IN, Required] uint16 Operation,
    [IN, Required] CIM_SettingsDefineState REF SettingsState,
    [IN, OUT] CIM_LogicalElement REF TargetElement,
    [IN, OUT] CIM_ReplicationGroup REF TargetGroup,
    [IN] uint64 TargetElementCount,
    [IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
    [OUT] CIM_Synchronized REF Synchronization,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData,
    [OUT] CIM_ConcreteJob REF Job,
    [IN] CIM_SettingData REF TargetSettingGoal,
    [IN] CIM_ResourcePool REF TargetPool,
    [IN] uint16 WaitForCopyState);
```

This method allows a client to modify (or start a job to modify) the SettingsDefineState association between the storage objects and SynchronizationAspect. The modification could range from introducing the target elements, which creates new FileSystemSynchronized associations, to dissolving the SettingsDefineState associations all together.

With the Copy To Target operation, the supplied SettingsState is deleted since an “active” Synchronization is created to associate the source and the target elements (or groups).

The parameters are:

- Operation: This parameter describes the type of modification to be made to the related associations, for example, Copy To Target, which initiates the copy operation from the point-in-time view to the supplied targets.
- SettingsState: Refers to the associations between the source elements and the SynchronizationAspect instances. If an associated source element is part of a consistency group, all members of the group shall be paired with the appropriate target elements.
- TargetElement: If TargetElement is supplied, TargetGroup and TargetCount shall be null.
  - As an input, if the point-in-time has only one source element, this parameter supplies the target element.
  - As an output, refers to the created target storage element (i.e., the replica). If a job is created, the target element may not be available immediately.
- TargetGroup: If TargetGroup is supplied, TargetElement and TargetElementCount shall be null.
  - As an input, refers to a target group to use. If the source has only one element, the presence of a group creates a one-to-many association between the source and the target elements. If TargetGroup is supplied, TargetElement and TargetCount shall be null."
  - As an output, refers to the created target group (i.e., the replica group). If a job is created, the target group may not be available immediately.
- TargetElementCount: This parameter applies to one-source-to-many-target-elements. It is possible to create multiple copies of a source element. If TargetCount is supplied, TargetElement and TargetGroup shall be null.
- TargetAccessPoint: Reference to target access point information. If null, service does not need access information to access the target elements/group.
- Synchronization: The reference to the replication association describing the elements/groups relationship.
- ReplicationSettingData: See CreateElementReplica's parameters (16.3.2.1).
- Job: See CreateElementReplica's parameters (16.3.2.1).
- TargetSettingGoal: See CreateElementReplica's parameters (16.3.2.1).
- TargetPool: See CreateElementReplica's parameters (16.3.2.1).
- WaitForCopyState: See CreateElementReplica's parameters (16.3.2.1).

#### 16.3.2.9 ModifyListSettingsDefineState

```
uint32 ReplicationService.ModifyListSettingsDefineState(
    [IN, Required] uint16 Operation,
    [IN, Required] CIM_SettingsDefineState REF SettingsStates[],
    [IN, OUT] CIM_LogicalElement REF TargetElements[],
    [IN, OUT] CIM_ReplicationGroup REF TargetGroups[],
    [IN] uint64 TargetElementCount,
```

```
[IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
[OUT] CIM_Synchronized REF Synchronizations[],
[IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
    string ReplicationSettingData,
[OUT] CIM_ConcreteJob REF Job,
[IN] CIM_SettingData REF TargetSettingGoal,
[IN] CIM_ResourcePool REF TargetPool,
[IN] uint16 WaitForCopyState);
```

This method is similar to `ReplicationService.ModifySettingsDefineState` (16.3.2.8), except that it accepts a list of `SettingsDefineState` associations.

#### 16.3.2.10 GetAvailableTargetElements

```
uint32 ReplicationService.GetAvailableTargetElements(
    [IN, Required] CIM_LogicalElement REF SourceElement,
    [IN, Required] uint16 SyncType,
    [IN, Required] uint16 Mode,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData,
    [IN] CIM_ComputerSystem REF TargetComputerSystem,
    [IN] CIM_ServiceAccessPoint REF TargetAccessPoint,
    [IN] CIM_SettingData REF TargetSettingGoal,
    [IN] CIM_ResourcePool REF TargetPools[],
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_LogicalElement REF Candidates[] );
```

This method allows a client to get (or start a job to get) all of the candidate target elements for the supplied source element. If a job is started, once the job completes, examine the `AffectedJobElement` associations for candidate targets. The parameters are:

- **SourceElement:** The source storage object which may be a `FileSystem` or storage object.
- **SyncType:** See `CreateElementReplica`'s parameters (16.3.2.1).
- **Mode:** See `CreateElementReplica`'s parameters (16.3.2.1).
- **ReplicationSettingData:** See `CreateElementReplica`'s parameters (16.3.2.1). The parameter is useful for requesting a specific combination of thinly and fully provisioned elements.
- **TargetComputerSystem:** Reference to target computer system. If this parameter and `TargetAccessPoint` are null, only local targets are returned.
- **TargetAccessPoint:** Reference to target access point information. If this parameter and `TargetComputerSystem` are null, only local targets are returned.
- **TargetSettingGoal:** Desired target `StorageSetting`. If null, settings of the source elements shall be used.
- **TargetPools[]:** The storage pools for the target elements. If null, all storage pools (on the given systems) are examined.
- **Job:** See `CreateElementReplica`'s parameters (16.3.2.1).
- **Candidates[]:** The list of the candidate target elements found.

#### 16.3.2.11 GetPeerSystems

```
uint32 ReplicationService.GetPeerSystems(
```

```
[IN] uint16 Options,
[OUT] CIM_ConcreteJob REF Job,
[OUT] CIM_ComputerSystem REF Systems[] );
```

This method allows a client to get (or start a job to get) all of the peer systems. A peer system is a system that is known and visible to the FileSystem Replication Service. Peer systems are discovered through discovery services and/or implementation specific services. If a job is started, once the job completes, examine the AffectedJobElement associations for the peer systems. The parameters are:

- Options: This parameter specifies whether to return all known peer systems or only the systems that are currently reachable. If null, all known systems are returned, whether they are currently reachable or not.
- Job: See CreateElementReplica's parameters (16.3.2.1).
- Systems[]: The list of peer computer systems.

#### 16.3.2.12 GetServiceAccessPoints

```
uint32 ReplicationService.GetServiceAccessPoints(
    [IN] CIM_ComputerSystem REF System,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_ServiceAccessPoint REF ServiceAccessPoints[] );
```

This method allows a client to get (or start a job to get) ServiceAccessPoints associated with a peer system. If a job is started, once the job completes, examine the AffectedJobElement associations for the peer system's ServiceAccessPoints. The parameters are as follows:

- System: A reference to the computer system.
- Job: See CreateElementReplica's parameters (16.3.2.1).
- ServiceAccessPoints[]: An array of references to ServiceAccessPoints associated with the supplied system.

#### 16.3.2.13 GetReplicationRelationships

```
uint32 ReplicationService.GetReplicationRelationships(
    [IN] uint16 Type,
    [IN] uint16 SyncType,
    [IN] uint16 Mode,
    [IN] uint16 Locality,
    [IN] uint16 CopyState,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_Synchronized REF Synchronizations[] );
```

This method allows a client to get (or start a job to get) all of the synchronization relationships known to the processing replication service. If a job is started, once the job completes, examine the AffectedJobElement associations for the synchronization relationships. The parameters are as follows:

- Type: The type of synchronization relationships, for example, FileSystemSynchronized or FileSystemGroupSynchronized. If this parameter is not supplied, all such relationships are retrieved.
- SyncType: See CreateElementReplica's parameters (16.3.2.1). If this parameter is not supplied, all SyncTypes are retrieved.
- Mode: See CreateElementReplica's parameters (16.3.2.1). If this parameter is not supplied, all Modes are retrieved.
- Locality: Describes the desired locality. If this parameter is not supplied, all replication relationships are retrieved, regardless of the locality of elements. Choices are: Local only -- Source and target elements are

contained in the same system; and Remote only -- Source and target elements are contained in two different systems.

- CopyState: Only retrieve synchronization relationships that currently this CopyState (see Table 276). If this parameter is not supplied, relationships are retrieved regardless of their current CopyState.
- Job: See CreateElementReplica's parameters (16.3.2.1).
- Synchronizations[]: An array of elements found.

#### 16.3.2.14 GetReplicationRelationshipInstances

```
uint32 ReplicationService.GetReplicationRelationshipInstances(
    [IN] uint16 Type,
    [IN] uint16 SyncType,
    [IN] uint16 Mode,
    [IN] uint16 Locality,
    [IN] uint16 CopyState,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT, EmbeddedInstance("CIM_Synchronized")]
    string Synchronizations[] );
```

This method allows a client to get (or start a job to get) all of the synchronization relationship instances known to the processing replication service. If a job is started, once the job completes, examine the AffectedJobElement associations for the synchronization relationships.

The output parameter Synchronizations is an array of embedded instances. For the explanation of the remaining parameters, see the method ReplicationService.GetReplicationRelationships (16.3.2.13).

#### 16.3.2.15 AddReplicationEntity

```
uint32 ReplicationService.AddReplicationEntity(
    [Required, IN, EmbeddedInstance("CIM_ReplicationEntity")]
    string ReplicationEntity,
    [IN] boolean Persistent,
    [IN] string InstanceNamespace,
    [OUT] CIM_ReplicationEntity REF ReplicationEntityPath);
```

This method allows a client to introduce a new instance of ReplicationEntity in the specified Namespace. The parameters are:

- ReplicationEntity: A required parameter containing the information for the ReplicationEntity.
- Persistent: If true, the instance must persist across a Management Server reboot. If null, the value will be based on the default value of the class in the MOF. Use the intrinsic method ModifyInstance to change the Persistency value.
- InstanceNamespace: Namespace of created instance. If null, created instance will be in the same namespace as the service. Namespace must already exist.
- ReplicationEntityPath: A reference to the created instance.

#### 16.3.2.16 AddServiceAccessPoint

```
uint32 ReplicationService.AddServiceAccessPoint(
    [Required, IN, EmbeddedInstance("CIM_ServiceAccessPoint")]
    string ServiceAccessPoint,
    [IN] string InstanceNamespace,
```



```
[OUT] CIM_ServiceAccessPoint REF ServiceAccessPointPath);
```

This method allows a client to introduce a new instance of ServiceAccessPoint in the specified Namespace. The parameters are:

- ServiceAccessPoint: A required parameter containing the information for the ServiceAccessPoint, or a subclass of the class ServiceAccessPoint, for example, a RemoteServiceAccessPoint.
- InstanceNamespace: Namespace of created instance. If null, created instance will be in the same namespace as the service. Namespace must already exist.
- ServiceAccessPointPath: A reference to the created instance.

#### 16.3.2.17 AddShareSecret

```
uint32 ReplicationService.AddSharedSecret(
    [Required, IN, EmbeddedInstance("CIM_SharedSecret")]
    string SharedSecret,
    [IN] CIM_ServiceAccessPoint REF ServiceAccessPoint,
    [IN] string InstanceNamespace,
    [OUT] CIM_SharedSecret REF SharedSecretPath);
```

This method allows a client to introduce a new instance of SharedSecret in the specified Namespace and optionally associate it to an instance of a ServiceAccessPoint. The parameters are:

- SharedSecret: A required parameter containing the information for the SharedSecret.
- ServiceAccessPoint: Associate created instance to this ServiceAccessPoint. If null, no such association is established.
- InstanceNamespace: Namespace of created instance. If null, created instance will be in the same namespace as the service. Namespace must already exist.
- SharedSecretPath: A reference to the created instance.

#### 16.3.2.18 CreateRemoteReplicationCollection

```
uint32 ReplicationService.CreateRemoteReplicationCollection(
    [IN] string ElementName,
    [IN] CIM_ServiceAccessPoint REF LocalAccessPoints[],
    [IN] CIM_ServiceAccessPoint REF RemoteAccessPoints[],
    [IN] CIM_ComputerSystem REF RemoteComputerSystem,
    [IN] boolean Active,
    [IN] boolean DeleteOnUnassociated,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_ConnectivityCollection REF ConnectivityCollection );
```

A method to create (or start a job to create) a new instance of RemoteReplicationCollection, and optionally supply the remote system and the paths (i.e., i.e., ProtocolEndpoints) that are used to perform replication operations to/from the remote system. The parameters are:

- ElementName: A end user relevant name for the element being created. If NULL, then a system supplied default name will be used. The value will be stored in the 'ElementName' property for the created element.
- LocalAccessPoints: An array of references to local ServiceAccessPoints (for example, ProtocolEndpoints) that allow communication to the remote system.
- RemoteAccessPoints: An array of references to remote ServiceAccessPoints (for example, ProtocolEndpoints) that allow communication to the remote system.

- RemoteComputerSystem: A reference to the remote system.
- Active: If true, the instance of RemoteReplicationCollection will be enabled and allows replication operations to to the remote system. Use the intrinsic method ModifyInstance to change this property after the RemoteReplicationCollection is created.
- DeleteOnUnAssociated: If true, the instance of RemoteReplicationCollection will be deleted when it is no longer associated to a ServiceAccessPoint. Use the intrinsic method ModifyInstance to change this property after the RemoteReplicationCollection is created.
- Job: Reference to the job (may be NULL if job is completed) doing the work.
- ConnectivityCollection: Reference to the created instance of RemoteReplicationCollection.

#### 16.3.2.19 AddToRemoteReplicationCollection

```
uint32 ReplicationService.AddToRemoteReplicationCollection(
    [IN] CIM_ServiceAccessPoint REF LocalAccessPoints[],
    [IN] CIM_ServiceAccessPoint REF RemoteAccessPoints[],
    [IN] CIM_ComputerSystem REF RemoteComputerSystem,
    [OUT] CIM_ConcreteJob REF Job,
    [Required, IN] CIM_ConnectivityCollection REF ConnectivityCollection );
```

A method to add (or start a job to add) additional service access points (i.e., ProtocolEndpoints) and/or remote systems associations to an existing instance of RemoteReplicationCollection.

Generally, both AccessPoints and RemoteComputerSystem parameters are supplied to establish the access points to a remote ComputerSystem; however, if parameter AccessPoints is NULL, then only the RemoteComputerSystem is added for the existing AccessPoints associated to the RemoteReplicationCollection. If RemoteComputerSystem is NULL, then only AccessPoints are added for the existing remote ComputerSystems known to the RemoteReplicationCollection.

#### 16.3.2.20 RemoveFromRemoteReplicationCollection

```
uint32 ReplicationService.RemoveFromRemoteReplicationCollection(
    [IN] CIM_ServiceAccessPoint REF LocalAccessPoints[],
    [IN] CIM_ServiceAccessPoint REF RemoteAccessPoints[],
    [OUT] CIM_ConcreteJob REF Job,
    [Required, IN] CIM_ConnectivityCollection REF ConnectivityCollection );
```

A method to remove (or start a job to remove) service access points (i.e., ProtocolEndpoints) and/or remote systems associations from an existing instance of RemoteReplicationCollection.

Generally, both AccessPoints and RemoteComputerSystem parameters are supplied to remove the access points to a remote ComputerSystem; however, if parameter AccessPoints is NULL, then only the remote ComputerSystem is removed for the existing AccessPoints associated to the RemoteReplicationCollection. If ComputerSystem is NULL, then only AccessPoints are removed from the existing remote ComputerSystems known to the RemoteReplicationCollection. See the method CreateRemoteReplicationCollection for description of the parameters.

### 16.3.3 Capabilities Method

#### 16.3.3.1 ConvertSyncTypeToReplicationType

```
uint32 FileSystemReplicationServiceCapabilities.ConvertSyncTypeToReplicationType(
    [IN] uint16 SyncType,
    [IN] uint16 Mode,
    [IN] uint16 LocalOrRemote,
    [OUT] uint16 SupportedReplicationTypes );
```

The majority of the methods in this class accept `ReplicationType` which represents a combination of `SyncType`, `Mode`, and `Local/Remote`. This method accepts the supplied information and returns the corresponding `ReplicationType`, which can be passed to other methods to get the additional capabilities.

Table 284, Table 285, Table 286 and Table 287 show the values for the `CovertSyncTypeToReplicationType` parameters. These values also appear in the value maps in the appropriate MOF files.

**Table 284 - SyncTypes**

SyncType	Value
Mirror	6
Snapshot	7
Clone	8

**Table 285 - Mode**

Mode	Value
Synchronous	2
Asynchronous	2

**Table 286 - Locality**

Locality	Value
Local	2
Remote	3

**Table 287 - ReplicationTypes**

ReplicationType	Value
Synchronous Mirror Local	2
Asynchronous Mirror Local	3
Synchronous Mirror Remote	4
Asynchronous Mirror Remote	5
Synchronous Snapshot Local	6
Asynchronous Snapshot Local	7
Synchronous Snapshot Remote	8
Asynchronous Snapshot Remote	9
Synchronous Clone Local	10
Asynchronous Clone Local	11
Synchronous Clone Remote	12
Asynchronous Clone Remote	13

### 16.3.3.2 ConvertReplicationTypeToSyncType

```
uint32 FileSystemReplicationServiceCapabilities.ConvertReplicationTypeToSyncType(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SyncType,
    [OUT] uint16 Mode,
    [OUT] uint16 LocalOrRemote );
```

This method does the opposite of the method ConvertSyncTypeToReplicationType. This method translates ReplicationType to the corresponding SyncType, Mode, and Local/Remote.

### 16.3.3.3 GetSupportedFeatures

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedFeatures(
    [IN] uint16 ReplicationType,
    [OUT] uint16 Features[] );
```

For a given ReplicationType, this method returns the supported features, as listed in Table 288

**Table 288 - Features**

Feature	Description
"Replication Groups"	Elements in a replication group are supported in a replication operation.
"Multi-hop element replication"	A target element can also act as the source for another copy operation.
"Each hop must have same SyncType"	In a multi-hop replication, the new hop must have the same SyncType as the previous hop.
"Multi-hop requires advance notice"	The service needs to know when multi-hopping is intended to allow the service to do the appropriate set up. The parameter ReplicationSettingData specifies the number of hops intended.
"Requires full discovery of target ComputerSystem"	Provider requires the remote ComputerSystems to be discovered. The absence of this capability indicates the service supports undiscovered resources.
"Service suspends source I/O when necessary"	Provider is able to suspend I/O to source elements before splitting the target elements. Otherwise, the client needs to quiesce the application before issuing the split command.
"Targets allocated from Any storage pool"	Specialized storage pools are not required for the target elements, as long as the pool is not reserved for special activities.
"Targets allocated from Shared storage pool"	Targets are allocated from storage pools reserved for Replication Services.
"Targets allocated from Exclusive storage pool"	Targets are allocated from exclusive storage pools.
"Targets allocated from Multiple storage pools"	Targets are allocated from multiple specialized, exclusive pools.
"Targets require reserved elements"	The target elements must have a specific Usage value. For example, reserved for "Local Replica Target" (mirror), reserved for "Delta Replica Target" (Snapshot), etc.

**Table 288 - Features**

Feature	Description
"Target is associated to SynchronizationAspect"	The target element is associated to SynchronizationAspect via SettingsDefineState. SynchronizationAspect contains the point-in-time timestamp and the source element reference used to copy to the target element.
"Source is associated to SynchronizationAspect"	The source element is associated to SynchronizationAspect via the SettingsDefineState association. SynchronizationAspect contains the point-in-time information of the source data.
"Error recovery from Broken state Automatic"	For example, if the connection between the source and target elements is broken (CopyState = Broken or Partitioned), once the connection is restored, the copy operation continues automatically. If the error recovery is not automatic, it requires manual intervention to restart the copy operation. Use ModifyReplicaSynchronization, with Operation set to Resume.
"Target must remain associated to source"	A dependent target element must remain associated to source element at all times.
"Remote resource requires remote CIMOM"	Client is required to interact with two providers: the provider controlling the source element and the provider controlling the target element.
"Synchronized clone target detaches automatically"	The clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation.
"Restore operation requires fracture"	The "Restore from Replica" operation requires the synchronization relationship to be fractured after restore is completed -- indicated in the property Synchronized.ProgressStatus - "Requires fracture".
"Resync operation requires activate"	For the copy operation to continue, the synchronization relationship must be activated -- indicated in the property Synchronized.ProgressStatus - "Requires activate".
"Copy operation requires offline source"	Instrumentation requires the source element to be offline (not-ready) to ensure data does not change before starting the copy operation.
"Adjustable CopyPriority"	Priority of copy operation versus the host I/O can be adjusted.

#### 16.3.3.4 GetSupportedGroupFeatures

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedGroupFeatures(
    [IN] uint16 ReplicationType,
    [OUT] uint16 GroupFeatures[] );
```

For a given ReplicationType, this method returns the supported replication group features, as listed in Table 289.

**Table 289 - Group Features**

Group Features	Description
"One-to-many replication"	One source element can be copied to multiple target elements in a group.
"Many-to-many replication"	One or more elements in the source group and one or more elements in the target group.
"Consistency enabled for all groups"	By default, all groups are <i>Consistent</i>
"Empty replication groups allowed"	It is possible to have a replication group with no members; otherwise, an empty group gets deleted automatically
"Source group must have more than one element"	One members replication groups are not supported.
"Composite Groups"	A replication group can have members from different ComputerSystems.
"Multi-hop group replication"	A target replication group can also act as a source for another copy operation.
"Each hop must have same SyncType"	The SyncType of each hop must be the same, e.g., mirror, snapshot, clone.
"Group can only have one single relationship active"	At any given time, only one relationship in the source group can be active.
"Source element can be removed from group"	A source element can be removed even when the group is associated with another replication group.
"Target element can be removed from group"	A target element can be removed even when the group is associated with another replication group.
"Group can persist"	The replication group can persist across the Provider reboot (group is not temporary).
"Group is nameable"	A user friendly name can be given to a replication group (ElementName)
"Supports target element count"	It is possible to supply one source element and request more than one target element copies.
"Synchronized clone target detaches automatically"	The clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation
"Restore operation requires fracture"	The "Restore from Replica" operation requires the synchronization relationship to be fractured after restore is completed -- indicated in the property Synchronized.ProgressStatus - "Requires fracture".
"Resync operation requires activate"	For the copy operation to continue, the synchronization relationship must be activated -- indicated in the property Synchronized.ProgressStatus - "Requires activate".
"Copy operation requires offline source"	Instrumentation requires the source element to be offline (not-ready) to ensure data does not change before starting the copy operation.

### 16.3.3.5 GetSupportedCopyStates

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedCopyStates(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedCopyStates[],
    [OUT] boolean HostAccessible[] );
```

For a given ReplicationType, this method returns the supported CopyStates (see Table 276) and a parallel array to indicate whether for a given CopyState the target element is host accessible or not (true or false).

### 16.3.3.6 GetSupportedGroupCopyStates

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedGroupCopyStates(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedCopyStates[] );
```

For a given ReplicationType, this method returns the supported replication group CopyStates (seeTable 276).

### 16.3.3.7 GetSupportedWaitForCopyStates

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedWaitForCopyStates(
    [IN] uint16 ReplicationType,
    [IN] uint16 MethodName,
    [OUT] uint16 SupportedCopyStates[] );
```

This method, for a given ReplicationType and method, returns the supported CopyStates that can be specified in the method's WaitForCopyState parameter.

### 16.3.3.8 GetSupportedConsistency

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedConsistency(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedConsistency[] );
```

For a given ReplicationType, this method returns the supported Consistency, as listed in Table 290.

**Table 290 - Consistency**

Consistency	Description
"Sequentially Consistent"	Provider guarantees ordered write consistency.

### 16.3.3.9 GetSupportedOperations

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedOperations(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported Operations on a FileSystemSynchronized association that can be supplied to the ModifyReplicaSynchronization method. Table 291 shows the possible Operations that an implementation may support.

**Table 291 - Operations**

Operation	Description	Special Consideration
"Abort"	Abort the copy operation if it is possible.	
"Activate Consistency"	Enable consistency.	
"Activate"	Activate an "Inactive" or "Prepared" FileSystemSynchronized association.	
"AddSyncPair"	Add source and target elements of a FileSystemSynchronized association to the source and target replication groups. The SyncType of the associations must be the same.	
"Deactivate Consistency"	Disable consistency	
"Deactivate"	Stop the copy operation. Writes to source element are allowed.	Snapshot: Writes to target element after point-in-time is created are lost (pointers removed)

**Table 291 - Operations**

Operation	Description	Special Consideration
"Detach"	Remove the association between the source and target elements. Detach does not delete the target element.	
"Dissolve"	Dissolve the synchronization association between two storage objects, however, the target element continues to exist.	Snapshot: This operation also creates a SettingsDefineState association between the source element and an instance of SynchronizationAspect if the ReplicationType supports it.
"Failover"	Enable the read and write operations from the host to the target element. This operation useful for situations when the source element is unavailable.	
"Failback"	Switch the read/write activities from the host back to source element. Update source element from target element with writes to target during the failover period.	
"Fracture"	Separate the target element from the source element.	
"RemoveSyncPair"	Remove the elements associated via the FileSystemSynchronized association from the source and the target groups.	
"Resync Replica"	Resynchronize a fractured target element. Or, from a Broken or Aborted relationship.	To continue from the <i>Broken</i> state, the problem should be corrected first before resyncing the replica. Also, to continue from the <i>Aborted</i> state.
"Restore from Replica"	Copy target element to the source element	To ensure integrity of data, restoring to a source element which is the source of multiple copy operations, the implementation may impose additional restrictions ranging from not supporting the restore operation to such a source element to preventing multiple restore operations simultaneously. Also, after the operation is completed, it may be necessary to fracture the synchronization relationship. See GetSupportedFeatures in capabilities.
"Resume"	Continue the copy operation of a suspended relationship.	
"Reset To Sync"	Change Mode to Synchronous.	
"Reset To Async"	Change Mode to Asynchronous	
"Return To ResourcePool"	Delete a Snapshot target.	
"Reverse Roles"	Switch the source and the target element roles.	
"Split"	Separate the source and the target elements in a <i>consistent</i> manner.	
"Suspend"	Stop the copy operation in such a way that it can be resumed.	
"Unprepare"	Causes the synchronization to be reinitialized and stop in Prepared state.	



Table 292 compares the action of similar Operations.

**Table 292 - Comparison of Similar Operations**

Operations	Description
Activate vs. Resume	Activate: Activates a ReplicationSynchronizes association that has a CopyState of "Inactive." Resume: Resumes a FileSystemSynchronized association that has a CopyState of "Suspended".
Deactivate vs. Suspend	Deactivate: Stops the copy operation. In the case of Snapshots, all writes to target element are deleted (pointers to changed data are removed). While inactive, writes to source element will not be committed to target element once activated. Suspend: Stops the copy operation. All writes to target element are preserved. Once resumed, pending writes to target element are committed.
Fracture vs. Split	Fracture: Source and target elements are separated "abruptly." Split: Source and target elements are separated in an orderly fashion. Consistency of target elements is maintained.
Detach vs. Dissolve	Detach: The association between the source and target element must be first Fractured/Split before it can be Detached. Dissolve: The association can have a CopyState of Synchronized. Additionally, Dissolve can create a SettingsDefineState association based on GetSupportedFeatures (16.3.3.3) Capabilities.
Unsynchronized vs. Skewed	Unsynchronized: The source element contains data that has not been copied to the target element. Most likely, the copy operation is in the process of updating the target element (ProgressStatus=Synchronizing). Skewed: The target element has been updated by a host (e.g. target of a snapshot). Resynchronization is not automatic and requires an explicit "Resync" operation (i.e., ModifySynchronization)

#### 16.3.3.10 GetSupportedGroupOperations

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedGroupOperations(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported replication group Operations (see Table 290) on a FileSystemGroupSynchronized association that can be supplied to the ModifyReplicaSynchronization method.

#### 16.3.3.11 GetSupportedListOperations

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedListOperations(
    [IN] uint16 ReplicationType,
    [IN] uint16 SynchronizationType,
    [OUT] uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported replication Operations (see Table 290) on a list of associations that can be supplied to the ModifyListSynchronization method. The parameter SynchronizationType specifies the operations as they apply to a list of FileSystemSynchronized or FileSystemGroupSynchronized. If SynchronizationType is not specified, FileSystemSynchronized is assumed.

#### 16.3.3.12 GetSupportedSettingsDefineStateOperations

```
uint32 FileSystemReplicationServiceCapabilities.
    GetSupportedSettingsDefineStateOperations(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported operations on a SettingsDefineState association that can be supplied to the ModifySettingsDefineState method. Table 293 shows the list of SettingsDefineState operations that an implementation may support.

**Table 293 - SettingsDefineState Operations**

SettingsDefineState Operations	Description	Special Consideration
"Activate Consistency"	Enable consistency	
"Deactivate Consistency"	Disable consistency	
"Delete"	Remove the SettingsDefineState association. Instance of SynchronizationAspect may also be deleted if it is not shared with other elements.	
"Copy To Target"	Introduces the target elements and forms the necessary associations between the source and the target elements i.e., FileSystemSynchronized and FileSystemGroupSynchronized.	

#### 16.3.3.13 GetSupportedThinProvisioningFeatures

```
uint32 FileSystemReplicationServiceCapabilities.  
    GetSupportedThinProvisioningFeatures(  
        [IN] uint16 ReplicationType,  
        [OUT] uint16 SupportedThinProvisioningFeatures[] );
```

For a given ReplicationType this method returns the supported features related to thin provisioning. Table 294 shows the list of the Thin Provisioning Features an implementation may support.

A client can request a specific thin provisioning policy in the ReplicationSettingData parameter of the appropriate method call. See the property ReplicationSettingData.ThinProvisioningPolicy for the supported options for a copy operation.

**Table 294 - Thin Provisioning Features**

Feature	Description
"Thin provisioning is not supported"	The replication service does not distinguish between thinly and fully provisioned elements. The service treats all elements as fully provisioned elements.
"Zeros written in unused allocated blocks of target"	Applies to copying from a thinly provisioned element to a fully provisioned element. The implementation needs to allocate "real" storage blocks on the target side for the corresponding blocks of the source element that are unused. The implementation then writes zeros in the unused blocks of the target element.
"Unused allocated blocks of target are not initialized"	Applies to copying from a thinly provisioned element to a fully provisioned element. The implementation needs to allocate "real" storage blocks on the target side for the corresponding blocks of the source element that are unused.

#### 16.3.3.14 GetSupportedMaximum

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedMaximum(  
    [IN] uint16 ReplicationType,  
    [IN] uint16 Component,  
    [OUT] uint64 MaxValue );
```

This method accepts a ReplicationType and a component, it then returns a static numeric value representing the maximum number of the specified component that the service supports. A value of 0 indicates unlimited components of the given type. In all cases the maximum value is bounded by the availability of resources on the computer system. If the information is not known, the method returns 7 which indicates "Information is not available"

Effectively, this method informs clients of the edge conditions.

Table 295 shows the list of components that can be specified

**Table 295 - Components**

Components	Description
"Number of groups"	Maximum number of groups supported by the replication service.
"Number of elements per source group"	Maximum number of elements in a group that can be used as a source group.
"Number of elements per target group"	Maximum number of elements in a group that can be used as a target group.
"Number of target elements per source element"	Maximum number of target elements per source element.
"Number of total source elements"	Maximum number of total source elements supported by the service.
"Number of total target elements"	Maximum number of total target elements supported by the source.
"Number of peer systems"	Maximum number of peer systems that replication service can communicate with.
"Number of hops in multi-hop replication"	Maximum number of hops in multi-hop replication the service can manage.

#### 16.3.3.15 GetDefaultConsistency

```
uint32 FileSystemReplicationServiceCapabilities.GetDefaultConsistency(
    [IN] uint16 ReplicationType,
    [OUT] uint16 DefaultConsistency );
```

This method for a given ReplicationType, returns the default consistency value for the replication groups. Table 296 shows the list of possible Default Consistency values that an implementation may offer.

**Table 296 - Default Consistency**

DefaultConsistency	Description
"No default consistency"	Replication groups are not declared as consistent.
"Sequentially Consistent"	By default, a newly created replication group is declared to be consistent

#### 16.3.3.16 GetDefaultGroupPersistency

```
uint32 FileSystemReplicationServiceCapabilities.GetDefaultGroupPersistency(
    [OUT] uint16 DefaultGroupPersistency );
```

This method returns the default persistency for a newly created group. Table 297 shows the list of possible Group Persistency values that an implementation may offer.

**Table 297 - Default Group Persistency**

DefaultGroupPersistency	Description
"No default persistency"	Replication groups are not declared as persistent across the Provider reboots.
"Persistent"	By default, a newly created replication group is declared to be persistent across the Provider reboot (group is not temporary).

### 16.3.3.17 GetSupportedReplicationSettingData

```
uint32 FileSystemReplicationServiceCapabilities.  
    GetSupportedReplicationSettingData(  
        [IN] uint16 ReplicationType,  
        [IN] uint16 PropertyName,  
        [OUT] uint16 SupportedValues[] );
```

This method, for a given ReplicationType, returns an array of supported settings that can be utilized in an instance of the ReplicationSettingData class. See the MOF for the ReplicationSettingData class for the value map of the properties. Explanation of some of the properties appears below.

Table 298 shows the values for the property ReplicationSettingData.CopyMethodology.

**Table 298 - Copy Methodologies**

CopyMethodologies	Description
"Other"	A methodology not listed in this table.
"Implementation decides"	Implementation determines a suitable methodology
"Full-Copy"	All data is copied to the target element.
"Incremental-Copy"	Only changed data is copied to the target element.
"Differential-Copy"	Only the new writes are copied to the target element.
"Copy-On-Write"	Affected data is copied on the first write to the source or to the target elements.
"Copy-On-Access"	Affected data is copied on the first access to the source element.
"Delta-Update"	Difference based replication where initially the source element is copied to the target element. Then, at regular intervals, only changes to the source element that have taken place since the previous copy operation are incrementally updated to the target element. This copy operation is also referred to as asynchronous mirroring.
"Snap-And-Clone"	The service creates a snapshot of the source element first, then uses the snapshot as the source of the copy operation to the target element.

Table 299 shows the values for the property ReplicationSettingData.TargetElementSuppliers.

**Table 299 - Target Element Suppliers**

TargetElementSuppliers	Description
"Use existing"	Use existing elements only. If appropriate elements are not available, returns an error.
"Create new"	Create new target elements only.
"Use and create"	If appropriate elements are not available, create new target elements.
"Instrumentation decides"	
"Client must supply"	Client must supply target elements.

Table 300 shows the values for the property ReplicationSettingData.ThinProvisioningPolicy.

**Table 300 - ThinProvisioningPolicy**

Feature	Description
"Copy thin source to thin target"	Thinly provisioned source element is copied to a thinly provisioned target element.
"Copy thin source to full target"	Thinly provisioned source element is copied to a fully provisioned target element.
"Copy full source to thin target"	Fully provisioned source element is copied to a thinly provisioned target element.
"Provisioning of target same as source"	Provisioning of the target element is the same as the provisioning of the source element.
"Target pool decides provisioning of target element"	In the call to the CreateElementReplica or CreateGroupReplica method, the storage pool for the target elements is supplied. The supplied storage pool decides the provisioning of the created target elements.
"Implementation decides provisioning of target"	Vendor specific.

#### 16.3.3.18 GetDefaultReplicationSettingData

```
uint32 FileSystemReplicationServiceCapabilities.GetDefaultReplicationSettingData(
    [IN] uint16 ReplicationType,
    [OUT, EmbeddedObject]
    string DefaultInstance );
```

This method, for a given ReplicationType, returns the default ReplicationSettingData as an instance. Use this method to determine the implementation behavior for replication settings that do not have a distinct capability method.

#### 16.3.3.19 GetSupportedConnectionFeatures

```
uint32 FileSystemReplicationServiceCapabilities.GetSupportedConnectionFeatures(
    [IN] CIM_ProtocolEndpoint REF connection,
    [OUT] uint16 SupportedConnectionFeatures[] );
```

This method accepts a connection reference and returns specific features of that connection. Table 301 shows the list of possible Connection Features that an implementation may support.

**Table 301 - Connection Features**

ConnectionFeatures	Description
"Unidirectional to ProtocolEndpoint"	Direction of data flow to this ProtocolEndpoint, from a remote system (by default the connection is bi-directional).
"Unidirectional from ProtocolEndpoint"	Direction of data flow from this ProtocolEndpoint to a remote system (by default the connection is bi-directional).

#### 16.3.3.20 GetSupportedStorageCompressionFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedStorageCompressionFeatures(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedStorageCompressionFeatures[] );
```

For a given ReplicationType this method returns the supported features related to storage compression. Table 302 shows the list of the Storage Compression Features an implementation may support.

**Table 302 - Storage Compression Features**

Feature	Description
"Storage compression is not supported"	The replication service does not support storage compression. Only uncompressed elements are accepted.
"Compressed source to compressed target"	The replication service supports copying from compressed source element to compressed target element.
"Compressed source to uncompressed target"	The replication service supports copying from compressed source element to uncompressed target element.
"Uncompressed source to compressed target"	The replication service supports copying from uncompressed source element to compressed target element.
"Compression of target same as source"	The source element is copied to a target with the same compression setting as the source.
Target pool decides compression of target element"	In the call to the CreateElementReplica or CreateGroupReplica method, the storage pool for the target elements is supplied. The supplied storage pool decides the compression of the created target elements.
"Implementation decides compression of target"	Leaves implementation to decide compression setting of the target.

## 16.4 Use Cases

In general, creating and managing replicas involves the following steps:

- Decide on the SyncType of replica (Mirror, Snapshot, Clone) and Mode (Synchronous, Asynchronous). See 16.1.3.1: SyncTypes.
- Locate the hosted instance of ReplicationService. See 16.1.3: Filesystem Replication Services Discovery.
- Locate the instance of FileSystemReplicationServiceCapabilities. Utilize its properties and methods to determine the applicable capabilities offered by the implementation for the desired ReplicationType (includes SyncType and Mode). See 16.1.3: Filesystem Replication Services Discovery.
- Use the method ReplicationService.GetAvailableTargetElements to locate appropriate target elements. Depending on the implementation, it is also possible to allow the service to locate target elements. See 16.3.2.10: GetAvailableTargetElements.
- Verify StoragePools have sufficient free capacity for the target elements.
- If necessary, use the ReplicationService's group manipulation methods to create and populate source and target groups. See 16.3: Methods.
- Invoke the appropriate extrinsic method of the ReplicationService to create a replica. See 16.3: Methods.
- Monitor the copy operation's progress by examining the replication associations properties, or subscribe to the appropriate indications -- including storage pool low space alert indications. 16.1.6: State Management For Associated Replicas and 16.1.10: Indication.
- Invoke the method ReplicationService.ModifyReplicaSynchronization to modify a replica. For example, "split" a replica from its source element. See 16.3: Methods.

## 16.5 CIM Elements

Table 303 describes the CIM elements for Filesystem Replication Services.

**Table 303 - CIM Elements for Filesystem Replication Services**

Element Name	Requirement	Description
16.5.1: CIM_AllocatedResources	Optional	This is a SystemSpecificCollection for collecting components that are being used by the FileSystem Replication Services profile (e.g., FileSystem, LogicalDisk, etc.) that supports Cascading.
16.5.2: CIM_ElementCapabilities	Mandatory	Associates FileSystemReplicationCapabilities and ReplicationService, or FileSystemReplicationServiceCapabilities and ReplicationService.
16.5.3: CIM_FileSystemGroupSynchronized	Conditional	Experimental. Conditional requirement: Required if groups are supported. Associates source and target groups, or a source element to a target group.
16.5.4: CIM_FileSystemReplicationServiceCapabilities	Mandatory	Experimental. A set of properties and methods that describe the capabilities of a replication services provider.
16.5.5: CIM_FileSystemSynchronized	Mandatory	Experimental. Associates replica target element to source element.
16.5.6: CIM_HostedAccessPoint (ForProtocolEndpoint)	Conditional	Conditional requirement: Required if remote replication is supported. Associates ProtocolEndpoint to the ComputerSystem on which it is hosted.
16.5.7: CIM_HostedAccessPoint (ForRemoteServiceAccessPoint)	Conditional	Conditional requirement: Required if remote replication is supported. Associates RemoteServiceAccessPoint to the ComputerSystem.
16.5.8: CIM_HostedCollection (Allocated Resources)	Mandatory	This would associate the AllocatedResources collection to the top level system for the FileSystem Replication Services Profile using Cascading.
16.5.9: CIM_HostedCollection (Between ComputerSystem and RemoteReplicationCollection)	Conditional	Conditional requirement: Required if groups are supported. Associates the RemoteReplicationCollection (ConnectivityCollection) to the hosting System.
16.5.10: CIM_HostedCollection (Between ComputerSystem and ReplicationGroup)	Conditional	Conditional requirement: Required if groups are supported. Associates the replication group to the hosting System.
16.5.11: CIM_HostedCollection (Remote Resources)	Conditional	Conditional requirement: This is required if CIM_RemoteResources is modeled. This would associate the RemoteResources collection to the top level system for the Replication Services Profile in support of Cascading.
16.5.12: CIM_HostedService	Mandatory	
16.5.13: CIM_MemberOfCollection (Allocated Resources)	Optional	This supports collecting replication components. This is required to support the AllocatedResources collection for Cascading.
16.5.14: CIM_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection)	Conditional	Conditional requirement: Required if remote replication is supported. Associates ProtocolEndpoints to RemoteReplicationCollection (ConnectivityCollection).

**Table 303 - CIM Elements for Filesystem Replication Services**

Element Name	Requirement	Description
16.5.15: CIM_MemberOfCollection (Remote Resources)	Optional	This supports collecting all Shadow instances of components that the Replication Service has available to use. This is optional when used to support the RemoteResources collection (the RemoteResources collection is optional).
16.5.16: CIM_OrderedMemberOfCollection	Conditional	Conditional requirement: Required if groups are supported. Associates ReplicationGroup to storage elements.
16.5.17: CIM_ProtocolEndpoint	Conditional	Conditional requirement: Required if remote replication is supported. Special purpose endpoint that represents connections between systems.
16.5.18: CIM_RemoteReplicationCollection	Conditional	Conditional requirement: Required if remote replication is supported. A RemoteReplicationCollection groups together a set of ProtocolEndpoints of the same 'type' (i.e., class) which are able to communicate with each other. The ProtocolEndpoints are used by Replication Services.
16.5.19: CIM_RemoteResources	Optional	This is a SystemSpecificCollection for collecting components that may be allocated by the Replication Services profile (e.g., FileSystem) that supports Cascading.
16.5.20: CIM_RemoteServiceAccessPoint	Conditional	Conditional requirement: Required if remote replication is supported. A ServiceAccessPoint for replication service.
16.5.21: CIM_ReplicaPoolForStorage	Optional	Associates special storage pool for Snapshots (delta replicas) to a source element.
16.5.22: CIM_ReplicationEntity	Optional	Represents a replication entity such as an entity known by its World Wide Name (WWN).
16.5.23: CIM_ReplicationGroup	Conditional	Experimental. Conditional requirement: Required if groups are supported. Represents a group of elements participating in a replication activity.
16.5.24: CIM_ReplicationService	Mandatory	Experimental. Base class for FileSystem Replication Services. Methods are described in the Extrinsic Methods clause.
16.5.25: CIM_ReplicationSettingData	Optional	Experimental. Contains special options for use by methods of Replication Services.
16.5.26: CIM_SAPAvailableForElement	Conditional	Conditional requirement: Required if remote replication is supported. This association identifies the element that is serviced by the ServiceAccessPoint.
16.5.27: CIM_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection)	Conditional	Conditional requirement: Required if remote replication is supported. Associates Replication Service to RemoteReplicationCollection (ConnectivityCollection).
16.5.28: CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity)	Optional	Associates Replication Service to ReplicationEntity.
16.5.29: CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup)	Conditional	Conditional requirement: Required if groups are supported. Associates Replication Service to Replication Group.



**Table 303 - CIM Elements for Filesystem Replication Services**

Element Name	Requirement	Description
16.5.30: CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)	Optional	Associates a replication group to an instance of SynchronizationAspect.
16.5.31: CIM_SettingsDefineState (Between storage object and SynchronizationAspect)	Optional	Associates a storage object to an instance of SynchronizationAspect.
16.5.32: CIM_SharedSecret	Conditional	Conditional requirement: Required if remote replication is supported.
16.5.33: CIM_SynchronizationAspect	Optional	Experimental. Keeps track of the source of a copy operation, even after FileSystemSynchronized is removed. Also keeps track of point-in-time.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FileSystemSynchronized	Mandatory	All instance creation indications for FileSystemSynchronized.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FileSystemGroupSynchronized	Conditional	Conditional requirement: Required if groups are supported. All instance creation indications for FileSystemGroupSynchronized.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SynchronizationAspect	Optional	All instance creation indications for SynchronizationAspect.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FileSystemSynchronized AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-FileSystemSynchronized')	Conditional	Conditional requirement: Required if semi-fixed indication filters are supported. CQL -Instance deletion indications for a specific FileSystemSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FileSystemSynchronized	Optional	All instance deletion indications for FileSystemSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FileSystemGroupSynchronized AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-FileSystemGroupSynchronized')	Conditional	Conditional requirement: Required if groups and semi-fixed indication filters are supported. CQL - Instance deletion indications for a specific FileSystemGroupSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FileSystemGroupSynchronized	Optional	All instance deletion indications for FileSystemGroupSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SynchronizationAspect	Optional	All instance deletion indications for SynchronizationAspect.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileSystemSynchronized AND SourceInstance.CIM_FileSystemSynchronized::CopyState <> PreviousInstance.CIM_FileSystemSynchronized::CopyState AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-FileSystemSynchronized')	Conditional	Conditional requirement: Required if semi-fixed indication filters are supported. CQL - Synchronization state transition for a specific replica association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileSystemSynchronized AND SourceInstance.CIM_FileSystemSynchronized::CopyState <> PreviousInstance.CIM_FileSystemSynchronized::CopyState	Optional	CQL -Synchronization state transition for replica associations.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileSystemSynchronized AND SourceInstance.CIM_FileSystemSynchronized::ProgressStatus <> PreviousInstance.CIM_FileSystemSynchronized::ProgressStatus AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-FileSystemSynchronized')	Optional	CQL -Progress status transition for a specific replica association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileSystemSynchronized AND SourceInstance.CIM_FileSystemSynchronized::ProgressStatus <> PreviousInstance.CIM_FileSystemSynchronized::ProgressStatus	Optional	CQL -Progress status transition for replica associations.

**Table 303 - CIM Elements for Filesystem Replication Services**

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileSystemGroupSynchronized AND SourceInstance.CIM_FileSystemGroupSynchronized::CopyState <> PreviousInstance.CIM_FileSystemGroupSynchronized::CopyState AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-FileSystemGroupSynchronized')	Conditional	Conditional requirement: Required if groups and semi-fixed indication filters are supported. CQL - Synchronization state transition for a specific replication group association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FileSystemGroupSynchronized AND SourceInstance.CIM_FileSystemGroupSynchronized::CopyState <> PreviousInstance.CIM_FileSystemGroupSynchronized::CopyState	Optional	CQL -Synchronization state transition for replication group associations.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM4'	Mandatory	Be notified when CopyState is set to Broken.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM5'	Mandatory	Remaining pool space either below warning threshold set for the pool or there is no remaining space in the pool.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM6'	Mandatory	Be notified of changes in RemoteReplicationCollection (ConnectivityCollections).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM7'	Mandatory	Be notified of changes in ProtocolEndpoints.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM8'	Conditional	Be notified when CopyState is Fractured if the implementation supports this state.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM9'	Conditional	Be notified when CopyState is Invalid if the implementation supports this state.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM10'	Mandatory	Be notified when the CopyState is Inactive.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM11'	Optional	Be notified when the CopyState is Split.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND MessageID='FSM12'	Mandatory	Be notified when the CopyState returns to a normal condition.

### 16.5.1 CIM\_AllocatedResources

An instance of a default CIM\_AllocatedResources defines the set of components that are allocated and in use by the Replication Services Profile. CIM\_AllocatedResources is subclassed from CIM\_SystemSpecificCollection. At least one instance of the CIM\_AllocatedResources shall exist for the Replication Services Profile and shall be hosted by one of its ComputerSystems (typically the top level ComputerSystem).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 304 describes class CIM\_AllocatedResources.

**Table 304 - SMI Referenced Properties/Methods for CIM\_AllocatedResources**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the AllocatedResources collection (e.g., Allocated FileSystem).
ElementType		Mandatory	The type of remote resources collected by the AllocatedResources collection. For this version of SMI-S, the only value supported is '2' (Any Type).
CollectionDiscriminator		Mandatory	Experimental. This is an array of values that shall contain one or more values from the list: 'SNIA:Target Volumes', 'SNIA:Source Volumes', 'SNIA:Target Volume Group', 'SNIA:Source Volume Group'.

### 16.5.2 CIM\_ElementCapabilities

Associates FileSystemReplicationCapabilities and ReplicationService.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 305 describes class CIM\_ElementCapabilities.

**Table 305 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities**

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

### 16.5.3 CIM\_FileSystemGroupSynchronized

Experimental.

Requirement: Required if groups are supported.

### 16.5.4 CIM\_FileSystemReplicationServiceCapabilities

Experimental. This class defines all of the capability properties for the replication services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 306 describes class CIM\_FileSystemReplicationServiceCapabilities.

**Table 306 - SMI Referenced Properties/Methods for CIM\_FileSystemReplicationServiceCapabilities**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User Friendly name.
SupportedReplicationTypes		Mandatory	Enumeration indicating the supported SyncType/Mode/Local or Remote combinations. Values: 2: Synchronous Mirror Local 3: Asynchronous Mirror Local 4: Synchronous Mirror Remote 5: Asynchronous Mirror Remote 6: Synchronous Snapshot Local 7: Asynchronous Snapshot Local 8: Synchronous Snapshot Remote 9: Asynchronous Snapshot Remote 10: Synchronous Clone Local 11: Asynchronous Clone Local 12: Synchronous Clone Remote 13: Asynchronous Clone Remote.
SupportedStorageObjects		Mandatory	Enumeration indicating the supported storage objects. Values: 2: FileSystem 3: LogicalFile.
SupportedAsynchronousActions		Mandatory	Identify replication methods using job control. Values: 2: CreateElementReplica 3: CreateGroupReplica 4: CreateSynchronizationAspect 5: ModifyReplicaSynchronization 6: ModifyListSynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 9: GetPeerSystems 10: GetReplicationRelationships 11: GetServiceAccessPoints 19: CreateListReplica.

**Table 306 - SMI Referenced Properties/Methods for CIM\_FileSystemReplicationServiceCapabilities**

Properties	Flags	Requirement	Description & Notes
SupportedSynchronousActions		Mandatory	Identify replication methods not using job control. Values: 2: CreateElementReplica 3: CreateGroupReplica 4: CreateSynchronizationAspect 5: ModifyReplicaSynchronization 6: ModifyListSynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 9: GetPeerSystems 10: GetReplicationRelationships 11: GetServiceAccessPoints 12: CreateGroup 13: DeleteGroup 14: AddMembers 15: RemoveMembers 16: AddReplicationEntity 17: AddServiceAccessPoint 18: AddSharedSecret 19: CreateListReplica.
ConvertSyncTypeToReplicationType()		Mandatory	
ConvertReplicationTypeToSyncType()		Mandatory	
GetSupportedCopyStates()		Mandatory	
GetSupportedGroupCopyStates()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedWaitForCopyStates()		Optional	
GetSupportedFeatures()		Mandatory	
GetSupportedGroupFeatures()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedConsistency()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedOperations()		Mandatory	
GetSupportedGroupOperations()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedListOperations()		Optional	
GetSupportedSettingsDefineStateOperations()		Optional	
GetSupportedThinProvisioningFeatures()		Optional	
GetSupportedStorageCompressionFeatures()		Optional	
GetSupportedMaximum()		Optional	

**Table 306 - SMI Referenced Properties/Methods for CIM\_FileSystemReplicationServiceCapabilities**

Properties	Flags	Requirement	Description & Notes
GetDefaultConsistency()		Conditional	Conditional requirement: Required if groups are supported.
GetDefaultGroupPersistency()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedReplicationSettingData()		Optional	
GetDefaultReplicationSettingData()		Optional	
GetSupportedConnectionFeatures()		Optional	

### 16.5.5 CIM\_FileSystemSynchronized

Experimental. Associates replica target element to source element.

Created By: Extrinsic: CreateElementReplica, CreateGroupReplica, CreateListReplica

Modified By: Extrinsic: ModifyReplicaSynchronization

Deleted By: Extrinsic: ModifyReplicaSynchronization

Requirement: Mandatory

Table 307 describes class CIM\_FileSystemSynchronized.

**Table 307 - SMI Referenced Properties/Methods for CIM\_FileSystemSynchronized**

Properties	Flags	Requirement	Description & Notes
WhenSynced	N	Optional	Date and time synchronization of the elements is achieved.
WhenEstablished	N	Optional	Specifies when the association was established.
WhenSynchronized	N	Optional	Specifies when the CopyState has a value of Synchronized.
WhenActivated	N	Optional	Specifies when the association was activated.
WhenSuspended	N	Optional	Specifies when the association was suspended.
SyncMaintained		Mandatory	Boolean indicating whether synchronization is maintained.
SyncType		Mandatory	Type of association between source and target groups. Values: 6: Mirror 7: Snapshot 8: Clone.
Mode		Mandatory	Specifies when target elements are updated. Values: 2: Synchronous 3: Asynchronous.

**Table 307 - SMI Referenced Properties/Methods for CIM\_FileSystemSynchronized**

Properties	Flags	Requirement	Description & Notes
RequestedCopyState		Optional	Indicates the last requested or desired state for the association. Values: 4: Synchronized 6: Fractured 7: Split 8: Inactive 9: Suspended 10: Failedover 11: Prepared 12: Aborted 15: Not Applicable 16: Partitioned 17: Invalid.
ReplicaType		Optional	
CopyState		Mandatory	State of association between source and target groups. Values: 2: Initialized 3: Unsynchronized 4: Synchronized 5: Broken 6: Fractured 7: Split 8: Inactive 9: Suspended 10: FailedOver 11: Prepared 12: Aborted 13: Skewed 14: Mixed 15: Not Applicable 16: Partitioned 17: Invalid.

**Table 307 - SMI Referenced Properties/Methods for CIM\_FileSystemSynchronized**

Properties	Flags	Requirement	Description & Notes
ProgressStatus	N	Optional	Status of association between source and target groups. Values: 2: Completed 3: Dormant 4: Initializing 5: Preparing 6: Synchronizing 7: Resyncing 8: Restoring 9: Fracturing 10: Splitting 11: Failing over 12: Failing back 13: Aborting 14: Mixed 15: Not Applicable 16: Suspending 17: Requires fracture 18: Requires resync 19: Requires activate 20: Pending 21: Detaching 22: Requires detach.
PercentSynced	N	Optional	Specifies the percent of the work completed to reach synchronization. For synchronized associations (e.g. SyncType Mirror), while fractured, the percent difference between source and target elements can derived by subtracting PercentSynced from 100.
CopyPriority	MN	Optional	CopyPriority allows the priority of background copy engine I/O to be managed relative to host I/O operations during a sequential background copy operation. Values: 0: Not Managed 1: Low 2: Same (as host I/O) 3: High 4: Urgent.
UndiscoveredElement	N	Optional	Specifies whether the source, the target, or both elements involved in a copy operation are undiscovered. If NULL both source and target elements are considered discovered. Values: 2: SystemElement 3: SyncedElement 4: Both.



**Table 307 - SMI Referenced Properties/Methods for CIM\_FileSystemSynchronized**

Properties	Flags	Requirement	Description & Notes
FailedCopyStopsHostIO	N	Optional	If true, the storage array tells host to stop sending data to source element if copying to a remote element fails. To set this property initially, use ReplicationSettingData parameter in create method. To modify this property, use ModifyInstance intrinsic method.
CopyRecoveryMode	N	Optional	Describes whether the copy operation continues after a broken link is restored. If Manual, the CopyState is set to Suspended after the link is restored. It is required to issue the Resume operation to continue. To set this property initially, use ReplicationSettingData parameter in create method. To modify this property, use ModifyInstance intrinsic method. Values:  2: Automatic 3: Manual 4: Implementation decides.
ReadOnly	N	Optional	This property specifies whether the source, the target, or both elements are "read only" to the host. Values:  2: SystemElement 3: SyncedElement 4: Both.
SyncedElement		Mandatory	
SystemElement		Mandatory	

#### 16.5.6 CIM\_HostedAccessPoint (ForProtocolEndpoint)

Associates ProtocolEndpoint to the System on which it is hosted.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 308 describes class CIM\_HostedAccessPoint (ForProtocolEndpoint).

**Table 308 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint (ForProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Hosting System.
Dependent		Mandatory	The access points that are hosted on this System.

#### 16.5.7 CIM\_HostedAccessPoint (ForRemoteServiceAccessPoint)

Associates RemoteServiceAccessPoint to the ComputerSystem.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 309 describes class CIM\_HostedAccessPoint (ForRemoteServiceAccessPoint).

**Table 309 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint (ForRemoteServiceAccessPoint)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Hosting System.
Dependent		Mandatory	The access points that are hosted on this System.

#### 16.5.8 CIM\_HostedCollection (Allocated Resources)

CIM\_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the Replication Services profile, it is used to associate the Allocated Resources to the top level Computer System of the Replication Services Profile in support of Cascading.

CIM\_HostedCollection is subclassed from CIM\_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 310 describes class CIM\_HostedCollection (Allocated Resources).

**Table 310 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Allocated Resources)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

#### 16.5.9 CIM\_HostedCollection (Between ComputerSystem and RemoteReplicationCollection)

Associates the RemoteReplicationCollection (ConnectivityCollection) to the hosting System.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if groups are supported.

Table 311 describes class CIM\_HostedCollection (Between ComputerSystem and RemoteReplicationCollection).

**Table 311 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Between ComputerSystem and RemoteReplicationCollection)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

### 16.5.10 CIM\_HostedCollection (Between ComputerSystem and ReplicationGroup)

Associates the replication group to the hosting System.

Created By: Extrinsic: CreateGroup

Modified By: Extrinsic: DeleteGroup, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Required if groups are supported.

Table 312 describes class CIM\_HostedCollection (Between ComputerSystem and ReplicationGroup).

**Table 312 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Between ComputerSystem and ReplicationGroup)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

### 16.5.11 CIM\_HostedCollection (Remote Resources)

CIM\_HostedCollection defines a SystemSpecificCollection in the context of a scoping System. It represents a Collection that only has meaning in the context of a System, and/or whose elements are restricted by the definition of the System. In the FileSystem Replication Services Profile, it is used to associate the Remote Resources to the top level Computer System of the FileSystem Replication Services Profile that supports Cascading.

CIM\_HostedCollection is subclassed from CIM\_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if CIM\_RemoteResources is modeled.

Table 313 describes class CIM\_HostedCollection (Remote Resources).

**Table 313 - SMI Referenced Properties/Methods for CIM\_HostedCollection (Remote Resources)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

### 16.5.12 CIM\_HostedService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 314 describes class CIM\_HostedService.

**Table 314 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Replication Service hosted on the System.

#### 16.5.13 CIM\_MemberOfCollection (Allocated Resources)

This use of MemberOfCollection is to collect all allocated shadow component instances (in the AllocatedResources collection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 315 describes class CIM\_MemberOfCollection (Allocated Resources).

**Table 315 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Allocated Resources)**

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

#### 16.5.14 CIM\_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection)

Associates ProtocolEndpoints to RemoteReplicationCollection (ConnectivityCollection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 316 describes class CIM\_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection).

**Table 316 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection)**

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

#### 16.5.15 CIM\_MemberOfCollection (Remote Resources)

This use of MemberOfCollection is to collect all shadow components (in the RemoteResources collection). Each association (and the RemoteResources collection, itself) is created through external means.

Created By: Static  
 Modified By: Static  
 Deleted By: Static  
 Requirement: Optional

Table 317 describes class CIM\_MemberOfCollection (Remote Resources).

**Table 317 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection (Remote Resources)**

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

#### 16.5.16 CIM\_OrderedMemberOfCollection

Associates ReplicationGroup to storage elements.

Created By: Extrinsic: CreateGroup  
 Modified By: Extrinsic: AddMembers, RemoveMembers  
 Deleted By: Extrinsic: DeleteGroup, RemoveMembers  
 Requirement: Required if groups are supported.

Table 318 describes class CIM\_OrderedMemberOfCollection.

**Table 318 - SMI Referenced Properties/Methods for CIM\_OrderedMemberOfCollection**

Properties	Flags	Requirement	Description & Notes
AssignedSequence		Mandatory	
Collection		Mandatory	
Member		Mandatory	

#### 16.5.17 CIM\_ProtocolEndpoint

Special purpose endpoint that represents connections between systems.

Created By: Static  
 Modified By: Static  
 Deleted By: Static  
 Requirement: Required if remote replication is supported.

Table 319 describes class CIM\_ProtocolEndpoint.

**Table 319 - SMI Referenced Properties/Methods for CIM\_ProtocolEndpoint**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	

**Table 319 - SMI Referenced Properties/Methods for CIM\_ProtocolEndpoint**

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	
ProtocolType		Mandatory	Value always reflects protocol type. Values: 1: Other 6: Ethernet CSMA/CD 7: ISO 802.3 CSMA/CD 8: ISO 802.4 Token Bus 15: FDDI 56: Fibre Channel 117: Gigabit Ethernet 4096: IPv4 4097: IPv6 4098: IPv4/IPv6 4111: TCP.
OtherTypeDescription	N	Optional	String identifying the Other connection protocol.
OperationalStatus		Mandatory	An array containing the operational status of protocol end point.

#### 16.5.18 CIM\_RemoteReplicationCollection

Collects the ProtocolEndpoints/ServiceAccessPoints used by Replication Services.

Created By: Extrinsic: CreateRemoteReplicationCollection

Modified By: Extrinsic: AddToRemoteReplicationCollection, RemoveFromRemoteReplicationCollection

Deleted By: Extrinsic: Static

Requirement: Required if remote replication is supported.

Table 320 describes class CIM\_RemoteReplicationCollection.

**Table 320 - SMI Referenced Properties/Methods for CIM\_RemoteReplicationCollection**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque.
ElementName		Optional	User Friendly name.
ConnectivityStatus		Mandatory	An enumeration describing the current or potential connectivity between endpoints in this collection. Values: 2: Connectivity - Up 3: No Connectivity - Down 4: Partitioned - Partial connectivity.
Active	N	Optional	Indicates that this collection is currently active and allows replication activities to the remote elements.
DeleteOnUnassociated	N	Optional	If true, this instance of RemoteReplicationCollection will be deleted when it is no longer associated with an access point.

### 16.5.19CIM\_RemoteResources

An instance of a default CIM\_RemoteResources defines the set of shadow components that are available to be used by the Replication Services Profile that supports Cascading. CIM\_RemoteResources is subclassed from CIM\_SystemSpecificCollection. One instance of the CIM\_RemoteResources would exist and shall be hosted by the top level ComputerSystems of the Replication Services Profile that supports Cascading.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 321 describes class CIM\_RemoteResources.

**Table 321 - SMI Referenced Properties/Methods for CIM\_RemoteResources**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the RemoteResources collection (e.g., Remote FileSystem).
ElementType		Mandatory	The type of remote resources collected by the RemoteResources collection. This shall be '2' (Any Type).
CollectionDiscriminator		Mandatory	Experimental. This is an array of values that shall contain one or more values from the list: 'SNIA:Target Volumes', 'SNIA:Source Volumes', 'SNIA:Target Volume Group', 'SNIA:Source Volume Group', 'SNIA:Remote Storage Pools'.

### 16.5.20CIM\_RemoteServiceAccessPoint

Created By: Extrinsic: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 322 describes class CIM\_RemoteServiceAccessPoint.

**Table 322 - SMI Referenced Properties/Methods for CIM\_RemoteServiceAccessPoint**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
ElementName		Optional	User Friendly name.

**Table 322 - SMI Referenced Properties/Methods for CIM\_RemoteServiceAccessPoint**

Properties	Flags	Requirement	Description & Notes
AccessInfo		Mandatory	Access or addressing information or a combination of this information for a remote connection. This information can be a host name, network address, or similar information.
InfoFormat		Mandatory	The format of the Management Address (i.e., AccessInfo). For example: "Host Name", "IPv4 Address", "IPv6 Address", "URL". See MOF for the complete list and values.

#### 16.5.21 CIM\_ReplicaPoolForStorage

Associates special storage pool for Snapshots (delta replicas) to a source element.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 323 describes class CIM\_ReplicaPoolForStorage.

**Table 323 - SMI Referenced Properties/Methods for CIM\_ReplicaPoolForStorage**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

#### 16.5.22 CIM\_ReplicationEntity

Represents a replication entity such as an entity known by its World Wide Name (WWN).

Created By: Extrinsic: AddReplicationEntity

Modified By: Static

Deleted By: Static

Requirement: Optional



Table 324 describes class CIM\_ReplicationEntity.

**Table 324 - SMI Referenced Properties/Methods for CIM\_ReplicationEntity**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
Type		Mandatory	Indicates how to interpret the information appearing in EntityID. Values: 2: StoragePool 3: StorageExtent 4: StorageVolume 5: LogicalDisk 6: Filesystem 7: WWN 8: URI 9: Objectpath 10: Encoded in EntityID.
EntityID		Mandatory	An ID representing the resource identified by this entity. For example, the WWN or the URI of an element. The property Type is to be used to interpret the ID.
OtherTypeDescription	N	Optional	Populated when Type has the value of Other.
Persistent	MN	Optional	If false, the instance of this object, not the element represented by this entity, may be deleted at the completion of a copy operation.

### 16.5.23 CIM\_ReplicationGroup

Experimental. Represents a group of elements participating in a replication activity.

Created By: Extrinsic: CreateGroup

Modified By: Extrinsic: AddMembers, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Required if groups are supported.

Table 325 describes class CIM\_ReplicationGroup.

**Table 325 - SMI Referenced Properties/Methods for CIM\_ReplicationGroup**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class.
Persistent	MN	Optional	If false, the group, not the elements associated with the group, may be deleted at the completion of a copy operation.
DeleteOnEmptyElement	M	Mandatory	If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty.

**Table 325 - SMI Referenced Properties/Methods for CIM\_ReplicationGroup**

Properties	Flags	Requirement	Description & Notes
DeleteOnUnassociated	M	Mandatory	If true, the group will be deleted when the group is no longer associated with another group. This can happen if all synchronization associations to the individual elements of the group are dissolved.
ConsistentPointInTime	N	Optional	If it is true, it means the point-in-time was created at an exact time with no I/O activities in such a way the data is consistent among all the elements of the group. This property is only valid when the group is a target of a copy operation.

#### 16.5.24 CIM\_ReplicationService

Experimental. Base class for FileSystem Replication Services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 326 describes class CIM\_ReplicationService.

**Table 326 - SMI Referenced Properties/Methods for CIM\_ReplicationService**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
CreateElementReplica()		Mandatory	
CreateGroupReplica()		Conditional	Conditional requirement: Required if groups are supported.
CreateListReplica()		Optional	
CreateSynchronizationAspect()		Optional	
ModifyReplicaSynchronization()		Mandatory	
ModifyListSynchronization()		Optional	
ModifySettingsDefineState()		Optional	
CreateGroup()		Conditional	Conditional requirement: Required if groups are supported.
DeleteGroup()		Conditional	Conditional requirement: Required if groups are supported.
AddMembers()		Conditional	Conditional requirement: Required if groups are supported.
RemoveMembers()		Conditional	Conditional requirement: Required if groups are supported.
GetAvailableTargetElements()		Optional	
GetPeerSystems()		Optional	

**Table 326 - SMI Referenced Properties/Methods for CIM\_ReplicationService**

Properties	Flags	Requirement	Description & Notes
GetReplicationRelationships()		Optional	
GetServiceAccessPoints()		Optional	
AddReplicationEntity()		Optional	
AddServiceAccessPoint()		Optional	
AddSharedSecret()		Optional	
CreateGroupReplicaFromElements()		Optional	
GetReplicationRelationshipInstance()		Optional	
ModifyListSettingsDefineState()		Optional	
CreateRemoteReplicationCollection()		Optional	
AddToRemoteReplicationCollection()		Optional	
RemoveFromRemoteReplicationCollection()		Optional	

#### 16.5.25 CIM\_ReplicationSettingData

Experimental. Contains special options for use by methods of Replication Services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 327 describes class CIM\_ReplicationSettingData.

**Table 327 - SMI Referenced Properties/Methods for CIM\_ReplicationSettingData**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User Friendly name.
Pairing	MN	Optional	Controls how source and target elements are paired. Values: 2: Instrumentation decides 3: Exact order 4: Optimum (If possible source and target elements on different adapters).
UnequalGroupsAction	MN	Optional	Indicates what should happen if number of elements in source and target are unequal. Values: 2: Return an error 3: Allow larger source group 4: Allow larger target group.

**Table 327 - SMI Referenced Properties/Methods for CIM\_ReplicationSettingData**

Properties	Flags	Requirement	Description & Notes
DesiredCopyMethodology	MN	Optional	Request specific copy methodology. Values: 1: Other 2: Instrumentation decides 3: Full-Copy 4: Incremental-Copy 5: Differential-Copy 6: Copy-On-Write 7: Copy-On-Access 8: Delta-Update 9: Snap-And-Clone.
TargetElementSupplier	MN	Optional	If target elements are not supplied, this property indicates where the target elements should come from. Values: 1: Use existing elements 2: Create new elements 3: Use existing or Create new elements 4: Instrumentation decides 5: Client must supply.
ThinProvisioningPolicy	MN	Optional	If the target element is not supplied, this property specifies the provisioning of the target element. Values: 2: Copy thin source to thin target 3: Copy thin source to full target 4: Copy full source to thin target 5: Provisioning of target same as source 6: Target pool decides provisioning of target element 7: Implementation decides provisioning of target.
StorageCompressionPolicy	MN	Optional	If the target element is not supplied, this property specifies the compression of the target element. Values: 2: Copy compressed source to compressed target 3: Copy compressed source to uncompressed target 4: Copy uncompressed source to compressed target 5: Compression of target same as source 6: Target pool decides compression of target element 7: Implementation decides compression of target.
ConsistentPointInTime	MN	Optional	If it is true, it means the point-in-time to be created at an exact time with no I/O activities in such a way the data is consistent among all the elements or the group.
DeltaUpdateInterval	MN	Optional	If non-zero, it specifies the interval between the snapshots of source element, for example, every 23 minutes (00000000002300.000000:000). If zero or NULL, the implementation decides.
Multihop	MN	Optional	This property applies to multihop copy operation. It specifies the number of hops the starting source (or group) element is expected to be copied. Default is 1.

**Table 327 - SMI Referenced Properties/Methods for CIM\_ReplicationSettingData**

Properties	Flags	Requirement	Description & Notes
OnGroupOrListError	MN	Optional	This property applies to group or list operations. It specifies what the implementation should do if an error is encountered before all entries in the group or list are processed. Default is to Stop.  2: Continue 3: Stop.
CopyPriority	MN	Optional	This property sets the StorageSynchronized.CopyPriority property. CopyPriority allows the priority of background copy operation to be managed relative to host I/O operations during a sequential background copy operation.  0: Not Managed 1: Low 2: Same (as host I/O) 3: High 4: Urgent.
FailedCopyStopsHostIO	MN	Optional	If true, the storage array tells host to stop sending data to source element if copying to a remote element fails.
CopyRecoveryMode	MN	Optional	Describes whether the copy operation continues after a broken link is restored. If Manual, the CopyState is set to Suspended after the link is restored. It is required to issue the Resume operation to continue. Values:  2: Automatic 3: Manual 4: Implementation decides.
UnequalListsAction	MN	Optional	Indicates what should happen if number of elements in source and target lists are unequal. Values:  2: Return an error 3: Allow source list to be larger 4: Allow target list to be larger.
DeltaUpdateBlocks	MN	Optional	This property applies to Delta-Update copy methodology. If non-zero, it specifies the snapshots of source element should be created after this number of blocks have been modified. If both DeltaUpdateBlocks and DeltaUpdateInterval are specified the snapshot is created based on which criterion occurs first. If NULL or 0, the implementation decides the number of blocks.
ReadOnly	MN	Optional	This property specifies whether the source, the target, or both elements should be read only to the host. Values:  2: SystemElement (source) 3: SyncedElement (target) 4: Both.

#### 16.5.26 CIM\_SAPAvailableForElement

This association identifies the element that is serviced by the ProtocolEndpoint.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 328 describes class CIM\_SAPAvailableForElement.

**Table 328 - SMI Referenced Properties/Methods for CIM\_SAPAvailableForElement**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element.
AvailableSAP		Mandatory	The servicing protocol end point.

#### **16.5.27 CIM\_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection)**

Associates Replication Service to RemoteReplicationCollection (ConnectivityCollection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 329 describes class CIM\_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection).

**Table 329 - SMI Referenced Properties/Methods for CIM\_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection)**

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Replication Service.
AffectedElement		Mandatory	Remote Replication Collection.

#### **16.5.28 CIM\_ServiceAffectsElement (Between ReplicationService and ReplicationEntity)**

Associates Replication Service to ReplicationEntity.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 330 describes class CIM\_ServiceAffectsElement (Between ReplicationService and ReplicationEntity).

**Table 330 - SMI Referenced Properties/Methods for CIM\_ServiceAffectsElement (Between ReplicationService and ReplicationEntity)**

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Replication Service.
AffectedElement		Mandatory	Replication Entity.

### 16.5.29 CIM\_ServiceAffectsElement (Between ReplicationService and ReplicationGroup)

Associates Replication Service to Replication Group.

Created By: Extrinsic: CreateGroup

Modified By: Extrinsic: DeleteGroup, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Required if groups are supported.

Table 331 describes class CIM\_ServiceAffectsElement (Between ReplicationService and ReplicationGroup).

**Table 331 - SMI Referenced Properties/Methods for CIM\_ServiceAffectsElement (Between ReplicationService and ReplicationGroup)**

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Replication Service.
AffectedElement		Mandatory	Replication Group.

### 16.5.30 CIM\_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)

Associates a replication group to an instance of SynchronizationAspect.

Created By: Extrinsic: CreateSynchronizationAspect

Modified By: Extrinsic

Deleted By: Extrinsic: ModifySettingsDefineState, ModifyReplicaSynchronization

Requirement: Optional

Table 332 describes class CIM\_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect).

**Table 332 - SMI Referenced Properties/Methods for CIM\_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Storage Element.
SettingData		Mandatory	Synchronization Aspect.

### 16.5.31 CIM\_SettingsDefineState (Between storage object and SynchronizationAspect)

Associates a storage object to an instance of SynchronizationAspect.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 333 describes class CIM\_SettingsDefineState (Between storage object and SynchronizationAspect).

**Table 333 - SMI Referenced Properties/Methods for CIM\_SettingsDefineState (Between storage object and SynchronizationAspect)**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Storage Element.
SettingData		Mandatory	Synchronization Aspect.

#### 16.5.32 CIM\_SharedSecret

Created By: Extrinsic: AddSharedSecret

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 334 describes class CIM\_SharedSecret.

**Table 334 - SMI Referenced Properties/Methods for CIM\_SharedSecret**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
ServiceCreationClassName		Mandatory	Key.
ServiceName		Mandatory	Key.
RemoteID		Mandatory	Key, The identity of the client as known on the remote system.
Secret		Mandatory	A secret.

#### 16.5.33 CIM\_SynchronizationAspect

Experimental. Keeps track of source of a copy operation and point-in-time.

Created By: Extrinsic: CreateElementReplica, CreateListReplica, CreateSynchronizationAspect

Modified By: Extrinsic: ModifyReplicaSynchronization

Deleted By: Extrinsic: ModifyReplicaSynchronization, ModifySettingsDefineState

Requirement: Optional



Table 335 describes class CIM\_SynchronizationAspect.

**Table 335 - SMI Referenced Properties/Methods for CIM\_SynchronizationAspect**

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
SyncType		Mandatory	Type of association between source and target elements. Values: 6: Mirror 7: Snapshot 8: Clone.
ConsistencyEnabled		Conditional	Conditional requirement: Required if groups are supported. Set to true if consistency is enabled.
ElementName		Mandatory	An end user relevant name. The value will be stored in the ElementName property of the created SynchronizationAspect.
ConsistencyType		Conditional	Conditional requirement: Required if group consistency is enabled. Indicates the consistency type used by the groups. Values: 2: Sequential Consistency.
CopyStatus	N	Optional	Describes the status of copy operation. Values: 2: Not Applicable 3: Operation In Progress 4: Operation Completed.
CopyMethodology	N	Optional	Indicates the copy methodology utilized for copying. Values: 2: Implementation decides 3: Full-Copy 4: Incremental-Copy 5: Differential-Copy 6: Copy-On-Write 7: Copy-On-Access 8: Delta-Update 9: Snap-And-Clone.
WhenPointInTime	N	Optional	
SourceElement		Mandatory	Reference to the source element or the source group of a copy operation and/or a point-in-time.

## EXPERIMENTAL

---



---



## **Annex A (informative) SMI-S Information Model**

This standard is based on DMTF's CIM schema, version 2.45.0. The DMTF schema is available in the machine-readable Managed Object Format (MOF) format. DMTF MOFs are simultaneously released both as an "Experimental" and a "Final" version of the schema. This provides developers with early access to experimental parts of the models. Both versions are available at

<http://www.dmtf.org/standards/cim>

Content marked as "Experimental" or "Implemented" may be based on DMTF's Experimental MOFs.



## Annex B (Informative) State Transitions from Storage to File Shares

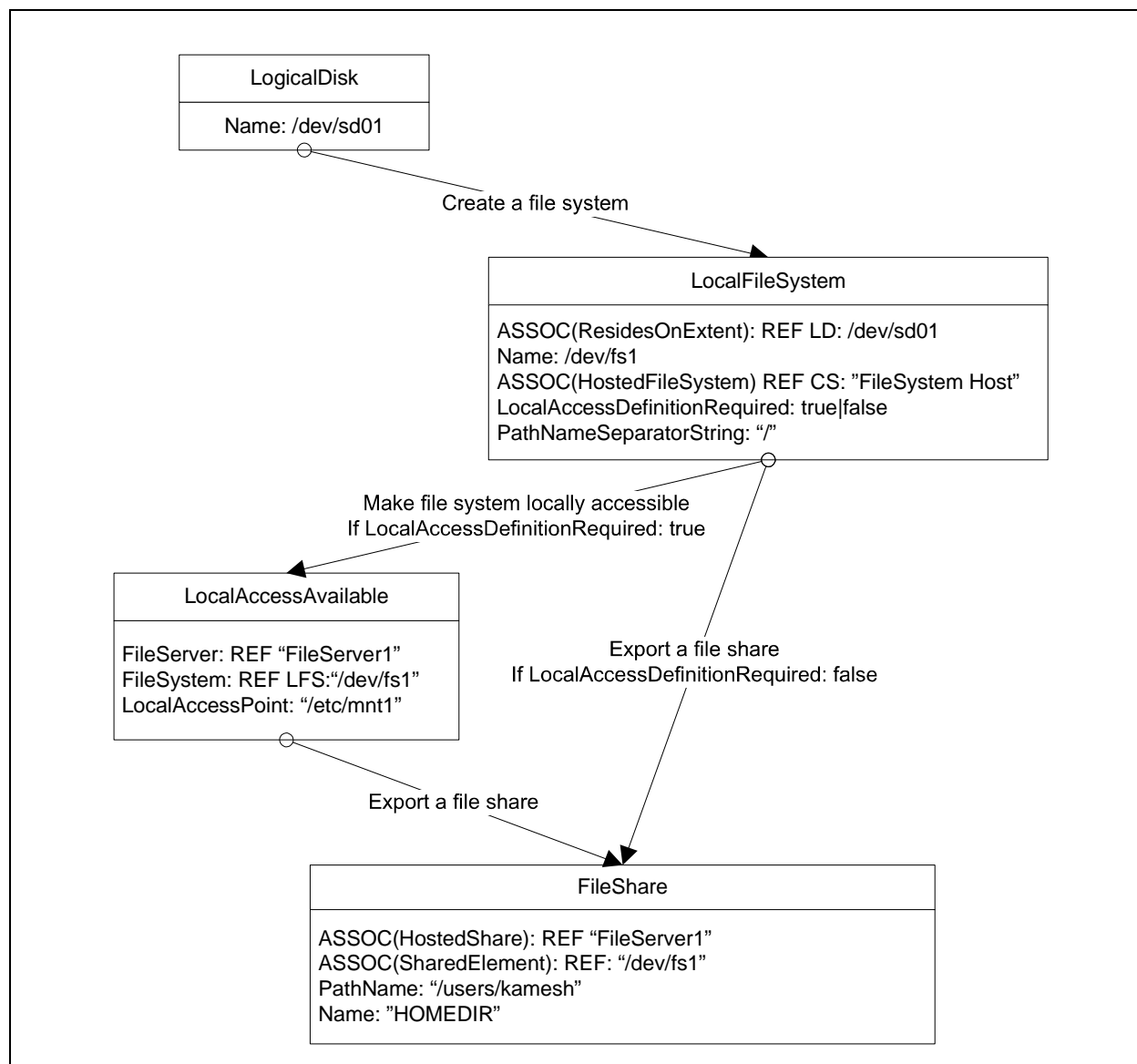
A filesystem is an abstract class that abstractly describes a hierarchical structuring of data into “files” contained within a tree of “directories” with a single “root” directory. A `LocalFileSystem` is a concrete class derived from `FileSystem` that implements it using one or more storage elements in which the storage element(s) has been structured to contain information about multiple files organized into directories as well as the content of these files. This internal organization of a `LocalFileSystem`, viz., what parts represent the components of files, what parts constitute directories, what the names of these files and directories are, how they are organized into a hierarchy, even the representation of the path to a file from the root directory through a sequence of sub-directories etc., is called “metadata” and is stored persistently inside the storage element(s). In addition to metadata, the internal organization contains information about ownership of files and directories, rights of users or other entities to access files and directories, and other attributes of files and directories. This information is sometimes included in metadata, but sometimes referred to independently as “attribute” information and is also stored persistently within the storage element(s). Finally, the contents of files are also stored persistently in the storage element(s).

In filesystem-related profiles, the collection of storage elements used by a `LocalFileSystem` is called a *LogicalDisk(s)*. There are multiple formats (both open and proprietary) for structuring a *LogicalDisk* into a `LocalFileSystem`—how the metadata, attributes, and content are stored and so on—that are referred to as the “type” of the `LocalFileSystem`. The information about the type of the `LocalFileSystem` (and possibly variant versions of the type) is also persistently stored in the *LogicalDisk*. The type of the `LocalFileSystem` in this and related profiles is represented as the “`FileSystemType`”.

**NOTE** The Volume Composition Profile describes how multiple *LogicalDisks* can be merged into a single one. It is assumed that if more than one storage element is used, they are composed into a single *LogicalDisk* using the Volume Composition Profile (see 17 Volume Composition Profile in *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*) or other profile that similarly merges multiple storage elements into a single *LogicalDisk*.

A `LocalFileSystem` is not an autonomous entity but is a hosted component of a larger system, usually a `ComputerSystem`. This is represented using the `HostedFileSystem` association between a `ComputerSystem` and the `LocalFileSystem`. Since the *LogicalDisk* is a `SystemDevice` of a `ComputerSystem`, it is frequently the case that the `LocalFileSystem` will be hosted by the same `ComputerSystem`, but this is not required. It is generally the case that a `LocalFileSystem` will have an independent internal name that may be used to refer to it but it is not necessary that the name be constructed independently of the name of the *LogicalDisk* or the name of the hosting `ComputerSystem`. Some systems require that this internal name be globally unique, but others rely on the uniqueness of the *LogicalDisk*’s name or on other identifiers. In SMI-S, it is a requirement that a `LocalFileSystem` have a unique `Name` property relative to the hosting `ComputerSystem` (`Name` being one of the key properties of the `FileSystem` class).

The process by which an element of storage is finally made available as a `FileShare` is represented by Figure B.1: “State Transitions From *LogicalDisk* to `FileShare`”. The process begins with an unused *LogicalDisk* that is owned by, or has been allocated to, the `ComputerSystem` for this purpose. The operation “Create a filesystem”, converts an unused *LogicalDisk* to a `LocalFileSystem`— Figure B.1: “State Transitions From *LogicalDisk* to `FileShare`” shows the name and the `ComputerSystem` that has a `HostedFileSystem` association to the `LocalFileSystem`. The other details of the `LocalFileSystem` are skipped.



**Figure B.1 - State Transitions From LogicalDisk to FileShare**

Even after a LogicalDisk has been internally organized into a LocalFileSystem, it may not be usable by an operational user. That's because the operational user needs a durable name (for referring to the LocalFileSystem) that is persistently supported by the implementation. There are multiple ways in which this problem has been solved. Since the LocalFileSystem must be hosted by a ComputerSystem and the LocalFileSystem has a unique name, a Uniform Resource Indicator (URI) can be constructed that is relative to the hosting ComputerSystem. However, an operational user needs to use an access path relative to the ComputerSystem that serves files to them (i.e., relative to a File Server), and this may differ from the hosting ComputerSystem.

Traditionally (i.e., before URIs), a LocalFileSystem was assigned a name in a hierarchical name space maintained by the File Server ComputerSystem. This assignment was called "mounting to" the name and the name was called the "mount-point" of the filesystem. For historical and other reasons, the hierarchical name space most commonly used for the purpose was based on the "root filesystem" of the File Server. This allowed a naming convention using "file path names" for objects in the namespace that could be

extended uniformly to the meta-data and content of the mounted filesystem (and would be represented in the SMI Specification as a property of a Capabilities element).

If the hosting ComputerSystem and the File Server ComputerSystem are the same, or can be referenced using a single identifier (for instance in a clustered computer system), or only one File Server can access a LocalFileSystem, it is possible to make the Name of the LocalFileSystem be the same as the mount-point. In that case, the act of “mounting to” the name is accomplished by default when the LocalFileSystem is created. But this does not work for implementations that allow a LocalFileSystem hosted by one ComputerSystem to be assigned differently named mount-points on multiple File Server ComputerSystems. The problem increases in complexity when a File Server can have multiple network identities (through a multiplicity of IP addresses and multiple fully-qualified domain names that map to each IP address).

Traditionally, Unix-based uni-processor systems have not made the Name of the LocalFileSystem the same as the mount-point. But many specialized systems follow such a policy, so whether mounting is not managed explicitly (because it is automatically specified by the name of the LocalFileSystem) or must be managed explicitly is a feature of an implementation.

In addition to specifying a mount-point for a LocalFileSystem, a File Server would also assign system resources needed for working with the LocalFileSystem. These include read and write buffers of appropriate capacity, restrictions on reading or writing (needed for systems that allow multiple mounts of a LocalFileSystem), and other implementation-dependent resources. The specification of these resources are explicitly manageable by some implementations and defaulted by others.

The terms “mount” and “mount-point” are also traditionally used for assigning a remote element (such as a shared file) a name in the local name space of a ComputerSystem. These terms by themselves appeared to be too generic for use in this specification, so the terms used are “make locally accessible” for “mount” and “local access point” for “mount-point”. The resources to be allocated for mounting are specified by “local access settings”.

In SMI-S, a “locally accessible filesystem” is represented by providing an association, LocalAccessAvailableToFS, from the File Server to the LocalFileSystem. In addition to the key reference properties, this association provides the LocalAccessPoint string array property that specifies the “local access point”. Referring back to Figure B.1: “State Transitions From LogicalDisk to FileShare”, the “Make a filesystem locally accessible” operation creates the LocalAccessAvailableToFS association between the File Server and the LocalFileSystem. This operation is supported in the Filesystem Manipulation Profile by providing appropriate parameters to the CreateFileSystem and ModifyFileSystem extrinsic methods. The LocalFileSystem element is not modified, but a new association is created. The LocalAccessPoint property provides the access point (shown in the standard Unix format as “/etc/mnt1”).

**NOTE** The intent behind implementing “Make a filesystem locally accessible” with CreateFileSystem and ModifyFileSystem methods is that it is preferable not to distinguish between implementations that implement a separate “Make Locally Accessible” function from those that do not.

The vendor implements this operation by providing the necessary parameters to the create and modify methods; this has the benefit that the operation does not have to be exposed separately to the management client. However all implementations that support multiple File Servers with independent names to access filesystems must support LocalAccessAvailableToFS as that is the only place where a file-server-specific name for the LocalFileSystem is specified (by the LocalAccessPoint property). A vendor that provides accessibility by default might have a FileSystem.Name property that also functions as a path name from each file server (in one sample implementation), so it is likely that LocalAccessAvailableToFS.LocalAccessPoint would be the same as the LocalFileSystem.Name property. The property LocalFileSystem.LocalAccessDefinitionRequired is required to indicate that this feature is used and that the client must examine that property to understand how a vendor implements this model.

The creation of a file share and its behavior are detailed in the related File Export and File Export Manipulation Profiles. Figure B.1: “State Transitions From LogicalDisk to FileShare” shows the “Export a file share” operation that creates a FileShare and an SharedElement association. The FileShare provides

a name “HOMEDIR” and is hosted by the File Server. The SharedElement association links to the LocalFileSystem and also provides the pathname “/users/kamesh” to the specific user’s home directory.

NOTE Once a LocalFileSystem has been made locally accessible via a File Server, the File Server can share its contents with remote operational users. The contents of such a filesystem can be shared all the way from the root directory at the top of the hierarchy, or the contents of sub-tree below some contained internal directory may be shared, or a specific file contained in the filesystem may be shared. When a directory (root or otherwise) is shared, all files and sub-directories of that directory are automatically also shared recursively. The semantics of sharing an individual file or directory are ultimately controlled by the implementation of the filesystem, so sharing cannot violate the access rules specified internally to the filesystem. In addition to specifying the object (file or directory) to be shared, the File Server may specify the protocol to use for sharing and a correlatable name by which remote users can refer to the shared object—the protocol, the unique server id, and the share name can be used to construct a URI for the shared object. The base URI can be extended to construct a reference URI for files or subdirectories within the shared object.

In SMI-S, there is a FileShare element created to represent the externally accessible share. This element is associated via SharedElement to the LocalFileSystem. The FileShare element will provide the PathName string property that specifies the shared object (the contained file or directory name).