Specification **DSP0117**

STATUS: Preliminary

Copyright © "2000" Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. DMTF specifications and documents may be reproduced for uses consistent with this purpose by members and non-members, if correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release cited should always be noted."

DMTF LDAP Schema for the CIM v2.4 Core Information Model v1.0 May 6, 2002

Abstract

This document presents an LDAP schema for the CIM version 2.4 Core Information Model [3].

Table of Contents

1. Introduction

This document presents an LDAPv3 [1,2,6] schema for the DMTF CIM Core 2.4 Model [3]. Abstract CIM classes are mapped to abstract LDAP classes. Concrete CIM classes are mapped to structural and auxiliary LDAP classes. CIM associations are mapped using a combination of auxiliary classes and structural LDAP classes. The content and structure rules provided here are suggestions and may be modified as needed to support a particular directory structure. Directory administrators do not need to subclass/instantiate all of the classes in this schema. They are free to choose the subset that meets their particular needs.

2. LDAP Mapping Considerations

2.1 Differences from the Core CIM Model

The LDAP schema presented here differs from the core CIM model in that not all classes in the CIM Core model have been mapped in this model. Specifically, the CIM_StatisticalInformation class and its subclasses, the CIM_Statistics class and its subclasses, and the DependencyContext association have not been mapped.

This LDAP schema is not mapped one-to-one, class for class, from CIM. It uses the following approaches:

- Abstract CIM classes (including associations) are mapped to abstract LDAP object classes. This has the side effect that the reference properties of an abstract CIM association are not mapped to attributes.
- Concrete CIM classes are mapped to a trio of LDAP classes:
    - an abstract class, which mirrors the CIM class hierarchy through the LDAP object class hierarchy mechanism
    - an auxiliary class, which allows for the CIM information to be attached to a pre-existing directory object instance
    - a structural class
- CIM associations are mapped according to their cardinality and properties. The cases for mapping associations are explained further in section 2.6

2.2 Changes from previous versions

This version of the LDAP schema has changes to the ABNF to correct errors that have been pointed out by directory implementers. Specifically, the syntax rules are now numeric object identifiers and equality rules have been added for multi-valued attributes.

The first version of this schema lacked the mapping from CIM concrete classes to multiple LDAP classes. Also, the method of naming reference attributes was changed to provide additional clarity and specificity when an instance of a structural LDAP class participates in different associations.

This version no longer requires DIT containment for weak associations. These are just considered a different flavor of one-to-many associations. Because of the use of structural LDAP objects to represent certain associations, cimAssociationInstance is no longer used in this mapping.

Finally, the scheme for the textual identification of the elements of the LDAP schema is changed in this version. Previously, LDAP object classes were identified as cimXXName, where XX was initially derived from the version of CIM from which the LDAP class was mapped. (If the CIM class changed later, XX would change in a subsequent mapping, but not necessarily in alignment with the revised CIM version number.) This was chosen for convenience only. As it leads to misunderstandings regarding synchronization with CIM versions, it has been dropped for a simpler scheme in this and subsequent versions. The new scheme for LDAP object classes is dlmXName for the mapping of the CIM class “Name”, where X is “1” in this version and increased by one each time the CIM class changes and a new LDAP class is produced. For consistency and to avoid confusion, the prefix “dlm” (DMTF LDAP Mapping) is used in the identification of other LDAP schema elements such as attributes. The exceptions to this naming change are the attributes arrayIndex, orderedCimKeys and orderedCimModelPath, which retain their name from the previous mapping version.

2.3 Helper Classes

2.3.1 dlmOtherIdentifyingInfoInstance

CIM defines the concept of an ordered array, which LDAP does not support. In the core CIM model, indexed arrays are only used in two abstract classes (CIM_ComputerSystem and CIM_LogicalDevice) to tie the values of two property arrays together. In the LDAP mapping, these properties are replaced with separate instances of dlmOtherIdentifyingInfoInstance that each contain a single pair of attribute values and are DIT contained by the parent class. The attribute dlmOtherIdentifyingInfo is defined in Section 3.3 and reused here and the attribute arrayIndex is defined as the RDN for this class. Finally, the structure rule is provided as a template to be filled in with structure rule pointers to structural rules defined for concrete sub-classes of dlm1ComputerSystem and dlm1LogicalDevice.

```
   ( 1.3.6.1.4.1.412.100.1.2.5 NAME 'arrayIndex'
     DESC 'the index of this child'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     EQUALITY caseIgnoreMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.101 NAME 'dlmIdentifyingDescription'
     DESC 'A free-form string providing explanation and
           details behind the entries in the dlmOtherIdentifyingInfo
           attribute.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.92 NAME
'dlmOtherIdentifyingInfoInstance'
     DESC 'helper class to tie indexed arrays in core model together'
     SUP top
     MUST ( arrayIndex )
     MAY ( dlmOtherIdentifyingInfo $ dlmIdentifyingDescription )
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.9 NAME
'dlmOtherIdentifyingInfoInstanceNameForm'
     OC dlmOtherIdentifyingInfoInstance
     MUST ( arrayIndex )
   )

   ( <core-sr-9> NAME 'dlmOtherIdentifyingInfoInstanceStructureRule'
     FORM dlmOtherIdentifyingInfoInstanceNameForm
   )
```

2.4 Naming considerations

To support naming in the LDAP mapping of the core schema, the attribute orderedCimKeys is defined, to provide the RDN for directory implementations.

```
   ( 1.3.6.1.4.1.412.100.1.2.1 NAME 'orderedCimKeys'
     DESC 'The model path for the instance (without propagated
           keys). May be used as an RDN'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
     EQUALITY octetStringMatch
   )
```

The value of this attribute is constructed by ordering the CIM keys [formatted as "<className>.<key>=<value>[,<key>=<value>]*"] of the object in the US-ASCII collation order of the property names. For an instance with propagated keys in the CIM namespace, the value of this attribute takes one of two forms: either it includes all of the instance's keys, or it includes only the non-propagated ones. Ordinarily the propagated keys will be included when the DIT hierarchy in which an instance appears does not reflect the CIM naming hierarchy represented by the propagation of keys via weak associations. When the DIT hierarchy does mirror the CIM naming hierarchy, the propagated keys are unnecessary and may be omitted. By consulting the CIM schema, a directory client can tell whether propagated keys may have been included.

In a previous version of this specification, the value of orderedCimKeys never included propagated keys. A second attribute, orderedCimModelPath, was used when propagated keys were required. Now that orderedCimKeys includes the case where propagated keys are included, orderedCimModelPath can be marked as "obsolete".

```
   ( 1.3.6.1.4.1.412.100.1.2.2 NAME 'orderedCimModelPath'
     DESC 'The model path for the instance (with propagated keys). May
           be used as an RDN'
     OBSOLETE
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  SINGLE-VALUE
     EQUALITY octetStringMatch
   )
```

2.5 Syntax Conversions

This section discusses specific conversions needed for the core schema. Other mappings may define additional conversion procedures.

2.5.1 CIM String and LDAP DirectoryString

Strings in CIM are stored as UCS-2 characters, while LDAP DirectoryStrings are stored in UTF-8 format. See [5] for more information on how to convert between these formats.

2.5.2 CIM DateTime and LDAP GeneralizedTime

CIM DateTime is used to store both timestamps and intervals in UCS-2. LDAP GeneralizedTime stores timestamps in a subset of UTF-8. See [5] for more information on how to convert between these formats.

2.6 Associations

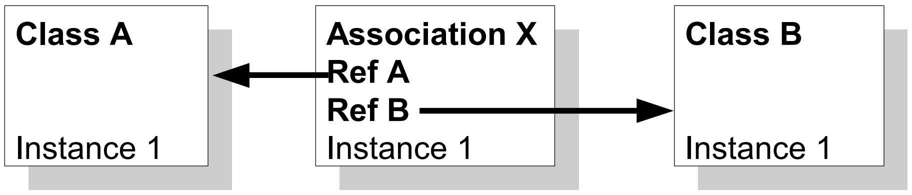
2.6.1 Types of associations

The various types of associations that may be encountered in a CIM model can be categorized as follows:

- One-to-one
- One-to-one with properties
- One-to-many
- One-to-many with properties
- Many-to-many
- Many-to-many with properties.

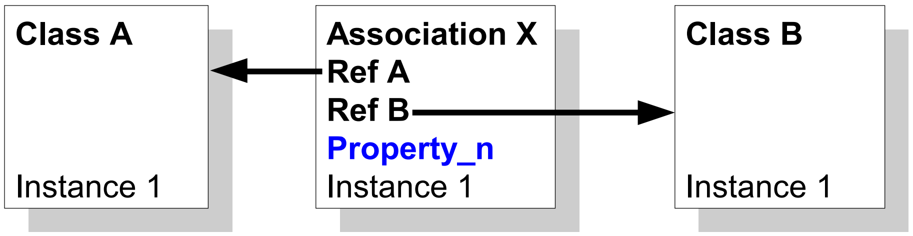
Each of these can be illustrated as follows

2.6.1.1 One-to-One



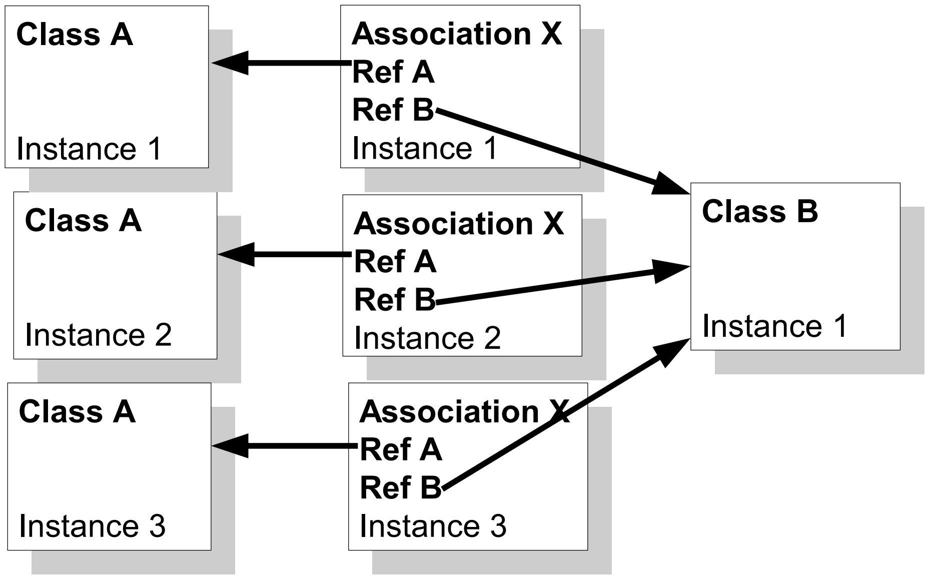
An instance of class A or class B can be referenced by no more than one instance of association X.

2.6.1.2 One-to-One with properties



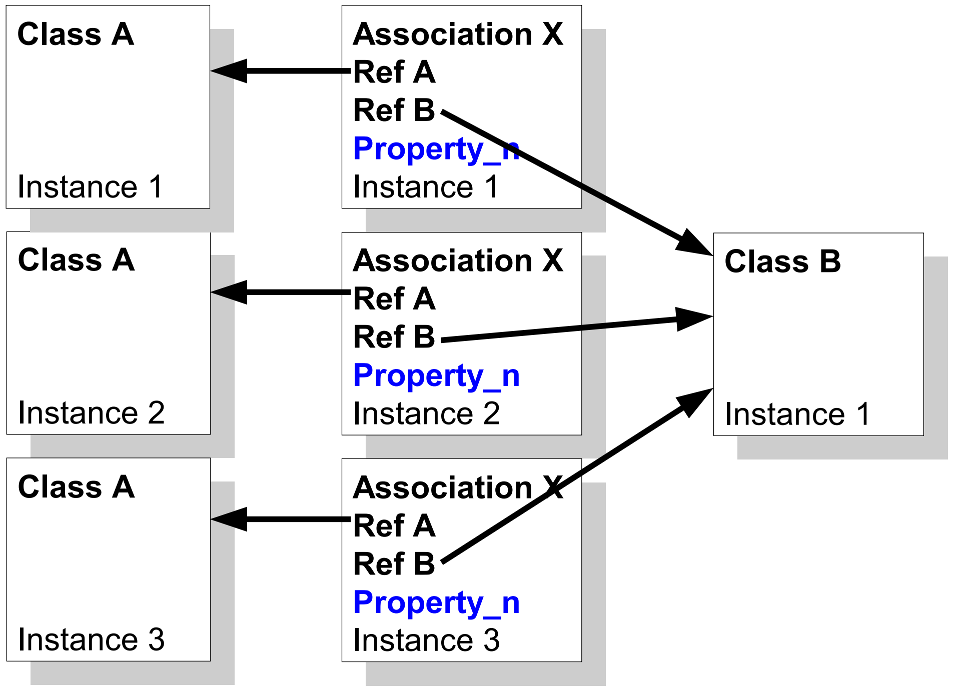
An instance of class A or class B can be referenced by no more than one instance of association X. The association instance has one or more properties that characterize the relationship. For example, if the association represented a serial link, it could have a property stating the speed of the link.

2.6.1.3 One-to-many



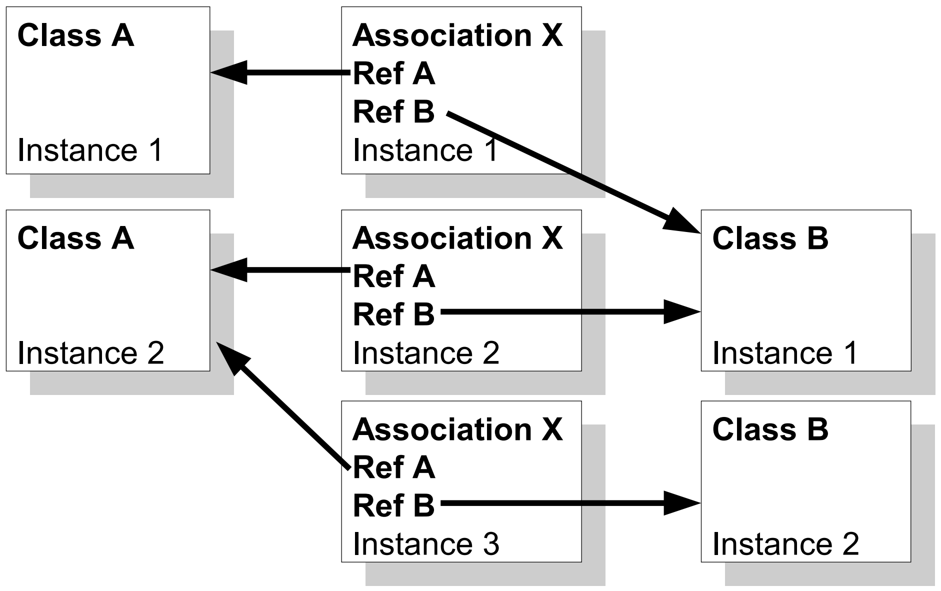
A single instance of class A may be referenced by no more than on instance of association X. A single instance of class B may be referenced by any number of instances of association X. Note: Ref B is not an array. There is an instance of association X for every A, B pair that is associated.

2.6.1.4 One-to-many with properties

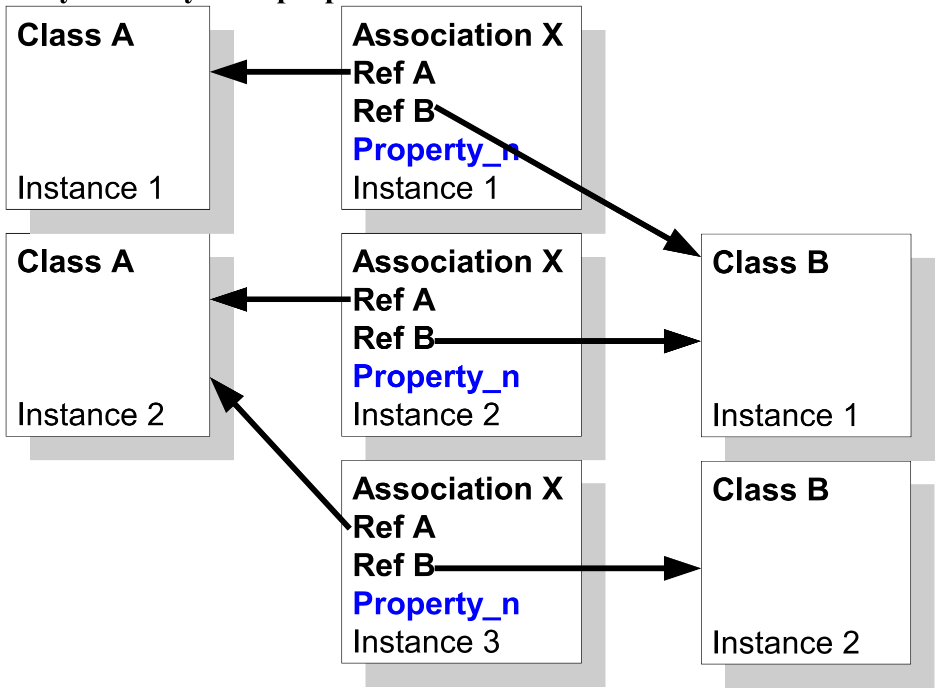


Each instance of the association has one or more properties that characterize the relationship. For example a network switch may connect to many workstations (assume each workstation can only support a single connection) is a star topology. Each link can be half or full duplex. A property contained in the association class could be used to model this.

2.6.1.5 Many-to-many



2.6.1.6 Many-to-many with properties

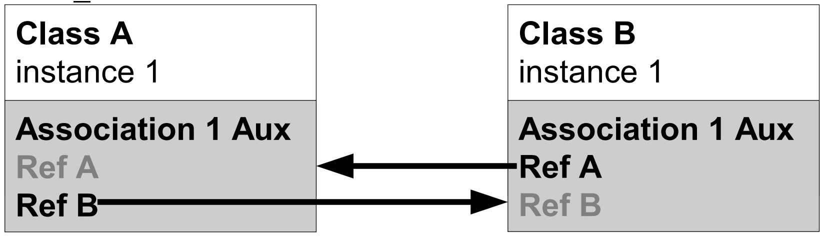


2.6.2 Mapping Associations

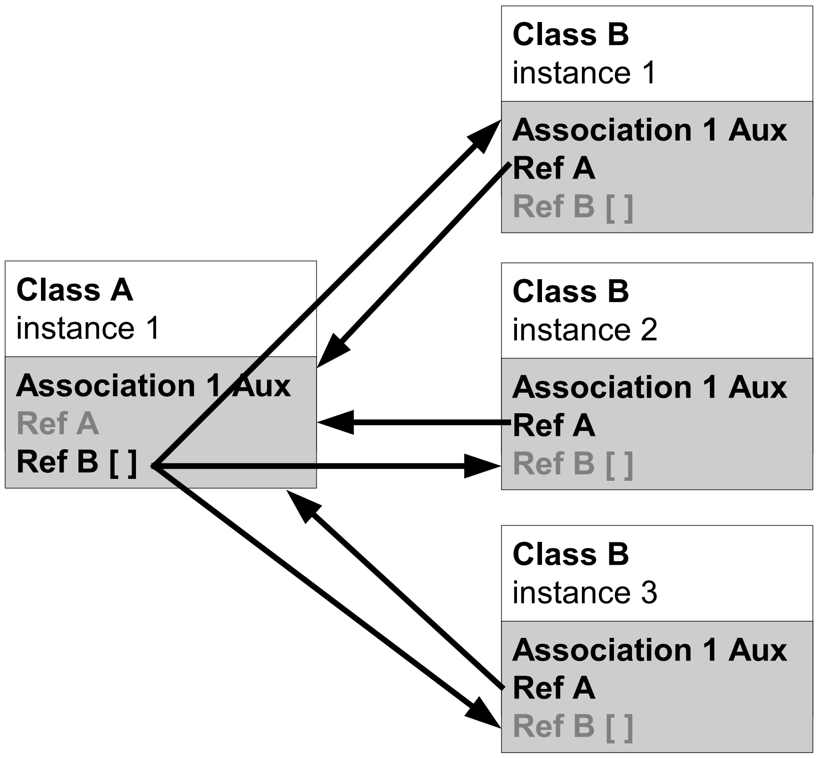
There are three distinct models used for mapping non-abstract associations in this document. Each has its own conventions for how such associations are not only mapped, but also implemented in the directory. The following sections discuss these conventions. Since all associations have referential properties, the term “additional properties” in the remainder of this section refers to non-referential properties. The approach in Section 2.6.2.3 may also be used to map associations with no additional properties and 1-to-1 or 1-to-many associations with additional properties if necessary.

2.6.2.1 No Additional Properties

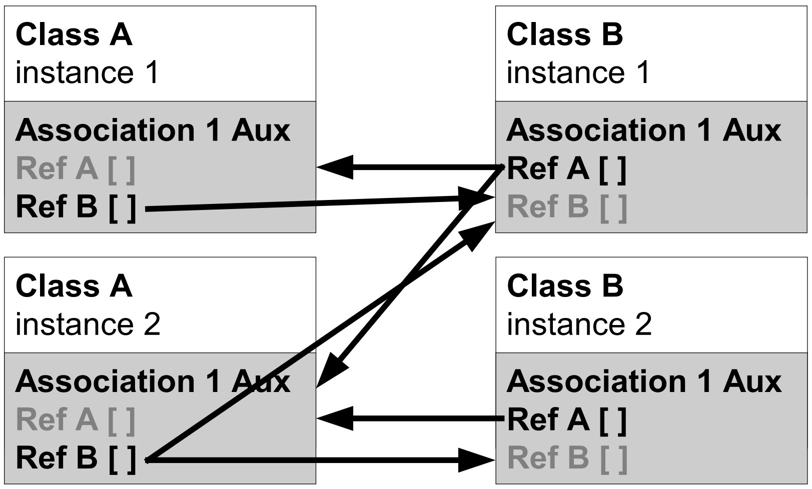
If a non-abstract association has no additional properties, then it is mapped as an auxiliary class that contains both referential properties as optional DN attributes. This class is attached to all structural objects that participate in the association, with the proper attribute being populated for that particular structural object. An example of this type of association is CIM_HostedService.



One-to-One



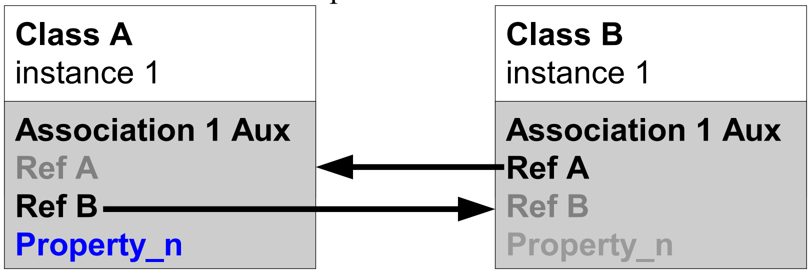
One-to-many



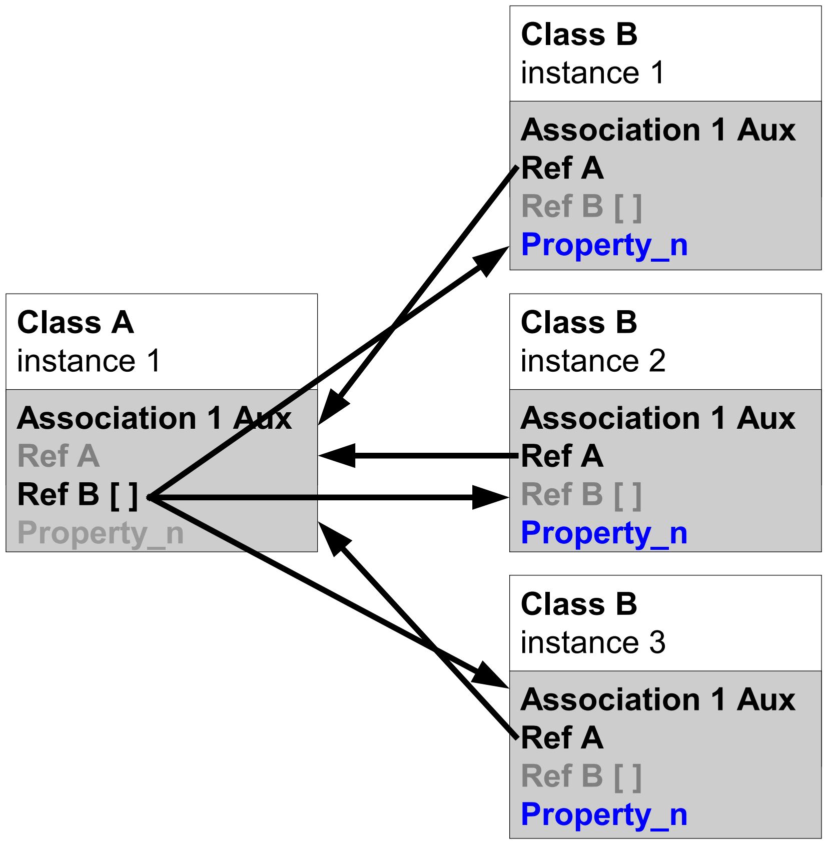
Many-to-Many

2.6.2.2 Additional Properties, 1-to-1 or 1-to-many

If a non-abstract association has additional properties, then the mapping is determined by the cardinality of the referential properties. In the case of a 1-to-1 or 1-to-many cardinality, the association is mapped as an auxiliary class with all properties mapped as optional attributes. The auxiliary class is attached to all structural objects participating in the association, with the referential attribute set appropriately. The additional properties are set for the auxiliary class that is attached to the many side of a 1-to-many association. The core model does not have an example of this class of association.



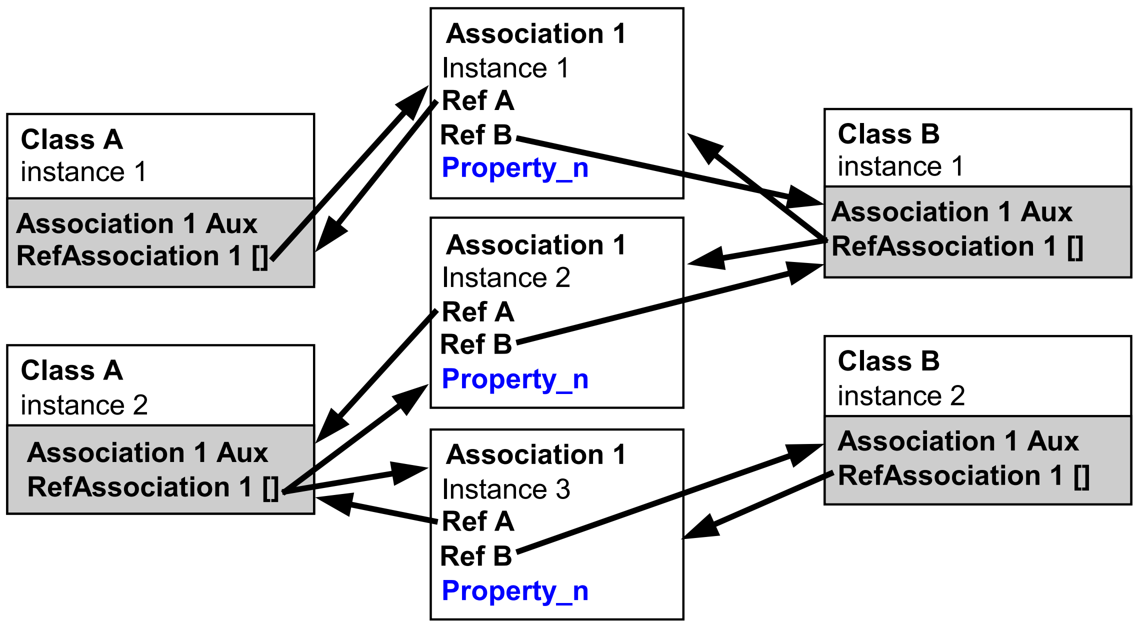
One-to-one with properties



One-to-many with properties

2.6.2.3 Additional Properties, many-to-many

For a non-abstract association with additional properties and a many-to-many cardinality, the most flexible mapping is to use a structural LDAP class that contains all properties of the association as optional attributes. Since this is a separate object in the directory, helper auxiliary classes are provided that are attached to the structural objects in the directory participating in the association. These helper classes contain a single optional attribute that points to the particular instance of the association that this object participates in. There is an instance of the structural class for every instance of the association. An example of this type of association is CIM_ServiceServiceDependency.



Many-to-many with properties

This approach may also be used to map associations with no additional properties and 1-to-1 or 1-to-many associations with additional properties if necessary.

2.6.2.4 Weak associations

Weak associations are one-to-one, or one-to-many and may or may not have properties. They may and should be mapped using the appropriate mechanism above. Weak implies additional semantics that maps well to DIT containment. Instances of weak classes may but are not required to be stored as children of the entries they are weak to. When such storage is used, application may utilize this to optimize association traversal.

3. Class Definitions

3.1 ManagedElement

This abstract class provides a base for non-association classes in CIM. Its addition is one of the major changes between CIM v2.2 and CIM v2.3.

```
( 1.3.6.1.4.1.412.100.2.2.103 NAME 'dlmCaption'
  DESC 'The Caption property is a short textual
        description (oneline string) of the object.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.104 NAME 'dlmDescription'
  DESC 'The Description property provides a textual
        description of the object.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.1 NAME 'dlm1ManagedElement'
  DESC 'ManagedElement is an abstract class that provides
```

```
      a common superclass (or top of the inheritance tree)
       for the non-association classes in the CIM Schema.'
 SUP top ABSTRACT
 MAY ( dlmCaption $ dlmDescription $ orderedCimModelPath
      $ orderedCimKeys )
)
```

3.2 ManagedSystemElement

This is the base class for the system element hierarchy. Any distinguishable component of a system is a candidate for inclusion in this class. Examples of this are software components, such as files and devices, such as disk drives and controllers, and physical components such as chips and cards.

```
( 1.3.6.1.4.1.412.100.2.2.105 NAME 'dlmInstallDate'
  DESC 'A datetime value indicating when the object was
        installed. A lack of a value does not indicate that
        the object is not installed.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.106 NAME 'dlmName'
  DESC 'The Name property defines the label by which the
        object is known. When subclassed, the Name property
        can be overridden to be a Key property.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.107 NAME 'dlmStatus'
  DESC 'A string indicating the current status of the
        object. Various operational and non-operational
        statuses are defined. Operational statuses are "OK",
        "Degraded", "Stressed" and "Pred Fail". "Stressed"
        indicates that the Element is functioning, but needs
        attention. Examples of "Stressed" states are overload,
       overheated, etc. The condition "Pred Fail" (failure
        predicted) indicates that an Element is functioning
        properly but predicting a failure in the near future.
        An example is a SMART-enabled hard drive.
        Non-operational statuses can also be specified. These
        are "Error", "NonRecover", "Starting", "Stopping",
        "Service", "No Contact" and "Lost Comm". "NonRecover"
        indicates that a non-recoverable error has occurred.
        "Service" describes an Element being configured,
        maintained, cleaned, or otherwise administered. This
        status could apply during mirror-resilvering of a
        disk, reload of a user permissions list, or other
        administrative task. Not all such work is on-line, yet
       the Element is neither "OK" nor in one of the other
        states. "No Contact" indicates that the current
        instance of the monitoring system has knowledge of
        this Element but has never been able to establish
        communications with it. "Lost Comm" indicates that the
       ManagedSystemElement is known to exist and has been
        contacted successfully in the past, but is currently
        unreachable.  Value Mappings are "OK", "Error",
        "Degraded", "Unknown", "Pred Fail", "Starting",
        "Stopping", "Service", "Stressed", "NonRecover", "No
        Contact", "Lost Comm"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.2 NAME 'dlm1ManagedSystemElement'
  DESC 'ManagedSystemElement is the base class for the
        System Element hierarchy. Membership Criteria: Any
        distinguishable component of a System is a candidate
        for inclusion in this class. Examples: software
        components, such as files; and devices, such as disk
        drives and controllers, and physical components such
        as chips and cards.'
  SUP dlm1ManagedElement ABSTRACT
  MAY ( dlmInstallDate $ dlmName $ dlmStatus )
)
```

3.3 PhysicalElement

This class acts as the base class for any component of a system that has a distinct physical identity. Instances of this class can be defined in terms of labels that can be physically attached to the object. All processes, files, and logical devices are considered not to be physical elements. For example, it is not possible to attach a label to a modem. It is only possible to attach a label to the card that implements the modem. The same card could also implement a LAN adapter. This is an example of a single physical element (the card) hosting more than one logical device.

```
( 1.3.6.1.4.1.412.100.2.2.108 NAME 'dlmCreationClassName'
  DESC 'CreationClassName indicates the name of the class
       or the subclass used in the creation of an instance.
        When used with the other key properties of this class,
       this property allows all instances of this class and
        its subclasses to be uniquely identified.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.109 NAME 'dlmManufactureDate'
  DESC 'Date that this PhysicalElement was manufactured.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.110 NAME 'dlmManufacturer'
  DESC 'The name of the organization responsible for
        producing the PhysicalElement. This may be the entity
        from whom the Element is purchased, but this is not
        necessarily true. The latter information is contained
        in the Vendor property of Product.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.111 NAME 'dlmModel'
  DESC 'The name by which the PhysicalElement is
        generally known.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
)
```

```
( 1.3.6.1.4.1.412.100.2.2.112 NAME 'dlmOtherIdentifyingInfo'
  DESC 'OtherIdentifyingInfo captures additional data,
        beyond asset tag information, that could be used to
        identify a Physical Element. One example is bar code
        data associated with an Element that also has an asset
       tag. Note that if only bar code data is available and
        is unique/able to be used as an Element key, this
        property would be NULL and the bar code data used as
        the class key, in the Tag property.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.113 NAME 'dlmPartNumber'
  DESC 'The part number assigned by the organization
        responsible for producing or manufacturing the
        PhysicalElement.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.114 NAME 'dlmPoweredOn'
  DESC 'Boolean indicating that the PhysicalElement is
        powered on (TRUE), or is currently off (FALSE).'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.115 NAME 'dlmSKU'
  DESC 'The stock keeping unit number for this
        PhysicalElement.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.116 NAME 'dlmSerialNumber'
  DESC 'A manufacturer-allocated number used to identify
        the Physical Element.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.117 NAME 'dlmTag'
  DESC 'An arbitrary string that uniquely identifies the
        Physical Element and serves as the Element"s key.  The
       Tag property can contain information such as asset tag
        or serial number data. The key for PhysicalElement is
        placed very high in the object hierarchy in order to
        independently identify the hardware/entity, regardless
       of physical placement in or on Cabinets, Adapters, etc.
        For example, a hotswappable or removeable component
        may be taken from its containing (scoping) Package and
       be temporarily unused.  The object still continues to
        exist - and may even be inserted into a different
        scoping container.  Therefore, the key for Physical
        Element is an arbitrary string and is defined
        independently of any placement or location-oriented
        hierarchy.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
)
```

```
( 1.3.6.1.4.1.412.100.2.2.118 NAME 'dlmVersion'
  DESC 'A string indicating the version of the
        PhysicalElement.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.3 NAME 'dlm1PhysicalElement'
  DESC 'Subclasses of PhysicalElement define any
        component of a System that has a distinct physical
        identity. Instances of this class can be defined in
        terms of labels that can be physically attached to the
       object. All Processes, Files, and LogicalDevices are
        considered not to be Physical Elements. For example,
        it is not possible to attach a label to a modem. It is
       only possible to attach a label to the card that
        implements the modem. The same card could also
        implement a LAN  adapter. These are tangible Managed
        System Elements (usually actual hardware items) that
        have a physical manifestation of some sort. A Managed
        System Element is not necessarily a discrete
        component. For example, it is possible for a single
        Card (which is a type of Physical Element) to host
        more than one Logical Device. The card would be
        represented by a single Physical Element associated
        with multiple Logical Devices.'
  SUP dlm1ManagedSystemElement ABSTRACT
  MAY ( dlmCreationClassName $ dlmManufactureDate $
        dlmManufacturer $ dlmModel $ dlmOtherIdentifyingInfo $
       dlmPartNumber $ dlmPoweredOn $ dlmSKU $ dlmSerialNumber $
       dlmTag $ dlmVersion )
)
```

3.4 LogicalElement

This class is the base class for all the components of a system that represent abstract system components, such as files, processes, or system capabilities as logical devices.

```
( 1.3.6.1.4.1.412.100.2.1.3.4 NAME 'dlm1LogicalElement'
  DESC 'LogicalElement is a base class for all the
        components of a System that represent abstract system
        components, such as Files, Processes, or system
        capabilities in the form of Logical Devices.'
  SUP dlm1ManagedSystemElement ABSTRACT
)
```

3.5 System

This class is a logical element that aggregates an enumerable set of managed system elements and operates as a functional whole. Within any particular subclass of system, there is a well-defined list of managed system element classes whose instances must be aggregated.

```
( 1.3.6.1.4.1.412.100.2.2.119 NAME 'dlmNameFormat'
  DESC 'The System object and its derivatives are Top
        Level Objects of CIM. They provide the scope for
        numerous components. Having unique System keys is
        required. A heuristic can be defined in individual
```

```
             System subclasses to attempt to always generate the
             same System Name Key. The NameFormat property
             identifies how the System name was generated, using
             the subclass" heuristic.'
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
     )

     ( 1.3.6.1.4.1.412.100.2.2.120 NAME 'dlmPrimaryOwnerContact'
       DESC 'A string that provides information on how the
             primary system owner can be reached (e.g. phone
             number, email address, ...).'
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
     )

     ( 1.3.6.1.4.1.412.100.2.2.121 NAME 'dlmPrimaryOwnerName'
       DESC 'The name of the primary system owner.'
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
     )

     ( 1.3.6.1.4.1.412.100.2.2.122 NAME 'dlmRoles'
       DESC 'An array (bag) of strings that specify the roles
             this System plays in the IT-environment. Subclasses of
            System may override this property to define explicit
             Roles values. Alternately, a Working Group may
             describe the heuristics, conventions and guidelines
             for specifying Roles. For example, for an instance of
             a networking system, the Roles property might contain
             the string, "Switch" or "Bridge".'
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
       EQUALITY caseExactMatch
     )

     ( 1.3.6.1.4.1.412.100.2.1.3.5 NAME 'dlm1System'
       DESC 'A System is a LogicalElement that aggregates an
             enumerable set of Managed System Elements. The
             aggregation operates as a functional whole. Within any
            particular subclass of System, there is a well-defined
             list of Managed System Element classes whose instances
            must be aggregated.'
       SUP dlm1LogicalElement ABSTRACT
       MAY ( dlmCreationClassName $ dlmName $ dlmNameFormat $
             dlmPrimaryOwnerContact $ dlmPrimaryOwnerName $
             dlmRoles )
     )
```

3.6 ComputerSystem

This class is derived from System and represents a special collection of managed system elements that provide compute capabilities. Thus, it serves as aggregation point to associate one or more of the following elements: file systems, operating systems, processors and memory (volatile and/or non-volatile storage).

```
     ( 1.3.6.1.4.1.412.100.2.2.123 NAME 'dlmDedicated'
       DESC 'Enumeration indicating whether the ComputerSystem
            is a special-purpose System (ie, dedicated to a
             particular use), versus being "general purpose". For
             example, one could specify that the System is
```

```
        dedicated to "Print" (value=11) or acts as a "Hub"
        (value=8).  Values are 0="Not Dedicated",
        1="Unknown", 2="Other", 3="Storage", 4="Router",
        5="Switch", 6="Layer 3 Switch", 7="Central Office
        Switch", 8="Hub", 9="Access Server", 10="Firewall",
        11="Print", 12="I/O", 13="Web Caching",
        14="Management"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  EQUALITY integerMatch
)

( 1.3.6.1.4.1.412.100.2.1.3.6 NAME 'dlm1ComputerSystem'
  DESC 'A class derived from System that is a special
        collection of ManagedSystemElements. This collection
        provides compute capabilities and serves as
        aggregation point to associate one or more of the
        following elements: FileSystem, OperatingSystem,
        Processor and Memory (Volatile and/or NonVolatile
        Storage).'
  SUP dlm1System ABSTRACT
  MAY ( dlmDedicated $ dlmNameFormat )
)   )
```

3.7 AdminDomain

This abstract class represents a special grouping of MSEs that are all administered by the same user or group of users.

```
( 1.3.6.1.4.1.412.100.2.1.3.93 NAME 'dlm1AdminDomain'
  DESC 'This is a special grouping of ManagedSystemElements that
      are all administered by the same user or group of users.
      It serves as an aggregation point to associate one or more
      of the following elements: network devices, such as
      routers and switches, servers, and other resources that
      can be accessed by end systems. This grouping of devices
      plays an essential role in ensuring that the same
      administrative POLICY is applied to all of the devices
      in the grouping.'
  SUP dlm1System ABSTRACT
)
```

3.8 LogicalDevice

This class represents an abstraction or emulation of a hardware entity that may or may not be realized in physical hardware. Any characteristics of a logical device that are used to manage its operation or configuration are contained in, or associated with, this object.

```
( 1.3.6.1.4.1.412.100.2.2.124 NAME 'dlmAdditionalAvailability'
  DESC 'Additional availability and status of the Device,
       beyond that specified in the Availability property. The
       Availability property denotes the primary status and
        availability of the Device. In some cases, this will
        not be sufficient to denote the complete status of the
       Device.  In those cases, the AdditionalAvailability
        property can be used to provide further information.
        For example, a Device"s primary Availability may be
        "Off line" (value=8), but it may also be in a low
        power state (AdditonalAvailability value=14), or the
        Device could be running Diagnostics (Additional
        Availability value=5, "In Test").  Values are
        1="Other", 2="Unknown", 3="Running/Full Power",
        4="Warning", 5="In Test", 6="Not Applicable", 7="Power
       Off", 8="Off Line", 9="Off Duty", 10="Degraded",
        11="Not Installed", 12="Install Error", 13="Power Save
       - Unknown", 14="Power Save - Low Power Mode", 15="Power
       Save - Standby", 16="Power Cycle", 17="Power Save -
        Warning", 18="Paused", 19="Not Ready", 20="Not
        Configured", 21="Quiesced"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  EQUALITY integerMatch
)

( 1.3.6.1.4.1.412.100.2.2.125 NAME 'dlmAvailability'
  DESC 'The primary availability and status of the
        Device. (Additional status information can be
        specified using the AdditionalAvailability array
        property.) For example, the Availability property
        indicates that the Device is running and has full
        power (value=3), or is in a warning (4), test (5),
        degraded (10) or power save state (values 13-15 and
        17). Regarding the Power Save states, these are
        defined as follows: Value 13 ("Power Save - Unknown\
        Values are 1="Other", 2="Unknown", 3="Running/Full
        Power", 4="Warning", 5="In Test", 6="Not Applicable",
        7="Power Off", 8="Off Line", 9="Off Duty",
        10="Degraded", 11="Not Installed", 12="Install Error",
       13="Power Save - Unknown", 14="Power Save - Low Power
        Mode", 15="Power Save - Standby", 16="Power Cycle",
        17="Power Save - Warning", 18="Paused", 19="Not
        Ready", 20="Not Configured", 21="Quiesced"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.126 NAME 'dlmDeviceID'
  DESC 'An address or other identifying information to
        uniquely name the LogicalDevice.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.127 NAME 'dlmErrorCleared'
  DESC 'ErrorCleared is a boolean property indicating
        that the error reported in LastErrorCode is now
        cleared.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.128 NAME 'dlmErrorDescription'
  DESC 'ErrorDescription is a free-form string supplying
        more information about the error recorded in
        LastErrorCode, and information on any corrective
        actions that may be taken.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
)
```

```
( 1.3.6.1.4.1.412.100.2.2.129 NAME 'dlmLastErrorCode'
  DESC 'LastErrorCode captures the last error code
        reported by the LogicalDevice.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.130 NAME 'dlmMaxQuiesceTime'
  DESC 'Maximum time in milliseconds, that a Device can
        run in a "Quiesced" state. A Device"s state is defined
       in its Availability and Additional Availability
        properties, where "Quiesced" is conveyed by the value
        21. What occurs at the end of the time limit is
        device-specific. The Device may unquiesce, may offline
       or take other action. A value of 0 indicates that a
        Device can remain quiesced indefinitely. The value is
        considered to be MilliSeconds.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.131 NAME 'dlmPowerManagementCapabilities'
  DESC 'Indicates the specific power-related capabilities
       of a LogicalDevice. The array values, 0="Unknown",
        1="Not Supported" and 2="Disabled" are
        self-explanatory. The value, 3="Enabled" indicates
        that the power management features are currently
        enabled but the exact feature set is unknown or the
        information is unavailable. "Power Saving Modes
        Entered Automatically" (4) describes that a Device can
       change its power state based on usage or other
        criteria. "Power State Settable" (5) indicates that
        the SetPowerState method is supported. "Power Cycling
        Supported" (6) indicates that the SetPowerState method
       can be invoked with the PowerState input variable set
        to 5 ("Power Cycle"). "Timed Power On Supported" (7)
        indicates that the SetPowerState method can be invoked
       with the Power State input variable set to 5 ("Power
        Cycle")  Values are 0="Unknown", 1="Not Supported",
        2="Disabled", 3="Enabled", 4="Power Saving Modes
        Entered Automatically", 5="Power State Settable",
        6="Power Cycling Supported", 7="Timed Power On
        Supported"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  EQUALITY integerMatch
)

( 1.3.6.1.4.1.412.100.2.2.132 NAME 'dlmPowerManagementSupported'
  DESC 'Boolean indicating that the Device can be power
        managed - ie, put into a power save state. This
        boolean does not indicate that power management
        features are currently enabled, or if enabled, what
        features are supported. Refer to the
        PowerManagementCapabilities array for this
        information. If this boolean is false, the integer
        value 1, for the string, "Not Supported", should be
        the only entry in the PowerManagementCapabilities
        array.'
```

```
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.133 NAME 'dlmPowerOnHours'
    DESC 'The number of consecutive hours that this Device
          has been powered, since its last power cycle. The
          value is considered to be Hours.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.134 NAME 'dlmStatusInfo'
    DESC 'StatusInfo is a string indicating whether the
          Logical Device is in an enabled (value = 3), disabled
          (value = 4) or some other (1) or unknown (2) state. If
         this property does not apply to the LogicalDevice, the
          value, 5 ("Not Applicable”).  Values are 1="Other",
          2="Unknown", 3="Enabled", 4="Disabled", 5="Not
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.135 NAME 'dlmTotalPowerOnHours'
    DESC 'The total number of hours that this Device has
          been powered. The value is considered to be Hours.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.1.3.7 NAME 'dlm1LogicalDevice'
    DESC 'An abstraction or emulation of a hardware entity,
         that may or may not be Realized in physical hardware.
          Any characteristics of a LogicalDevice that are used
          to manage its operation or configuration are contained
         in, or associated with, the LogicalDevice object.
          Examples of the operational properties of a Printer
          would be paper sizes supported, or detected errors.
          Examples of the configuration properties of a Sensor
          Device would be threshold settings. Various
          configurations could exist for a LogicalDevice. These
          configurations could be contained in Setting objects
          and associated with the LogicalDevice.'
    SUP dlm1LogicalElement ABSTRACT
    MAY ( dlmAdditionalAvailability $ dlmAvailability $
          dlmCreationClassName $ dlmDeviceID $ dlmErrorCleared $
         dlmErrorDescription $ dlmLastErrorCode $ dlmMaxQuiesceTime $
          dlmPowerManagementCapabilities $
          dlmPowerManagementSupported $ dlmPowerOnHours $
          dlmStatusInfo $ dlmTotalPowerOnHours )
  )
```

3.9 Service

This class represents a Logical Element that contains the information necessary to represent and manage the functionality provided by a device and/or software feature. A service is a general-purpose object to configure and manage the implementation of functionality. It is not the functionality itself.

```
  ( 1.3.6.1.4.1.412.100.2.2.136 NAME 'dlmStartMode'
    DESC 'StartMode is a string value indicating whether
```

```
        the Service is automatically started by a System,
        Operating System, etc. or only started upon request.
        Value Mapping are "Automatic", "Manual"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.2.137 NAME 'dlmStarted'
  DESC 'Started is a boolean indicating whether the
        Service has been started (TRUE), or stopped (FALSE).'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.8 NAME 'dlm1Service'
  DESC 'A Service is a Logical Element that contains the
        information necessary to represent and manage the
        functionality provided by a Device and/or
        SoftwareFeature. A Service is a general-purpose object
       to configure and manage the implementation of
        functionality.  It is not the functionality itself.'
  SUP dlm1LogicalElement ABSTRACT
  MAY ( dlmCreationClassName $ dlmName $ dlmStartMode $
        dlmStarted )
)
```

3.10 ServiceAccessPoint

This class represents the ability to use or invoke a service. Access points represent that a service is made available to other entities for use.

```
( 1.3.6.1.4.1.412.100.2.1.3.9 NAME 'dlm1ServiceAccessPoint'
  DESC 'ServiceAccessPoint represents the ability to
        utilize or invoke a Service.  Access points represent
        that a Service is made available to other entities for
       use.'
  SUP dlm1LogicalElement ABSTRACT
  MAY ( dlmCreationClassName $ dlmName )
)
```

3.11 Collection

This abstract class provides a common superclass for classes that represent collections of managed elements.

```
( 1.3.6.1.4.1.412.100.2.1.3.10 NAME 'dlm1Collection'
  DESC 'Collection is an abstract class that provides a
        common superclass for data elements that represent
        collections of ManagedElements and its subclasses.'
  SUP dlm1ManagedElement ABSTRACT
)
```

3.12 CollectionOfMSEs

This object allows the grouping of ManagedSystemElement objects for associating settings and configurations. It is abstract to require further definition and semantic refinement in subclasses. As this object does not carry any state or status information, it only represents a grouping or 'bag' of elements. So, it is incorrect to subclass groups that

have state/status from this class - an example is RedundancyGroup (which is correctly subclassed from LogicalElement).
Collections typically aggregate 'like' objects, and represent an optimization. Without collections, one is forced to define individual associations, to tie settings and configuration objects to individual ManagedSystemElements. There may be much duplication in assigning the same setting to multiple objects. In addition, using this object allows the determination that the setting and configuration associations are indeed the same for the collection's members. This information would otherwise be obtained by defining the collection in a proprietary way, and then querying the associations to determine if the collection set is completely covered.

```
( 1.3.6.1.4.1.412.100.2.2.138 NAME 'dlmCollectionID'
  DESC 'The identification of the Collection object. When
       subclassed, the CollectionID property can be overridden
       to be a Key property.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.11 NAME 'dlm1CollectionOfMSEs'
  DESC 'The CollectionOfMSEs object allows the grouping
        of Managed SystemElements for the purposes of
        associating Settings and Configurations. It is
        abstract to require further definition and semantic
        refinement in subclasses. The CollectionOfMSEs object
        does not carry any state or status information, but
        only represents a grouping or "bag" of Elements. For
        this reason, it is incorrect to subclass groups that
        have state/status from CollectionOfMSEs - an example
        is Redundancy Group (which is correctly subclassed
        from LogicalElement). Collections typically aggregate
        "like" objects, and represent an optimization. Without
       Collections, one is forced to define individual
        ElementSetting and ElementConfiguration associations,
        to tie Settings and Configuration objects to
        individual ManagedSystemElements. There may be much
        duplication in assigning the same Setting to multiple
        objects. In addition, using the Collection object
        allows the determination that the Setting and
        Configuration associations are indeed the same for the
       Collection"s members. This information would otherwise
        be obtained by defining the Collection in a
        proprietary manner, and then querying the
        ElementSetting and ElementConfiguration associations
        to determine if the Collection set is completely
        covered.'
  SUP dlm1Collection ABSTRACT
  MAY ( dlmCollectionID )
)
```

3.13 Configuration Classes

This object allows the grouping of sets of parameters (defined in Setting objects) and dependencies for one or more managed system elements. The configuration object represents a certain behavior, or a desired functional state for the managed system elements. The desired functional state is typically driven by external requirements such as time or location. For example, to connect to a Mail System from 'home', a dependency on a modem exists, but a dependency on a network adapter exists at 'work'. Settings for the pertinent logical devices can be defined and aggregated by the configuration. Therefore, two 'Connect to Mail' configurations may be defined grouping the relevant dependencies and setting objects.

```
( 1.3.6.1.4.1.412.100.2.1.3.12 NAME 'dlm1Configuration'
  DESC 'The Configuration object allows the grouping of
        sets of parameters (defined in Setting objects) and
        dependencies for one or more ManagedSystemElements.
        The Configuration object represents a certain
        behavior, or a desired functional state for the
        ManagedSystemElements. The desired functional state is
       typically driven by external requirements such as time
        or location. For example, to connect to a Mail System
        from "home", a dependency on a modem exists, but a
        dependency on a network adapter exists at "work".
        Settings for the pertinent LogicalDevices (in this
        example, POTSModem and NetworkAdapter) can be defined
        and aggregated by the Configuration. Therefore, two
        "Connect to Mail" Configurations may be defined
        grouping the relevant dependencies and Setting
        objects.'
  SUP dlm1ManagedElement ABSTRACT
)

( 1.3.6.1.4.1.412.100.2.1.3.13 NAME 'dlm1ConfigurationAuxClass'
  DESC 'The Configuration object allows the grouping of
        sets of parameters (defined in Setting objects) and
        dependencies for one or more ManagedSystemElements.
        The Configuration object represents a certain
        behavior, or a desired functional state for the
        ManagedSystemElements. The desired functional state is
       typically driven by external requirements such as time
        or location. For example, to connect to a Mail System
        from "home", a dependency on a modem exists, but a
        dependency on a network adapter exists at "work".
        Settings for the pertinent LogicalDevices (in this
        example, POTSModem and NetworkAdapter) can be defined
        and aggregated by the Configuration. Therefore, two
        "Connect to Mail" Configurations may be defined
        grouping the relevant dependencies and Setting
        objects.'
  SUP dlm1Configuration AUXILIARY
)

( 1.3.6.1.4.1.412.100.2.1.3.14 NAME 'dlm1ConfigurationInstance'
  DESC 'The Configuration object allows the grouping of
        sets of parameters (defined in Setting objects) and
        dependencies for one or more ManagedSystemElements.
        The Configuration object represents a certain
        behavior, or a desired functional state for the
        ManagedSystemElements. The desired functional state is
       typically driven by external requirements such as time
```

```
         or location. For example, to connect to a Mail System
         from "home", a dependency on a modem exists, but a
         dependency on a network adapter exists at "work".
         Settings for the pertinent LogicalDevices (in this
         example, POTSModem and NetworkAdapter) can be defined
         and aggregated by the Configuration. Therefore, two
         "Connect to Mail" Configurations may be defined
         grouping the relevant dependencies and Setting
         objects.'
     SUP dlm1Configuration
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.1 NAME
'dlm1ConfigurationInstanceNameForm1'
     OC dlm1ConfigurationInstance
     MUST ( orderedCimKeys )
   )

   ( <core-sr-1> NAME 'dlm1ConfigurationInstanceStructureRule1'
     Form dlm1ConfigurationInstanceNameForm1
   )
```

The following content rule specifies the auxiliary classes that may be attached to dlm1ConfigurationInstance.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.14 NAME
'dlm1ConfigurationInstanceContentRule'
     DESC 'Aux classes that can attach to
           dlm1ConfigurationInstance.'
     MAY ( dlm1ElementConfigurationAuxClass $
           dlm1CollectionConfigurationAuxClass $
           dlm1ConfigurationComponentAuxClass $
           dlm1SettingContextAuxClass )
   )
```

3.14 Setting

This class represents configuration-related and operational parameters for one or more managed system element(s). A managed system element may have multiple setting objects associated with it. The current operational values for an element's parameters are reflected by properties in the element itself or by properties in its associations. These properties do not have to be the same values present in the setting object. For example, a modem may have a setting baud rate of 56Kb/sec but be operating at 19.2Kb/sec.

```
   ( 1.3.6.1.4.1.412.100.2.2.139 NAME 'dlmSettingID'
     DESC 'The identifier by which the Setting object is
           known.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.15 NAME 'dlm1Setting'
     DESC 'The Setting class represents
           configuration-related and operational parameters for
           one or more ManagedSystem Element(s). A
           ManagedSystemElement may have multiple Setting objects
          associated with it. The current operational values for
```

```
         an Element"s parameters are reflected by properties in
        the Element itself or by properties in its
         associations. These properties do not have to be the
         same values present in the Setting object. For
         example, a modem may have a Setting baud rate of
         56Kb/sec but be operating at 19.2Kb/sec.'
    SUP dlm1ManagedElement ABSTRACT
    MAY ( dlmSettingID )
  )
```

3.15 Product Classes

This concrete class that is a collection of physical elements, software features and/or other products, acquired as a unit. Acquisition implies an agreement between supplier and consumer that may have implications to product licensing, support and warranty.

```
  ( 1.3.6.1.4.1.412.100.2.2.140 NAME 'dlmIdentifyingNumber'
    DESC 'Product identification such as a serial number on
        software, a die number on a hardware chip, or (for
         non-commercial Products) a project number.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.141 NAME 'dlmSKUNumber'
    DESC 'Product SKU (stock keeping unit) information.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.142 NAME 'dlmVendor'
    DESC 'The name of the Product"s supplier, or entity
         selling the Product (the manufacturer, reseller, OEM,
         etc.). Corresponds to the Vendor property in the
         Product object in the DMTF Solution Exchange Standard.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.143 NAME 'dlmWarrantyDuration'
    DESC 'If this Product is under warranty, the duration
         of the warranty in days. The value is considered to be
        Days.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.2.144 NAME 'dlmWarrantyStartDate'
    DESC 'If this Product is under warranty, the start date
        of the warranty.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE
  )

  ( 1.3.6.1.4.1.412.100.2.1.3.16 NAME 'dlm1Product'
    DESC 'Product is a concrete class that is a collection
         of PhysicalElements, SoftwareFeatures and/or other
         Products, acquired as a unit. Acquisition implies an
         agreement between supplier and consumer which may have
        implications to Product licensing, support and
         warranty. Non-commercial (e.g., in-house developed
         Products) should also be identified as an instance of
         Product.'
   SUP dlm1ManagedElement ABSTRACT
   MAY ( dlmIdentifyingNumber $ dlmName $ dlmSKUNumber $
         dlmVendor $ dlmVersion $ dlmWarrantyDuration $
         dlmWarrantyStartDate )
 )

 ( 1.3.6.1.4.1.412.100.2.1.3.17 NAME 'dlm1ProductAuxClass'
   DESC 'Product is a concrete class that is a collection
         of PhysicalElements, SoftwareFeatures and/or other
         Products, acquired as a unit. Acquisition implies an
         agreement between supplier and consumer which may have
        implications to Product licensing, support and
         warranty. Non-commercial (e.g., in-house developed
         Products) should also be identified as an instance of
         Product.'
   SUP dlm1Product AUXILIARY
 )

 ( 1.3.6.1.4.1.412.100.2.1.3.18 NAME 'dlm1ProductInstance'
   DESC 'Product is a concrete class that is a collection
         of PhysicalElements, SoftwareFeatures and/or other
         Products, acquired as a unit. Acquisition implies an
         agreement between supplier and consumer which may have
        implications to Product licensing, support and
         warranty. Non-commercial (e.g., in-house developed
         Products) should also be identified as an instance of
         Product.'
   SUP dlm1Product
 )

 ( 1.3.6.1.4.1.412.100.2.3.3.2 NAME 'dlm1ProductInstanceNameForm1'
   OC dlm1ProductInstance
   MUST ( orderedCimKeys )
 )

 ( <core-sr-2> NAME 'dlm1ProductInstanceStructureRule1'
   Form dlm1ProductInstanceNameForm1
 )
```

The following content rule specifies the auxiliary classes that may be attached to dlm1ProductInstance.

```
 ( 1.3.6.1.4.1.412.100.2.1.3.18 NAME 'dlm1ProductInstanceContentRule'
   DESC 'Aux classes that can attach to
         dlm1ProductInstance.'
   MAY ( dlm1ProductProductDependencyAuxClass $
         dlm1ProductSupportAuxClass $ dlm1ProductFRUAuxClass $
         dlm1ProductParentChildAuxClass $
         dlm1FRUIncludesProductAuxClass $
         dlm1ProductPhysicalElementsAuxClass )
 )
```

3.16 SupportAccess Classes

These classes define how to obtain help for a product.

```
   ( 1.3.6.1.4.1.412.100.2.2.145 NAME 'dlmCommunicationInfo'
     DESC 'CommunicationInfo provides the details of the
           Communication Mode. For example, if the
           CommunicationMode is "Phone", CommunicationInfo
           specifies the phone number to be called.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.146 NAME 'dlmCommunicationMode'
     DESC 'CommunicationMode defines the form of
           communication in order to obtain support. For example,
          phone communication (value =2), fax (3) or email (8)
           can be specified.  Values are 1="Other", 2="Phone",
           3="Fax", 4="BBS", 5="Online Service", 6="Web Page",
           7="FTP", 8="E-mail"'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.147 NAME 'dlmLocale'
     DESC 'Locale defines the geographic region and/or
           language dialect to which this Support resource
           pertains.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.148 NAME 'dlmSupportAccessId'
     DESC 'SupportAccessID is an arbitrary, free form string
          defined by the Product Vendor or by the organization
           that deploys the Product.  This property, since it is
           a key, should be unique throughout the enterprise.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.19 NAME 'dlm1SupportAccess'
     DESC 'The SupportAccess association defines how to
           obtain assistance for a Product.'
     SUP dlm1ManagedElement ABSTRACT
     MAY ( dlmCommunicationInfo $ dlmCommunicationMode $
           dlmDescription $ dlmLocale $ dlmSupportAccessId )
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.20 NAME 'dlm1SupportAccessAuxClass'
     DESC 'The SupportAccess association defines how to
           obtain assistance for a Product.'
     SUP dlm1SupportAccess AUXILIARY
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.21 NAME 'dlm1SupportAccessInstance'
     DESC 'The SupportAccess association defines how to
           obtain assistance for a Product.'
     SUP dlm1SupportAccess
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.3 NAME
'dlm1SupportAccessInstanceNameForm1'
     OC dlm1SupportAccessInstance
     MUST ( orderedCimKeys )
   )

   ( <core-sr-3> NAME 'dlm1SupportAccessInstanceStructureRule1'
     Form dlm1SupportAccessInstanceNameForm1
   )
```

The following content rule specifies the auxiliary classes that may be attached to dlm1SupportAccessInstance.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.21 NAME
'dlm1SupportAccessInstanceContentRule'
     DESC 'Aux classes that can attach to
           dlm1SupportAccessInstance.'
     MAY ( dlm1ProductSupportAuxClass )
   )
```

3.17 FRU Classes

These classes model vendor-defined collection of products and/or physical elements that is associated with a product for supporting, maintaining or upgrading that product at the customer's location. FRU is an acronym for 'field replaceable unit'.

```
   ( 1.3.6.1.4.1.412.100.2.2.149 NAME 'dlmFRUNumber'
     DESC 'FRU ordering information.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.150 NAME 'dlmRevisionLevel'
     DESC 'The FRU"s revision level.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.22 NAME 'dlm1FRU'
     DESC 'The FRU class is a vendor-defined collection of
           Products and/or PhysicalElements that is associated
           with a Product for the purpose of supporting,
           maintaining or upgrading that Product at the
           customer"s location. FRU is an acronym for "field
           replaceable unit". '
     SUP dlm1ManagedElement ABSTRACT
     MAY ( dlmDescription $ dlmFRUNumber $
           dlmIdentifyingNumber $ dlmName $ dlmRevisionLevel $
           dlmVendor )
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.23 NAME 'dlm1FRUAuxClass'
     DESC 'The FRU class is a vendor-defined collection of
           Products and/or PhysicalElements that is associated
           with a Product for the purpose of supporting,
           maintaining or upgrading that Product at the
           customer"s location. FRU is an acronym for "field
           replaceable unit". '
     SUP dlm1FRU AUXILIARY
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.24 NAME 'dlm1FRUInstance'
     DESC 'The FRU class is a vendor-defined collection of
           Products and/or PhysicalElements that is associated
           with a Product for the purpose of supporting,
           maintaining or upgrading that Product at the
           customer"s location. FRU is an acronym for "field
           replaceable unit". '
     SUP dlm1FRU
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.4 NAME 'dlm1FRUInstanceNameForm1'
     OC dlm1FRUInstance
     MUST ( orderedCimKeys )
   )

   ( <core-sr-4> NAME 'dlm1FRUInstanceStructureRule1'
     Form dlm1FRUInstanceNameForm1
   )
```

The following content rule specifies the auxiliary classes that may be attached to dlm1FRUInstance.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.24 NAME 'dlm1FRUInstanceContentRule'
     DESC 'Aux classes that can attach to dlm1FRUInstance.'
     MAY ( dlm1ProductFRUAuxClass $
           dlm1FRUPhysicalElementsAuxClass $
           dlm1FRUIncludesProductAuxClass )
   )
```

3.18 CollectedCollections Classes

These classes represent that a CollectionOfMSEs may itself be contained in another CollectionOfMSEs object.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.25 NAME 'dlm1CollectedCollections'
     DESC 'CollectedCollections is an aggregation
           association representing that a CollectionOfMSEs may
           itself be contained in a CollectionOfMSEs.'
     SUP top ABSTRACT
   )
   ( 1.3.6.1.4.1.412.100.2.2.151 NAME
'dlmCollectedCollectionsCollectionRef'
     DESC 'The "higher level" or parent element in the
           aggregation.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )

   ( 1.3.6.1.4.1.412.100.2.2.152 NAME
'dlmCollectedCollectionsCollectionInCollectionRef'
     DESC 'The "collected" Collection.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.26 NAME
'dlm1CollectedCollectionsAuxClass'
     DESC 'CollectedCollections is an aggregation
           association representing that a CollectionOfMSEs may
           itself be contained in a CollectionOfMSEs.'
```

```
    SUP dlm1CollectedCollections AUXILIARY
    MAY ( dlmCollectedCollectionsCollectionRef $
          dlmCollectedCollectionsCollectionInCollectionRef )
  )
```

3.19 LogicalIdentity

This auxiliary class represents an abstract and generic association, showing that two LogicalElements represent different aspects of the same underlying entity. This relationship conveys what could be defined with multiple inheritance. It is restricted to the 'logical' aspects of a ManagedSystemElement. In most scenarios, the equivalence of keys or some other identifying properties of the related elements determines the identity relationship. The association should only be used in well-understood scenarios. This is why the association is abstract - allowing more concrete definition and clarification in subclasses.

```
  ( 1.3.6.1.4.1.412.100.2.1.3.27 NAME 'dlm1LogicalIdentity'
    DESC 'LogicalIdentity is an abstract and generic
          association, indicating that two LogicalElements
          represent different aspects of the same underlying
          entity. This relationship conveys what could be
          defined with multiple inheritance. It is restricted to
         the "logical" aspects of a ManagedSystem Element. In
          most scenarios, the Identity relationship is
          determined by the equivalence of Keys or some other
          identifying properties of the related Elements. The
          association should only be used in well understood
          scenarios. This is why the association is abstract -
          allowing more concrete definition and clarification in
         subclasses. One of the scenarios where this
          relationship is reasonable is to represent that a
          Device is both a "bus" entity and a "functional"
          entity. For example, a Device could be both a USB
          (bus) and a Keyboard (functional) entity.'
    SUP top ABSTRACT
  )
```

3.20 ConfigurationComponent Classes

This association aggregates 'lower-level' configuration objects into a \'high-level' configuration. This enables the assembly of complex configurations by grouping together simpler ones.

```
  ( 1.3.6.1.4.1.412.100.2.1.3.28 NAME 'dlm1ConfigurationComponent'
    DESC 'ConfigurationComponent aggregates "lower-level"
          Configuration objects into a "high-level"
          Configuration. This enables the assembly of complex
          Configurations by grouping together simpler ones. For
          example, a logon policy for the United States could
          consist of two Configuration groups, one for the east
          coast and one for the west coast. Each of these could
          in turn consist of multiple Configurations to handle
          different aspects of the logon process.'
    SUP top ABSTRACT
  )
```

```
   ( 1.3.6.1.4.1.412.100.2.2.153 NAME
'dlmConfigurationComponentConfigComponentRef'
     DESC 'A Configuration that is part of a "higher-level"
           Configuration.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.154 NAME
'dlmConfigurationComponentConfigGroupRef'
     DESC 'The Configuration that aggregates additional
           Configurations. '
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.29 NAME
'dlm1ConfigurationComponentAuxClass'
     DESC 'ConfigurationComponent aggregates "lower-level"
           Configuration objects into a "high-level"
           Configuration. This enables the assembly of complex
           Configurations by grouping together simpler ones. For
           example, a logon policy for the United States could
           consist of two Configuration groups, one for the east
           coast and one for the west coast. Each of these could
           in turn consist of multiple Configurations to handle
           different aspects of the logon process.'
     SUP dlm1ConfigurationComponent AUXILIARY
     MAY ( dlmConfigurationComponentConfigComponentRef $
           dlmConfigurationComponentConfigGroupRef )
   )
```

3.21 ElementConfiguration Classes

This association relates a configuration object to one or more managed system elements. The configuration object represents a certain behavior, or a desired functional state for the associated managed system elements.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.30 NAME 'dlm1ElementConfiguration'
     DESC 'This association relates a Configuration object
           to one or more ManagedSystemElements. The
           Configuration object represents a certain behavior, or
          a desired functional state for the associated
           ManagedSystemElements.'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.155 NAME
'dlmElementConfigurationConfigurationRef'
     DESC 'The Configuration object that groups the Settings
          and dependencies associated with the
           ManagedSystemElement.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.156 NAME
'dlmElementConfigurationElementRef'
     DESC 'The ManagedSystemElement.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.31 NAME
'dlm1ElementConfigurationAuxClass'
     DESC 'This association relates a Configuration object
           to one or more ManagedSystemElements. The
           Configuration object represents a certain behavior, or
          a desired functional state for the associated
           ManagedSystemElements.'
     SUP dlm1ElementConfiguration AUXILIARY
     MAY ( dlmElementConfigurationConfigurationRef $
           dlmElementConfigurationElementRef )
   )
```

3.22 ConfigurationCollection Classes

These classes relate a Configuration object to one or more CollectionOfMSEs objects. The Configuration object represents a certain behavior, or a desired functional state for the associated collection.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.32 NAME 'dlm1CollectionConfiguration'
     DESC 'This association relates a Configuration object
           to one or more CollectionOfMSEs objects. The
           Configuration object represents a certain behavior, or
          a desired functional state for the associated
           Collection.'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.157 NAME
'dlmCollectionConfigurationCollectionRef'
     DESC 'The CollectionOfMSEs.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.158 NAME
'dlmCollectionConfigurationConfigurationRef'
     DESC 'The Configuration object that groups the Settings
          and dependencies associated with the Collection.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.33 NAME
'dlm1CollectionConfigurationAuxClass'
     DESC 'This association relates a Configuration object
           to one or more CollectionOfMSEs objects. The
           Configuration object represents a certain behavior, or
          a desired functional state for the associated
           Collection.'
     SUP dlm1CollectionConfiguration AUXILIARY
     MAY ( dlmCollectionConfigurationCollectionRef $
         dlmCollectionConfigurationConfigurationRef )
   )
```

3.23 ElementSetting Classes

These classes represent the association between managed system elements and the setting class(es) defined for them.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.34 NAME 'dlm1ElementSetting'
     DESC 'ElementSetting represents the association between
          Managed SystemElements and the Setting class(es)
           defined for them.'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.159 NAME 'dlmElementSettingElementRef'
     DESC 'The ManagedSystemElement.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.160 NAME 'dlmElementSettingSettingRef'
     DESC 'The Setting object associated with the
           ManagedSystem Element.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.35 NAME 'dlm1ElementSettingAuxClass'
     DESC 'ElementSetting represents the association between
          Managed SystemElements and the Setting class(es)
           defined for them.'
     SUP dlm1ElementSetting AUXILIARY
     MAY ( dlmElementSettingElementRef $
           dlmElementSettingSettingRef )
   )
```

3.24 DefaultSetting Classes

These classes represent the association between a ManagedSystemElement and the single Setting class that is defined to be the default setting for this element.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.36 NAME 'dlm1DefaultSetting'
     DESC 'DefaultSetting represents the association between
          a Managed SystemElement and the single Setting class
           that is defined to be the default setting for this
           Element.'
     SUP dlm1ElementSetting ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.161 NAME 'dlmDefaultSettingElementRef'
     DESC 'The ManagedSystemElement.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.162 NAME 'dlmDefaultSettingSettingRef'
     DESC 'The Setting object which is the default.'
```

```
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.37 NAME 'dlm1DefaultSettingAuxClass'
  DESC 'DefaultSetting represents the association between
       a Managed SystemElement and the single Setting class
        that is defined to be the default setting for this
        Element.'
  SUP dlm1DefaultSetting AUXILIARY
  MAY ( dlmDefaultSettingElementRef $
        dlmDefaultSettingSettingRef )
)
```

3.25 SettingContext Classes

These classes associate a setting with one or more configuration objects. For example, a network adapter's settings could change based on the site/network to which its hosting computer system is attached.

```
( 1.3.6.1.4.1.412.100.2.1.3.38 NAME 'dlm1SettingContext'
  DESC 'This relationship associates Configuration
        objects with Setting objects. For example, a
        NetworkAdapter"s Settings could change based on the
        site/network to which its hosting ComputerSystem is
        attached. In this case, the ComputerSystem would have
        two different Configuration objects, corresponding to
        the differences in network configuration for the two
        network segments. Configuration A would aggregate a
        Setting object for the NetworkAdapter when operating
        on segment \"ANet\", whereas Configuration B would
        aggregate a different NetworkAdapter Setting object,
        specific to segment \"BNet\". Note that many Settings
        of the computer are independent of the network
        Configuration. For example, both Configurations A and
        B would aggregate the same Setting object for the
        ComputerSystem"s MonitorResolution.'
  SUP top ABSTRACT
)

( 1.3.6.1.4.1.412.100.2.2.163 NAME 'dlmSettingContextContextRef'
  DESC 'The Configuration object that aggregates the
        Setting.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  EQUALITY distinguishedNameMatch
)

( 1.3.6.1.4.1.412.100.2.2.164 NAME 'dlmSettingContextSettingRef'
  DESC 'An aggregated Setting.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  EQUALITY distinguishedNameMatch
)

( 1.3.6.1.4.1.412.100.2.1.3.39 NAME 'dlm1SettingContextAuxClass'
  DESC 'This relationship associates Configuration
        objects with Setting objects. For example, a
        NetworkAdapter"s Settings could change based on the
        site/network to which its hosting ComputerSystem is
```

```
          attached. In this case, the ComputerSystem would have
          two different Configuration objects, corresponding to
          the differences in network configuration for the two
          network segments. Configuration A would aggregate a
          Setting object for the NetworkAdapter when operating
          on segment \"ANet\", whereas Configuration B would
          aggregate a different NetworkAdapter Setting object,
          specific to segment \"BNet\". Note that many Settings
          of the computer are independent of the network
          Configuration. For example, both Configurations A and
          B would aggregate the same Setting object for the
          ComputerSystem"s MonitorResolution.'
    SUP dlm1SettingContext AUXILIARY
    MAY ( dlmSettingContextContextRef $
          dlmSettingContextSettingRef )
  )
```

3.26 CollectionSetting Classes

These classes represent the association between a CollectionOfMSEs class and the Setting class(es) defined for them.

```
  ( 1.3.6.1.4.1.412.100.2.1.3.40 NAME 'dlm1CollectionSetting'
    DESC 'CollectionSetting represents the association
          between a CollectionOfMSEs class and the Setting
          class(es) defined for them.'
    SUP top ABSTRACT
  )

  ( 1.3.6.1.4.1.412.100.2.2.165 NAME
'dlmCollectionSettingCollectionRef'
    DESC 'The CollectionOfMSEs.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
    EQUALITY distinguishedNameMatch
  )

  ( 1.3.6.1.4.1.412.100.2.2.166 NAME 'dlmCollectionSettingSettingRef'
    DESC 'The Setting object associated with the
          Collection.'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
    EQUALITY distinguishedNameMatch
  )

  ( 1.3.6.1.4.1.412.100.2.1.3.41 NAME 'dlm1CollectionSettingAuxClass'
    DESC 'CollectionSetting represents the association
          between a CollectionOfMSEs class and the Setting
          class(es) defined for them.'
    SUP dlm1CollectionSetting AUXILIARY
    MAY ( dlmCollectionSettingCollectionRef $
          dlmCollectionSettingSettingRef )
  )
```

3.27 Dependency

This abstract class represents a generic association used to establish dependency relationships between objects.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.42 NAME 'dlm1Dependency'
     DESC 'Dependency is a generic association used to
           establish dependency relationships between
           ManagedElements.'
     SUP top ABSTRACT
   )
```

3.28 ServiceAccessBySAP Classes

These classes identify the access points for a service. For example, Netware, MacIntosh or Windows service access points may access a printer, which may be hosted on different system.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.43 NAME 'dlm1ServiceAccessBySAP'
     DESC 'ServiceAccessBySAP is an association that
           identifies the access points for a Service. For
           example, a printer may be accessed by Netware,
           MacIntosh or Windows ServiceAccess Points, potentially
          hosted on different Systems.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.167 NAME
'dlmServiceAccessBySAPAntecedentRef'
     DESC 'The Service. '
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.168 NAME
'dlmServiceAccessBySAPDependentRef'
     DESC 'An Access Point for a Service. Access points are
           dependent in this relationship since they have no
           function without a corresponding Service. '
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.44 NAME 'dlm1ServiceAccessBySAPAuxClass'
     DESC 'ServiceAccessBySAP is an association that
           identifies the access points for a Service. For
           example, a printer may be accessed by Netware,
           MacIntosh or Windows ServiceAccess Points, potentially
          hosted on different Systems.'
     SUP dlm1ServiceAccessBySAP AUXILIARY
     MAY ( dlmServiceAccessBySAPAntecedentRef $
           dlmServiceAccessBySAPDependentRef )
   )
```

3.29 HostedService

This class maps the association between a Service and the System on which it resides. While this could be represented with DIT containment, this class is provided to allow for more general relationships.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.45 NAME 'dlm1HostedService'
     DESC 'HostedService is an association between a Service
```

```
          and the System on which the functionality resides.  The
          cardinality of this association is 1-to-many.  A System
          may host many Services. Services are weak with respect
           to their hosting System. Heuristic:  A Service is
           hosted on the System where the LogicalDevices or
           SoftwareFeatures that implement the Service are
           located.  The model does not represent Services hosted
          across multiple systems.  This is modeled as an
           ApplicationSystem that acts as an aggregation point
           for Services, that are each located on a single host.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.169 NAME 'dlmHostedServiceDependentRef'
     DESC 'The Service hosted on the System.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.170 NAME 'dlmHostedServiceAntecedentRef'
     DESC 'The hosting System.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.46 NAME 'dlm1HostedServiceAuxClass'
     DESC 'HostedService is an association between a Service
          and the System on which the functionality resides.  The
          cardinality of this association is 1-to-many.  A System
          may host many Services. Services are weak with respect
           to their hosting System. Heuristic:  A Service is
           hosted on the System where the LogicalDevices or
           SoftwareFeatures that implement the Service are
           located.  The model does not represent Services hosted
          across multiple systems.  This is modeled as an
           ApplicationSystem that acts as an aggregation point
           for Services, that are each located on a single host.'
     SUP dlm1HostedService AUXILIARY
     MAY ( dlmHostedServiceDependentRef $
           dlmHostedServiceAntecedentRef )
   )
```

3.30 HostedAccessPoint

These classes map an association between a ServiceAccessPoint and the System that provides it. Like HostedService, this is provided for more general representations than what is available through DIT containment.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.47 NAME 'dlm1HostedAccessPoint'
     DESC 'HostedAccessPoint is an association between a
           Service AccessPoint and the System on which it is
           provided.  The cardinality of this association is
           1-to-many and is weak with respect to the System. Each
          System may host many ServiceAccessPoints.  Heuristic:
           If the implementation of the ServiceAccessPoint is
           modeled, it must be implemented by a Device or
           SoftwareFeature that is part of the System hosting the
          ServiceAccessPoint.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.171 NAME
'dlmHostedAccessPointDependentRef'
     DESC 'The SAP(s) that are hosted on this System.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.172 NAME
'dlmHostedAccessPointAntecedentRef'
     DESC 'The hosting System.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.48 NAME 'dlm1HostedAccessPointAuxClass'
     DESC 'HostedAccessPoint is an association between a
           Service AccessPoint and the System on which it is
           provided.  The cardinality of this association is
           1-to-many and is weak with respect to the System. Each
          System may host many ServiceAccessPoints.  Heuristic:
           If the implementation of the ServiceAccessPoint is
           modeled, it must be implemented by a Device or
           SoftwareFeature that is part of the System hosting the
          ServiceAccessPoint.'
     SUP dlm1HostedAccessPoint AUXILIARY
     MAY ( dlmHostedAccessPointDependentRef $
           dlmHostedAccessPointAntecedentRef )
   )
```

3.31 ProvidesServiceToElement Classes

These classes map an association is used to describe that ManagedSystemElements may be dependent on the functionality of one or more Services.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.49 NAME 'dlm1ProvidesServiceToElement'
     DESC 'ProvidesServiceToElement is used to describe that
          ManagedSystemElements may be dependent on the
           functionality of one or more Services. An example is
           that a Processor and an Enclosure (PhysicalElement)
           are dependent on AlertOn LAN Services to signal an
           incomplete or erroneous boot, and hardware-related
           errors.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.173 NAME
'dlmProvidesServiceToElementDependentRef'
     DESC 'The ManagedSystemElement dependent on the
           Service.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.174 NAME
'dlmProvidesServiceToElementAntecedentRef'
```

```
     DESC 'The Service provided.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.50 NAME
'dlm1ProvidesServiceToElementAuxClass'
     DESC 'ProvidesServiceToElement is used to describe that
          ManagedSystemElements may be dependent on the
           functionality of one or more Services. An example is
           that a Processor and an Enclosure (PhysicalElement)
           are dependent on AlertOn LAN Services to signal an
           incomplete or erroneous boot, and hardware-related
           errors.'
     SUP dlm1ProvidesServiceToElement AUXILIARY
     MAY ( dlmProvidesServiceToElementDependentRef $
           dlmProvidesServiceToElementAntecedentRef )
   )
```

3.32 ServiceServiceDependency Classes

These classes map an association between two services, showing that the latter is required to be present, required to have completed, or must be absent for the former Service to provide its functionality. For example, boot Services may be dependent on underlying BIOS disk and initialization services. For initialization services, the boot service is simply dependent on the initialization services completing.

```
   ( 1.3.6.1.4.1.412.100.2.2.175 NAME 'dlmRestartService'
     DESC 'this property describes that the antecedent
           service must be restarted after the dependent
           operation is complete.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.176 NAME 'dlmTypeOfDependency'
     DESC 'The nature of the Service to Service dependency.
           This property describes that the associated Service
           must have completed (value=2), must be started (3) or
           must not be started (4) in order for the Service to
           function.  Values are 0="Unknown", 1="Other",
           2="Service Must Have Completed", 3="Service Must Be
           Started", 4="Service Must Not Be Started"'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.51 NAME 'dlm1ServiceServiceDependency'
     DESC 'ServiceServiceDependency is an association
           between a Service and another Service, indicating that
          the latter is required to be present, required to have
           completed, or must be absent for the former Service to
          provide its functionality. For example, Boot Services
           may be dependent upon underlying BIOS Disk and
           initialization Services. In the case of the
           initialization Services, the Boot Service is simply
           dependent on the init Services completing.  For the
           Disk Services, Boot Services may actually utilize the
```

```
            SAPs of this Service.  This usage dependency is
            modeled via the ServiceSAPDependency association.'
      SUP dlm1ProvidesServiceToElement ABSTRACT
      MAY ( dlmRestartService $ dlmTypeOfDependency )
    )

    ( 1.3.6.1.4.1.412.100.2.2.177 NAME
'dlmServiceServiceDependencyAntecedentRef'
      DESC 'The required Service.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
    )

    ( 1.3.6.1.4.1.412.100.2.2.178 NAME
'dlmServiceServiceDependencyDependentRef'
      DESC 'The Service that is dependent on an underlying
            Service.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
    )

    ( 1.3.6.1.4.1.412.100.2.1.3.52 NAME
'dlm1ServiceServiceDependencyInstance'
      DESC 'ServiceServiceDependency is an association
            between a Service and another Service, indicating that
           the latter is required to be present, required to have
            completed, or must be absent for the former Service to
           provide its functionality. For example, Boot Services
            may be dependent upon underlying BIOS Disk and
            initialization Services. In the case of the
            initialization Services, the Boot Service is simply
            dependent on the init Services completing.  For the
            Disk Services, Boot Services may actually utilize the
            SAPs of this Service.  This usage dependency is
            modeled via the ServiceSAPDependency association.'
      SUP dlm1ServiceServiceDependency
      MAY ( dlmServiceServiceDependencyAntecedentRef $
            dlmServiceServiceDependencyDependentRef )
    )

    ( 1.3.6.1.4.1.412.100.2.3.3.5 NAME
'dlm1ServiceServiceDependencyInstanceNameForm1'
      OC dlm1ServiceServiceDependencyInstance
      MUST ( orderedCimKeys )
    )

    ( <core-sr-5> NAME
'dlm1ServiceServiceDependencyInstanceStructureRule1'
      Form dlm1ServiceServiceDependencyInstanceNameForm1
    )

    ( 1.3.6.1.4.1.412.100.2.2.179 NAME
'dlmServiceServiceDependencyHelperRef'
      DESC 'Pointer to ServiceServiceDependencyInstance.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
    )

    ( 1.3.6.1.4.1.412.100.2.1.3.53 NAME
 'dlm1ServiceServiceDependencyHelper'
      DESC 'Helper class for finding
            ServiceServiceDependency.'
      SUP top AUXILIARY
      MAY ( dlmHelperRefToServiceServiceDependency )
    )
```

3.33 ServiceSAPDependency Classes

These classes map an association between a service and a service access point showing that the referenced SAP is used by the service to provide its functionality. For example, boot services may invoke BIOS disk services (interrupts) to function.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.54 NAME 'dlm1ServiceSAPDependency'
     DESC 'ServiceSAPDependency is an association between a
           Service and a ServiceAccessPoint indicating that the
           referenced SAP is utilized by the Service to provide
           its functionality. For example, Boot Services may
           invoke BIOS" Disk Services (interrupts) in order to
           function.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.180 NAME
'dlmServiceSAPDependencyDependentRef'
     DESC 'The Service that is dependent on an underlying
           SAP.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.181 NAME
'dlmServiceSAPDependencyAntecedentRef'
     DESC 'The required ServiceAccessPoint'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.55 NAME
'dlm1ServiceSAPDependencyAuxClass'
     DESC 'ServiceSAPDependency is an association between a
           Service and a ServiceAccessPoint indicating that the
           referenced SAP is utilized by the Service to provide
           its functionality. For example, Boot Services may
           invoke BIOS" Disk Services (interrupts) in order to
           function.'
     SUP dlm1ServiceSAPDependency AUXILIARY
     MAY ( dlmServiceSAPDependencyDependentRef $
           dlmServiceSAPDependencyAntecedentRef )
   )
```

3.34 SAPSAPDependency Classes

These classes model an association between two service access points showing that the latter is required in order for the former to use or connect with its service. For example, to print at a network printer, local print access points must use underlying network-related SAPs, or protocol endpoints, to send the print request.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.56 NAME 'dlm1SAPSAPDependency'
     DESC 'SAPSAPDependency is an association between a
           Service AccessPoint and another ServiceAccessPoint
           indicating that the latter is required in order for
           the former ServiceAccess Point to utilize or connect
           with its Service. For example, to print at a network
           printer, local Print Access Points must utilize
           underlying network-related SAPs, or ProtocolEndpoints,
          in order to send the print request.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.182 NAME
'dlmSAPSAPDependencyAntecedentRef'
     DESC 'The required ServiceAccessPoint.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.183 NAME 'dlmSAPSAPDependencyDependentRef'
     DESC 'The ServiceAccessPoint that is dependent on an
           underlying SAP.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.57 NAME 'dlm1SAPSAPDependencyAuxClass'
     DESC 'SAPSAPDependency is an association between a
           Service AccessPoint and another ServiceAccessPoint
           indicating that the latter is required in order for
           the former ServiceAccess Point to utilize or connect
           with its Service. For example, to print at a network
           printer, local Print Access Points must utilize
           underlying network-related SAPs, or ProtocolEndpoints,
          in order to send the print request.'
     SUP dlm1SAPSAPDependency AUXILIARY
     MAY ( dlmSAPSAPDependencyAntecedentRef $
           dlmSAPSAPDependencyDependentRef )
   )
```

3.35 Realizes Classes

These classes define the mapping between a logical device and the physical component that implements the device.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.58 NAME 'dlm1Realizes'
     DESC 'Realizes is the association that defines the
           mapping between a Logical Device and the physical
           component that implements the Device.'
     SUP dlm1Dependency ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.184 NAME 'dlmRealizesDependentRef'
     DESC 'The LogicalDevice.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.185 NAME 'dlmRealizesAntecedentRef'
     DESC 'The physical component that implements the
           Device.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.59 NAME 'dlm1RealizesAuxClass'
     DESC 'Realizes is the association that defines the
           mapping between a Logical Device and the physical
           component that implements the Device.'
     SUP dlm1Realizes AUXILIARY
     MAY ( dlmRealizesDependentRef $
           dlmRealizesAntecedentRef )
   )
```

3.36 MemberOfCollection Classes

These classes establish membership of ManagedElement objects in a collection.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.60 NAME 'dlm1MemberOfCollection'
     DESC 'MemberOfCollection is an aggregation used to
           establish membership of ManagedElements in a
           Collection.'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.186 NAME
'dlmMemberOfCollectionCollectionRef'
     DESC 'The Collection that aggregates members'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.187 NAME 'dlmMemberOfCollectionMemberRef'
     DESC 'The aggregated member of the collection.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.61 NAME 'dlm1MemberOfCollectionAuxClass'
     DESC 'MemberOfCollection is an aggregation used to
           establish membership of ManagedElements in a
           Collection.'
     SUP dlm1MemberOfCollection AUXILIARY
     MAY ( dlmMemberOfCollectionCollectionRef $
           dlmMemberOfCollectionMemberRef )
   )
```

3.37 CollectedMSEs Classes

These classes represent a generic association used to establish the members of the grouping object, CollectionOfMSEs.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.62 NAME 'dlm1CollectedMSEs'
     DESC 'CollectedMSEs is a generic association used to
```

```
            establish the members of the grouping object,
            CollectionOf MSEs.'
      SUP dlm1MemberOfCollection ABSTRACT
    )

    ( 1.3.6.1.4.1.412.100.2.2.188 NAME 'dlmCollectedMSEsCollectionRef'
      DESC 'The grouping or "bag" object that represents the
            Collection.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
    )

    ( 1.3.6.1.4.1.412.100.2.2.189 NAME 'dlmCollectedMSEsMemberRef'
      DESC 'The members of the Collection.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
    )

    ( 1.3.6.1.4.1.412.100.2.1.3.63 NAME 'dlm1CollectedMSEsAuxClass'
      DESC 'CollectedMSEs is a generic association used to
            establish the members of the grouping object,
            CollectionOf MSEs.'
      SUP dlm1CollectedMSEs AUXILIARY
      MAY ( dlmCollectedMSEsCollectionRef $
            dlmCollectedMSEsMemberRef )
    )
```

3.38 Component

This abstract class maps a generic association used to establish 'part of' relationships between managed system elements. For example, the system component association defines parts of a system.

```
    ( 1.3.6.1.4.1.412.100.2.1.3.64 NAME 'dlm1Component'
      DESC 'Component is a generic association used to
            establish "part of" relationships between Managed
            System Elements. For example, the SystemComponent
            association defines parts of a System.'
      SUP top ABSTRACT
    )
```

3.39 SystemComponent Classes

These classes specialize dlmComponent to establish relationships between a system and the managed system elements of which it is composed.

```
    ( 1.3.6.1.4.1.412.100.2.1.3.65 NAME 'dlm1SystemComponent'
      DESC 'SystemComponent is a specialization of the
            Component association that establishes "part of"
            relationships between a System and the Managed System
            Elements of which it is composed.'
      SUP dlm1Component ABSTRACT
    )

    ( 1.3.6.1.4.1.412.100.2.2.190 NAME
'dlmSystemComponentPartComponentRef'
      DESC 'The child element that is a component of a
```

```
            System.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.191 NAME
'dlmSystemComponentGroupComponentRef'
      DESC 'The parent System in the Association.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.66 NAME 'dlm1SystemComponentAuxClass'
     DESC 'SystemComponent is a specialization of the
           Component association that establishes "part of"
           relationships between a System and the Managed System
           Elements of which it is composed.'
     SUP dlm1SystemComponent AUXILIARY
     MAY ( dlmSystemComponentPartComponentRef $
           dlmSystemComponentGroupComponentRef )
   )
```

3.40 SystemDevice Classes

These classes model the aggregation of a LogicalDevices by a System.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.67 NAME 'dlm1SystemDevice'
     DESC 'LogicalDevices may be aggregated by a System.
           This relationship is made explicit by the SystemDevice
          association. '
     SUP dlm1SystemComponent ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.192 NAME 'dlmSystemDevicePartComponentRef'
     DESC 'The LogicalDevice that is a component of a
           System.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.193 NAME
'dlmSystemDeviceGroupComponentRef'
      DESC 'The parent system in the Association.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.68 NAME 'dlm1SystemDeviceAuxClass'
     DESC 'LogicalDevices may be aggregated by a System.
           This relationship is made explicit by the SystemDevice
          association. '
     SUP dlm1SystemDevice AUXILIARY
     MAY ( dlmSystemDevicePartComponentRef $
           dlmSystemDeviceGroupComponentRef )
   )
```

3.41 ServiceComponent Classes

These classes model a set of subordinate services that are aggregated together to form a higher-level service.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.69 NAME 'dlm1ServiceComponent'
     DESC 'The ServiceComponent aggregation models a set of
           subordinate Services that are aggregated together to
           form a higher-level service.'
     SUP dlm1Component ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.194 NAME
'dlmServiceComponentGroupComponentRef'
     DESC 'The parent Service.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.195 NAME
'dlmServiceComponentPartComponentRef'
     DESC 'The component Service.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.70 NAME 'dlm1ServiceComponentAuxClass'
     DESC 'The ServiceComponent aggregation models a set of
           subordinate Services that are aggregated together to
           form a higher-level service.'
     SUP dlm1ServiceComponent AUXILIARY
     MAY ( dlmServiceComponentGroupComponentRef $
           dlmServiceComponentPartComponentRef )
   )
```

3.42 ProductParentChild Classes

These classes define a parent child hierarchy among products. For example, a product may come bundled with other products.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.71 NAME 'dlm1ProductParentChild'
     DESC 'The ProductParentChild association defines a
           parent child hierarchy among Products.  For example, a
          Product may come bundled with other Products. '
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.196 NAME 'dlmProductParentChildChildRef'
     DESC 'The child Product in the association.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.197 NAME 'dlmProductParentChildParentRef'
     DESC 'The parent Product in the association.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )
   ( 1.3.6.1.4.1.412.100.2.1.3.72 NAME 'dlm1ProductParentChildAuxClass'
     DESC 'The ProductParentChild association defines a
           parent child hierarchy among Products.  For example, a
          Product may come bundled with other Products. '
     SUP dlm1ProductParentChild AUXILIARY
     MAY ( dlmProductParentChildChildRef $
           dlmProductParentChildParentRef )
   )
```

3.43 CompatibleProduct Classes

These classes model an association between products can show a wide variety of information. For example, it can show that the two referenced products interoperate, that they can be installed together, that one can be the physical container for the other, etc.

```
  ( 1.3.6.1.4.1.412.100.2.2.198 NAME 'dlmCompatibilityDescription'
     DESC 'CompatibilityDescription is a free-form string
           defining how the two referenced Products interoperate
           or are compatible, any limitations to compatibility,
           etc.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.73 NAME 'dlm1CompatibleProduct'
     DESC 'CompatibleProduct is an association between
           Products that can indicate a wide variety of
           information. For example, it can indicate that the two
          referenced Products interoperate, that they can be
           installed together, that one can be the physical
           container for the other, etc. The string property,
           CompatibilityDescription, defines how the Products
           interoperate or are compatible, any limitations
           regarding interoperability or installation, ...'
     SUP top ABSTRACT
     MAY ( dlmCompatibilityDescription )
   )

   ( 1.3.6.1.4.1.412.100.2.2.199 NAME
'dlmCompatibleProductCompatibleProductRef'
     DESC 'The compatible Product.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.200 NAME 'dlmCompatibleProductProductRef'
     DESC 'The Product for which compatible offerings are
           defined.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.74 NAME 'dlm1CompatibleProductInstance'
     DESC 'CompatibleProduct is an association between
           Products that can indicate a wide variety of
           information. For example, it can indicate that the two
          referenced Products interoperate, that they can be
           installed together, that one can be the physical
           container for the other, etc. The string property,
           CompatibilityDescription, defines how the Products
```

```
            interoperate or are compatible, any limitations
            regarding interoperability or installation, ...'
      SUP dlm1CompatibleProduct
      MAY ( dlmCompatibleProductCompatibleProductRef $
            dlmCompatibleProductProductRef )
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.6 NAME
'dlm1CompatibleProductInstanceNameForm1'
      OC dlm1CompatibleProductInstance
      MUST ( orderedCimKeys )
   )

   ( <core-sr-6> NAME 'dlm1CompatibleProductInstanceStructureRule1'
      Form dlm1CompatibleProductInstanceNameForm1
   )

   ( 1.3.6.1.4.1.412.100.2.2.201 NAME 'dlmCompatibleProductHelperRef'
      DESC 'Pointer to CompatibleProductInstance.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.75 NAME 'dlm1CompatibleProductHelper'
      DESC 'Helper class for finding CompatibleProduct.'
      SUP top AUXILIARY
      MAY ( dlmHelperRefToCompatibleProduct )
   )
```

3.44 ProductProductDependency Classes

These classes model an association between two products, showing that one must be installed, or must be absent, for the other to function. This is conceptually equivalent to the service to service dependency association.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.76 NAME 'dlm1ProductProductDependency'
      DESC 'ProductProductDependency is an association
            between two Products, indicating that one must be
            installed, or must be absent, for the other to
            function. This is conceptually equivalent to the
            ServiceServiceDependency association.'
      SUP top ABSTRACT
      MAY ( dlmTypeOfDependency )
   )

   ( 1.3.6.1.4.1.412.100.2.2.202 NAME
'dlmProductProductDependencyDependentProductRef'
      DESC 'The Product that is dependent on another Product.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.203 NAME
'dlmProductProductDependencyRequiredProductRef'
      DESC 'The required Product.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )
   ( 1.3.6.1.4.1.412.100.2.1.3.77 NAME
'dlm1ProductProductDependencyInstance'
      DESC 'ProductProductDependency is an association
            between two Products, indicating that one must be
            installed, or must be absent, for the other to
            function. This is conceptually equivalent to the
            ServiceServiceDependency association.'
      SUP dlm1ProductProductDependency
      MAY ( dlmProductProductDependencyDependentProductRef $
            dlmProductProductDependencyRequiredProductRef )
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.7 NAME
'dlm1ProductProductDependencyInstanceNameForm1'
      OC dlm1ProductProductDependencyInstance
      MUST ( orderedCimKeys )
   )

   ( <core-sr-7> NAME
'dlm1ProductProductDependencyInstanceStructureRule1'
      Form dlm1ProductProductDependencyInstanceNameForm1
   )

   ( 1.3.6.1.4.1.412.100.2.2.204 NAME
'dlmProductProductDependencyHelperRef'
      DESC 'Pointer to ProductProductDependencyInstance.'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.78 NAME
'dlm1ProductProductDependencyHelper'
      DESC 'Helper class for finding
            ProductProductDependency.'
      SUP top AUXILIARY
      MAY ( dlmHelperRefToProductProductDependency )
   )
```

3.45 ProductSupport Classes

This classes represent the association between products and support access that conveys how support is obtained for the product. This is a many-to-many relationship, implying that various types of support are available for a product, and that the same support object can provide help for multiple products. This class defines two attributes that are self-explanatory.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.79 NAME 'dlm1ProductSupport'
     DESC 'ProductSupport is an association between Product
           and SupportAccess that conveys how support is obtained
          for the Product.  This is a many-to-many relationship,
           implying that various types of Support are available
           for a Product, and that the same Support object can
           provide assistance for multiple Products.'
     SUP top ABSTRACT
   )
```

```
( 1.3.6.1.4.1.412.100.2.2.205 NAME 'dlmProductSupportProductRef'
  DESC 'The Product.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  EQUALITY distinguishedNameMatch
)

( 1.3.6.1.4.1.412.100.2.2.206 NAME 'dlmProductSupportSupportRef'
  DESC 'Support for the Product.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  EQUALITY distinguishedNameMatch
)

( 1.3.6.1.4.1.412.100.2.1.3.80 NAME 'dlm1ProductSupportAuxClass'
  DESC 'ProductSupport is an association between Product
        and SupportAccess that conveys how support is obtained
       for the Product.  This is a many-to-many relationship,
        implying that various types of Support are available
        for a Product, and that the same Support object can
        provide assistance for multiple Products.'
  SUP dlm1ProductSupport AUXILIARY
  MAY ( dlmProductSupportProductRef $
        dlmProductSupportSupportRef )
)
```

3.46 ProductFRU Classes

These classes provides information regarding what product components have been or are being replaced.

```
( 1.3.6.1.4.1.412.100.2.1.3.81 NAME 'dlm1ProductFRU'
  DESC 'ProductFRU is an association between Product and
        FRU that provides information regarding what Product
        components have been or are being replaced.  The
        association is one to many, conveying that a Product
        can have many FRUs, and that a particular instance of
        a FRU is only applied to one (instance of a) Product.'
  SUP top ABSTRACT
)

( 1.3.6.1.4.1.412.100.2.2.207 NAME 'dlmProductFRUFRURef'
  DESC 'The FRU.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  EQUALITY distinguishedNameMatch
)

( 1.3.6.1.4.1.412.100.2.2.208 NAME 'dlmProductFRUProductRef'
  DESC 'The Product to which the FRU is applied.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
)

( 1.3.6.1.4.1.412.100.2.1.3.82 NAME 'dlm1ProductFRUAuxClass'
  DESC 'ProductFRU is an association between Product and
        FRU that provides information regarding what Product
        components have been or are being replaced.  The
        association is one to many, conveying that a Product
        can have many FRUs, and that a particular instance of
        a FRU is only applied to one (instance of a) Product.'
```

```
     SUP dlm1ProductFRU AUXILIARY
     MAY ( dlmProductFRUFRURef $ dlmProductFRUProductRef )
   )
```

3.47 ProductPhysicalElements Classes

These classes show the physical elements that make up a product.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.83 NAME 'dlm1ProductPhysicalElements'
     DESC 'Indicates the PhysicalElements that make up a
           Product.'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.209 NAME
'dlmProductPhysicalElementsComponentRef'
     DESC 'The PhysicalElement which is a part of the
           Product.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.2.210 NAME
'dlmProductPhysicalElementsProductRef'
     DESC 'The Product.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.84 NAME
'dlm1ProductPhysicalElementsAuxClass'
     DESC 'Indicates the PhysicalElements that make up a
           Product.'
     SUP dlm1ProductPhysicalElements AUXILIARY
     MAY ( dlmProductPhysicalElementsComponentRef $
           dlmProductPhysicalElementsProductRef )
   )
```

3.48 FRUPhysicalElements Classes

These classes show the physical elements that make up a FRU.

```
   ( 1.3.6.1.4.1.412.100.2.1.3.85 NAME 'dlm1FRUPhysicalElements'
     DESC 'Indicates the PhysicalElements that make up a
           FRU.'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.211 NAME 'dlmFRUPhysicalElementsFRURef'
     DESC 'The FRU.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.212 NAME
'dlmFRUPhysicalElementsComponentRef'
     DESC 'The PhysicalElement which is a part of the FRU.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )
   ( 1.3.6.1.4.1.412.100.2.1.3.86 NAME
'dlm1FRUPhysicalElementsAuxClass'
     DESC 'Indicates the PhysicalElements that make up a
           FRU.'
     SUP dlm1FRUPhysicalElements AUXILIARY
     MAY ( dlmFRUPhysicalElementsFRURef $
           dlmFRUPhysicalElementsComponentRef )
   )
```

3.49 FRUIncludesProduct Classes

These classes show that a FRU may be composed of other product(s).

```
   ( 1.3.6.1.4.1.412.100.2.1.3.87 NAME 'dlm1FRUIncludesProduct'
     DESC 'Indicates that a FRU may be composed of other
           Product(s).'
     SUP top ABSTRACT
   )

   ( 1.3.6.1.4.1.412.100.2.2.213 NAME 'dlmFRUIncludesProductFRURef'
     DESC 'The FRU.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.214 NAME
'dlmFRUIncludesProductComponentRef'
     DESC 'The Product which is a part of the FRU.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.88 NAME 'dlm1FRUIncludesProductAuxClass'
     DESC 'Indicates that a FRU may be composed of other
           Product(s).'
     SUP dlm1FRUIncludesProduct AUXILIARY
     MAY ( dlmFRUIncludesProductFRURef $
           dlmFRUIncludesProductComponentRef )
   )
```

3.50 Synchronized Classes

These classes indicate that two logical elements were aligned or made to be equivalent at the specified point in time. Preservation of synchronization is determined by the value of the dlmSyncMaintained attribute.

```
   ( 1.3.6.1.4.1.412.100.2.2.215 NAME 'dlmSyncMaintained'
     DESC 'Boolean indicating whether synchronization is
           maintained.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.216 NAME 'dlmWhenSynced'
     DESC 'The point in time that the Elements were
           synchronized.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE
   )
   ( 1.3.6.1.4.1.412.100.2.1.3.89 NAME 'dlm1Synchronized'
     DESC 'Indicates that two LogicalElements were aligned
           or made to be equivalent at the specified point in
           time. If the boolean property SyncMaintained is TRUE,
           then synchronization of the Elements is preserved.
           Both like and unlike objects may be synchronized. For
           example, two WatchDog timers may be aligned, or the
           contents of a LogicalFile may be synchronized with the
          contents of a StorageExtent.'
     SUP top ABSTRACT
     MAY ( dlmSyncMaintained $ dlmWhenSynced )
   )

   ( 1.3.6.1.4.1.412.100.2.2.217 NAME 'dlmSynchronizedSyncedElementRef'
     DESC 'SyncedElement represents another LogicalElement
           that is synchronized with the entity referenced as
           SystemElement.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.2.218 NAME 'dlmSynchronizedSystemElementRef'
     DESC 'SystemElement represents one LogicalElement that
           is synchronized with the entity referenced as
           SyncedElement.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE
   )

   ( 1.3.6.1.4.1.412.100.2.1.3.90 NAME 'dlm1SynchronizedInstance'
     DESC 'Indicates that two LogicalElements were aligned
           or made to be equivalent at the specified point in
           time. If the boolean property SyncMaintained is TRUE,
           then synchronization of the Elements is preserved.
           Both like and unlike objects may be synchronized. For
           example, two WatchDog timers may be aligned, or the
           contents of a LogicalFile may be synchronized with the
          contents of a StorageExtent.'
     SUP dlm1Synchronized
     MAY ( dlmSynchronizedSyncedElementRef $
           dlmSynchronizedSystemElementRef )
   )

   ( 1.3.6.1.4.1.412.100.2.3.3.8 NAME
'dlm1SynchronizedInstanceNameForm1'
     OC dlm1SynchronizedInstance
     MUST ( orderedCimKeys )
   )

   ( <core-sr-8> NAME 'dlm1SynchronizedInstanceStructureRule1'
     Form dlm1SynchronizedInstanceNameForm1
   )

   ( 1.3.6.1.4.1.412.100.2.2.219 NAME 'dlmSynchronizedHelperRef'
     DESC 'Pointer to SynchronizedInstance.'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     EQUALITY distinguishedNameMatch
   )
   ( 1.3.6.1.4.1.412.100.2.1.3.91 NAME 'dlm1SynchronizedHelper'
     DESC 'Helper class for finding Synchronized.'
     SUP top AUXILIARY
     MAY ( dlmSynchronizedHelperRef )
   )
```

4. References

Request For Comments (RFC) and Internet Draft documents are available from numerous mirror sites.

[1] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)," RFC 2251, December 1997.

[2] M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions," RFC 2252, December 1997.

[3] CIM, "CIM Core Model, v2.4," http://www.dmtf.org/spec/cims.html.

[4] F. Yergeau, "UTF-8, a transformation format of ISO 10646," RFC 2279, January 1998.

[5] DMTF, "LDAP Mapping Guidelines", http://www.dmtf.org/spec/denh.html

[6] M. Wahl, S. Kille, T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names," RFC 2253, December 1997

5. Acknowledgment

This work is a product of the DMTF LDAP Mapping Working Group and has benefited from many comments and discussions during this group's meetings.

A. Structural Rules

The following table states the structural rules defined in this document.

Rule	Structural Class	RDN Attribute	Superior Rules	Defined
<core-sr-1>	dlm1ConfigurationInstance	orderedCimKeys	*	3.13
<core-sr-2>	dlm1ProductInstance	orderedCimKeys	*	3.15
<core-sr-3>	dlm1SupportAccessInstance	orderedCimKeys	*	3.16
<core-sr-4>	dlm1FRUInstance	orderedCimKeys	*	3.17
<core-sr-5>	dlm1ServiceServiceDependencyInstance	orderedCimKeys	*	3.32
<core-sr-6>	dlm1CompatibleProductInstance	orderedCimKeys	*	3.43
<core-sr-7>	dlm1ProductProductDependencyInstance	orderedCimKeys	*	3.44
<core-sr-8>	dlm1SynchronizedInstance	orderedCimKeys	*	3.50
<core-sr-9>	dlmOtherIdentifyingInfoInstance	arrayIndex	**	2.3.2

* This mapping document does not provide suggestions regarding DIT placement of mapped top-level CIM objects.

** The superiors for this rule are not defined in this mapping. In subsequent DMTF CIM mapping documents that define mappings of non-abstract subclasses of CIM_ComputerSystem and CIM_LogicalDevice, it will be possible to define the possible superiors for cimOtherIdentifyingInfoInstance.

B. OID Assignments

The following three tables provides the summary of OID assignments made in this document

B.1 Object Classes

OID	Object Class	Section
1.3.6.1.4.1.412.100.2.1.3.92	dlmOtherIdentifyingInfoInstance	2.3.1
1.3.6.1.4.1.412.100.2.1.3.1	dlm1ManagedElement	3.1
1.3.6.1.4.1.412.100.2.1.3.2	dlm1ManagedSystemElement	3.2
1.3.6.1.4.1.412.100.2.1.3.3	dlm1PhysicalElement	3.3
1.3.6.1.4.1.412.100.2.1.3.4	dlm1LogicalElement	3.4
1.3.6.1.4.1.412.100.2.1.3.5	dlm1System	3.5
1.3.6.1.4.1.412.100.2.1.3.6	dlm1ComputerSystem	3.6
1.3.6.1.4.1.412.100.2.1.3.93	dlm1AdminDomain	3.7
1.3.6.1.4.1.412.100.2.1.3.7	dlm1LogicalDevice	3.8
1.3.6.1.4.1.412.100.2.1.3.8	dlm1Service	3.9
1.3.6.1.4.1.412.100.2.1.3.9	dlm1ServiceAccessPoint	3.10
1.3.6.1.4.1.412.100.2.1.3.10	dlm1Collection	3.11
1.3.6.1.4.1.412.100.2.1.3.11	dlm1CollectionOfMSEs	3.12
1.3.6.1.4.1.412.100.2.1.3.12	dlm1Configuration	3.13
1.3.6.1.4.1.412.100.2.1.3.13	dlm1ConfigurationAuxClass	3.13
1.3.6.1.4.1.412.100.2.1.3.14	dlm1ConfigurationInstance	3.13
1.3.6.1.4.1.412.100.2.1.3.15	dlm1Setting	3.14
1.3.6.1.4.1.412.100.2.1.3.16	dlm1Product	3.15
1.3.6.1.4.1.412.100.2.1.3.17	dlm1ProductAuxClass	3.15
1.3.6.1.4.1.412.100.2.1.3.18	dlm1ProductInstance	3.15
1.3.6.1.4.1.412.100.2.1.3.19	dlm1SupportAccess	3.16
1.3.6.1.4.1.412.100.2.1.3.20	dlm1SupportAccessAuxClass	3.16
1.3.6.1.4.1.412.100.2.1.3.21	dlm1SupportAccessInstance	3.16
1.3.6.1.4.1.412.100.2.1.3.22	dlm1FRU	3.17
1.3.6.1.4.1.412.100.2.1.3.23	dlm1FRUAuxClass	3.17
1.3.6.1.4.1.412.100.2.1.3.24	dlm1FRUInstance	3.17

1.3.6.1.4.1.412.100.2.1.3.25	dlm1CollectedCollections	3.18
1.3.6.1.4.1.412.100.2.1.3.26	dlm1CollectedCollectionsAuxClass	3.18
1.3.6.1.4.1.412.100.2.1.3.27	dlm1LogicalIdentity	3.19
1.3.6.1.4.1.412.100.2.1.3.28	dlm1ConfigurationComponent	3.20
1.3.6.1.4.1.412.100.2.1.3.29	dlm1ConfigurationComponentAuxClass	3.20
1.3.6.1.4.1.412.100.2.1.3.30	dlm1ElementConfiguration	3.21
1.3.6.1.4.1.412.100.2.1.3.31	dlm1ElementConfigurationAuxClass	3.21
1.3.6.1.4.1.412.100.2.1.3.32	dlm1CollectionConfiguration	3.22
1.3.6.1.4.1.412.100.2.1.3.33	dlm1CollectionConfigurationAuxClass	3.22
1.3.6.1.4.1.412.100.2.1.3.34	dlm1ElementSetting	3.23
1.3.6.1.4.1.412.100.2.1.3.35	dlm1ElementSettingAuxClass	3.23
1.3.6.1.4.1.412.100.2.1.3.36	dlm1DefaultSetting	3.24
1.3.6.1.4.1.412.100.2.1.3.37	dlm1DefaultSettingAuxClass	3.24
1.3.6.1.4.1.412.100.2.1.3.38	dlm1SettingContext	3.25
1.3.6.1.4.1.412.100.2.1.3.39	dlm1SettingContextAuxClass	3.25
1.3.6.1.4.1.412.100.2.1.3.40	dlm1CollectionSetting	3.26
1.3.6.1.4.1.412.100.2.1.3.41	dlm1CollectionSettingAuxClass	3.26
1.3.6.1.4.1.412.100.2.1.3.42	dlm1Dependency	3.27
1.3.6.1.4.1.412.100.2.1.3.43	dlm1ServiceAccessBySAP	3.28
1.3.6.1.4.1.412.100.2.1.3.44	dlm1ServiceAccessBySAPAuxClass	3.28
1.3.6.1.4.1.412.100.2.1.3.45	dlm1HostedService	3.29
1.3.6.1.4.1.412.100.2.1.3.46	dlm1HostedServiceAuxClass	3.29
1.3.6.1.4.1.412.100.2.1.3.47	dlm1HostedAccessPoint	3.30
1.3.6.1.4.1.412.100.2.1.3.48	dlm1HostedAccessPointAuxClass	3.30
1.3.6.1.4.1.412.100.2.1.3.49	dlm1ProvidesServiceToElement	3.31
1.3.6.1.4.1.412.100.2.1.3.50	dlm1ProvidesServiceToElementAuxClass	3.31
1.3.6.1.4.1.412.100.2.1.3.51	dlm1ServiceServiceDependency	3.32
1.3.6.1.4.1.412.100.2.1.3.52	dlm1ServiceServiceDependencyInstance	3.32
1.3.6.1.4.1.412.100.2.1.3.53	dlm1ServiceServiceDependencyHelper	3.32
1.3.6.1.4.1.412.100.2.1.3.54	dlm1ServiceSAPDependency	3.33
1.3.6.1.4.1.412.100.2.1.3.55	dlm1ServiceSAPDependencyAuxClass	3.33
1.3.6.1.4.1.412.100.2.1.3.56	dlm1SAPSAPDependency	3.34
1.3.6.1.4.1.412.100.2.1.3.57	dlm1SAPSAPDependencyAuxClass	3.34
1.3.6.1.4.1.412.100.2.1.3.58	dlm1Realizes	3.35
1.3.6.1.4.1.412.100.2.1.3.59	dlm1RealizesAuxClass	3.35
1.3.6.1.4.1.412.100.2.1.3.60	dlm1MemberOfCollection	3.36
1.3.6.1.4.1.412.100.2.1.3.61	dlm1MemberOfCollectionAuxClass	3.36
1.3.6.1.4.1.412.100.2.1.3.62	dlm1CollectedMSEs	3.37
1.3.6.1.4.1.412.100.2.1.3.63	dlm1CollectedMSEsAuxClass	3.37
1.3.6.1.4.1.412.100.2.1.3.64	dlm1Component	3.38

1.3.6.1.4.1.412.100.2.1.3.65	dlm1SystemComponent	3.39
1.3.6.1.4.1.412.100.2.1.3.66	dlm1SystemComponentAuxClass	3.39
1.3.6.1.4.1.412.100.2.1.3.67	dlm1SystemDevice	3.40
1.3.6.1.4.1.412.100.2.1.3.68	dlm1SystemDeviceAuxClass	3.40
1.3.6.1.4.1.412.100.2.1.3.69	dlm1ServiceComponent	3.41
1.3.6.1.4.1.412.100.2.1.3.70	dlm1ServiceComponentAuxClass	3.41
1.3.6.1.4.1.412.100.2.1.3.71	dlm1ProductParentChild	3.42
1.3.6.1.4.1.412.100.2.1.3.72	dlm1ProductParentChildAuxClass	3.42
1.3.6.1.4.1.412.100.2.1.3.73	dlm1CompatibleProduct	3.43
1.3.6.1.4.1.412.100.2.1.3.74	dlm1CompatibleProductInstance	3.43
1.3.6.1.4.1.412.100.2.1.3.75	dlm1CompatibleProductHelper	3.43
1.3.6.1.4.1.412.100.2.1.3.76	dlm1ProductProductDependency	3.44
1.3.6.1.4.1.412.100.2.1.3.77	dlm1ProductProductDependencyInstance	3.44
1.3.6.1.4.1.412.100.2.1.3.78	dlm1ProductProductDependencyHelper	3.44
1.3.6.1.4.1.412.100.2.1.3.79	dlm1ProductSupport	3.45
1.3.6.1.4.1.412.100.2.1.3.80	dlm1ProductSupportAuxClass	3.45
1.3.6.1.4.1.412.100.2.1.3.81	dlm1ProductFRU	3.46
1.3.6.1.4.1.412.100.2.1.3.82	dlm1ProductFRUAuxClass	3.46
1.3.6.1.4.1.412.100.2.1.3.83	dlm1ProductPhysicalElements	3.47
1.3.6.1.4.1.412.100.2.1.3.84	dlm1ProductPhysicalElementsAuxClass	3.47
1.3.6.1.4.1.412.100.2.1.3.85	dlm1FRUPhysicalElements	3.48
1.3.6.1.4.1.412.100.2.1.3.86	dlm1FRUPhysicalElementsAuxClass	3.48
1.3.6.1.4.1.412.100.2.1.3.87	dlm1FRUIncludesProduct	3.49
1.3.6.1.4.1.412.100.2.1.3.88	dlm1FRUIncludesProductAuxClass	3.49
1.3.6.1.4.1.412.100.2.1.3.89	dlm1Synchronized	3.50
1.3.6.1.4.1.412.100.2.1.3.90	dlm1SynchronizedInstance	3.50
1.3.6.1.4.1.412.100.2.1.3.91	dlm1SynchronizedHelper	3.50

B.2 Attributes

OID	Attribute	Section
1.3.6.1.4.1.412.100.1.2.5	arrayIndex	2.3.1
1.3.6.1.4.1.412.100.2.2.101	dlmIdentifyingDescription	2.3.1
1.3.6.1.4.1.412.100.1.2.1	orderedCimKeys	2.4
1.3.6.1.4.1.412.100.2.2.103	dlmCaption	3.1
1.3.6.1.4.1.412.100.2.2.104	dlmDescription	3.1
1.3.6.1.4.1.412.100.2.2.105	dlmInstallDate	3.2
1.3.6.1.4.1.412.100.2.2.106	dlmName	3.2
1.3.6.1.4.1.412.100.2.2.107	dlmStatus	3.2
1.3.6.1.4.1.412.100.2.2.108	dlmCreationClassName	3.3
1.3.6.1.4.1.412.100.2.2.109	dlmManufactureDate	3.3

1.3.6.1.4.1.412.100.2.2.110	dlmManufacturer	3.3
1.3.6.1.4.1.412.100.2.2.111	dlmModel	3.3
1.3.6.1.4.1.412.100.2.2.112	dlmOtherIdentifyingInfo	3.3
1.3.6.1.4.1.412.100.2.2.113	dlmPartNumber	3.3
1.3.6.1.4.1.412.100.2.2.114	dlmPoweredOn	3.3
1.3.6.1.4.1.412.100.2.2.115	dlmSKU	3.3
1.3.6.1.4.1.412.100.2.2.116	dlmSerialNumber	3.3
1.3.6.1.4.1.412.100.2.2.117	dlmTag	3.3
1.3.6.1.4.1.412.100.2.2.118	dlmVersion	3.3
1.3.6.1.4.1.412.100.2.2.119	dlmNameFormat	3.5
1.3.6.1.4.1.412.100.2.2.120	dlmPrimaryOwnerContact	3.5
1.3.6.1.4.1.412.100.2.2.121	dlmPrimaryOwnerName	3.5
1.3.6.1.4.1.412.100.2.2.122	dlmRoles	3.5
1.3.6.1.4.1.412.100.2.2.123	dlmDedicated	3.6
1.3.6.1.4.1.412.100.2.2.124	dlmAdditionalAvailability	3.8
1.3.6.1.4.1.412.100.2.2.125	dlmAvailability	3.8
1.3.6.1.4.1.412.100.2.2.126	dlmDeviceID	3.8
1.3.6.1.4.1.412.100.2.2.127	dlmErrorCleared	3.8
1.3.6.1.4.1.412.100.2.2.128	dlmErrorDescription	3.8
1.3.6.1.4.1.412.100.2.2.129	dlmLastErrorCode	3.8
1.3.6.1.4.1.412.100.2.2.130	dlmMaxQuiesceTime	3.8
1.3.6.1.4.1.412.100.2.2.131	dlmPowerManagementCapabilities	3.8
1.3.6.1.4.1.412.100.2.2.132	dlmPowerManagementSupported	3.8
1.3.6.1.4.1.412.100.2.2.133	dlmPowerOnHours	3.8
1.3.6.1.4.1.412.100.2.2.134	dlmStatusInfo	3.8
1.3.6.1.4.1.412.100.2.2.135	dlmTotalPowerOnHours	3.8
1.3.6.1.4.1.412.100.2.2.136	dlmStartMode	3.9
1.3.6.1.4.1.412.100.2.2.137	dlmStarted	3.9
1.3.6.1.4.1.412.100.2.2.138	dlmCollectionID	3.12
1.3.6.1.4.1.412.100.2.2.139	dlmSettingID	3.14
1.3.6.1.4.1.412.100.2.2.140	dlmIdentifyingNumber	3.15
1.3.6.1.4.1.412.100.2.2.141	dlmSKUNumber	3.15
1.3.6.1.4.1.412.100.2.2.142	dlmVendor	3.15
1.3.6.1.4.1.412.100.2.2.143	dlmWarrantyDuration	3.15
1.3.6.1.4.1.412.100.2.2.144	dlmWarrantyStartDate	3.15
1.3.6.1.4.1.412.100.2.2.145	dlmCommunicationInfo	3.16
1.3.6.1.4.1.412.100.2.2.146	dlmCommunicationMode	3.16
1.3.6.1.4.1.412.100.2.2.147	dlmLocale	3.16
1.3.6.1.4.1.412.100.2.2.148	dlmSupportAccessId	3.16
1.3.6.1.4.1.412.100.2.2.149	dlmFRUNumber	3.17

1.3.6.1.4.1.412.100.2.2.150	dlmRevisionLevel	3.17
1.3.6.1.4.1.412.100.2.2.151	dlmCollectedCollectionsCollectionRef	3.18
1.3.6.1.4.1.412.100.2.2.152	dlmCollectedCollectionsCollectionInCollectionRef	3.18
1.3.6.1.4.1.412.100.2.2.153	dlmConfigurationComponentConfigComponentRef	3.20
1.3.6.1.4.1.412.100.2.2.154	dlmConfigurationComponentConfigGroupRef	3.20
1.3.6.1.4.1.412.100.2.2.155	dlmElementConfigurationConfigurationRef	3.21
1.3.6.1.4.1.412.100.2.2.156	dlmElementConfigurationElementRef	3.21
1.3.6.1.4.1.412.100.2.2.157	dlmCollectionConfigurationCollectionRef	3.22
1.3.6.1.4.1.412.100.2.2.158	dlmCollectionConfigurationConfigurationRef	3.22
1.3.6.1.4.1.412.100.2.2.159	dlmElementSettingElementRef	3.23
1.3.6.1.4.1.412.100.2.2.160	dlmElementSettingSettingRef	3.23
1.3.6.1.4.1.412.100.2.2.161	dlmDefaultSettingElementRef	3.24
1.3.6.1.4.1.412.100.2.2.162	dlmDefaultSettingSettingRef	3.24
1.3.6.1.4.1.412.100.2.2.163	dlmSettingContextContextRef	3.25
1.3.6.1.4.1.412.100.2.2.164	dlmSettingContextSettingRef	3.25
1.3.6.1.4.1.412.100.2.2.165	dlmCollectionSettingCollectionRef	3.26
1.3.6.1.4.1.412.100.2.2.166	dlmCollectionSettingSettingRef	3.26
1.3.6.1.4.1.412.100.2.2.167	dlmServiceAccessBySAPAntecedentRef	3.28
1.3.6.1.4.1.412.100.2.2.168	dlmServiceAccessBySAPDependentRef	3.28
1.3.6.1.4.1.412.100.2.2.169	dlmHostedServiceDependentRef	3.29
1.3.6.1.4.1.412.100.2.2.170	dlmHostedServiceAntecedentRef	3.29
1.3.6.1.4.1.412.100.2.2.171	dlmHostedAccessPointDependentRef	3.30
1.3.6.1.4.1.412.100.2.2.172	dlmHostedAccessPointAntecedentRef	3.30
1.3.6.1.4.1.412.100.2.2.173	dlmProvidesServiceToElementDependentRef	3.31
1.3.6.1.4.1.412.100.2.2.174	dlmProvidesServiceToElementAntecedentRef	3.31
1.3.6.1.4.1.412.100.2.2.175	dlmRestartService	3.32
1.3.6.1.4.1.412.100.2.2.176	dlmTypeOfDependency	3.32
1.3.6.1.4.1.412.100.2.2.177	dlmServiceServiceDependencyAntecedentRef	3.32
1.3.6.1.4.1.412.100.2.2.178	dlmServiceServiceDependencyDependentRef	3.32
1.3.6.1.4.1.412.100.2.2.179	dlmServiceServiceDependencyHelperRef	3.32
1.3.6.1.4.1.412.100.2.2.180	dlmServiceSAPDependencyDependentRef	3.33
1.3.6.1.4.1.412.100.2.2.181	dlmServiceSAPDependencyAntecedentRef	3.33
1.3.6.1.4.1.412.100.2.2.182	dlmSAPSAPDependencyAntecedentRef	3.34
1.3.6.1.4.1.412.100.2.2.183	dlmSAPSAPDependencyDependentRef	3.34
1.3.6.1.4.1.412.100.2.2.184	dlmRealizesDependentRef	3.35
1.3.6.1.4.1.412.100.2.2.185	dlmRealizesAntecedentRef	3.35
1.3.6.1.4.1.412.100.2.2.186	dlmMemberOfCollectionCollectionRef	3.36
1.3.6.1.4.1.412.100.2.2.187	dlmMemberOfCollectionMemberRef	3.36
1.3.6.1.4.1.412.100.2.2.188	dlmCollectedMSEsCollectionRef	3.37
1.3.6.1.4.1.412.100.2.2.189	dlmCollectedMSEsMemberRef	3.37

1.3.6.1.4.1.412.100.2.2.190	dlmSystemComponentPartComponentRef	3.39
1.3.6.1.4.1.412.100.2.2.191	dlmSystemComponentGroupComponentRef	3.39
1.3.6.1.4.1.412.100.2.2.192	dlmSystemDevicePartComponentRef	3.40
1.3.6.1.4.1.412.100.2.2.193	dlmSystemDeviceGroupComponentRef	3.40
1.3.6.1.4.1.412.100.2.2.194	dlmServiceComponentGroupComponentRef	3.41
1.3.6.1.4.1.412.100.2.2.195	dlmServiceComponentPartComponentRef	3.41
1.3.6.1.4.1.412.100.2.2.196	dlmProductParentChildChildRef	3.42
1.3.6.1.4.1.412.100.2.2.197	dlmProductParentChildParentRef	3.42
1.3.6.1.4.1.412.100.2.2.198	dlmCompatibilityDescription	3.43
1.3.6.1.4.1.412.100.2.2.199	dlmCompatibleProductCompatibleProductRef	3.43
1.3.6.1.4.1.412.100.2.2.200	dlmCompatibleProductProductRef	3.43
1.3.6.1.4.1.412.100.2.2.201	dlmCompatibleProductHelperRef	3.43
1.3.6.1.4.1.412.100.2.2.202	dlmProductProductDependencyDependentProductRef	3.44
1.3.6.1.4.1.412.100.2.2.203	dlmProductProductDependencyRequiredProductRef	3.44
1.3.6.1.4.1.412.100.2.2.204	dlmProductProductDependencyHelperRef	3.44
1.3.6.1.4.1.412.100.2.2.205	dlmProductSupportProductRef	3.45
1.3.6.1.4.1.412.100.2.2.206	dlmProductSupportSupportRef	3.45
1.3.6.1.4.1.412.100.2.2.207	dlmProductFRUFRURef	3.46
1.3.6.1.4.1.412.100.2.2.208	dlmProductFRUProductRef	3.46
1.3.6.1.4.1.412.100.2.2.209	dlmProductPhysicalElementsComponentRef	3.47
1.3.6.1.4.1.412.100.2.2.210	dlmProductPhysicalElementsProductRef	3.47
1.3.6.1.4.1.412.100.2.2.211	dlmFRUPhysicalElementsFRURef	3.48
1.3.6.1.4.1.412.100.2.2.212	dlmFRUPhysicalElementsComponentRef	3.48
1.3.6.1.4.1.412.100.2.2.213	dlmFRUIncludesProductFRURef	3.49
1.3.6.1.4.1.412.100.2.2.214	dlmFRUIncludesProductComponentRef	3.49
1.3.6.1.4.1.412.100.2.2.215	dlmSyncMaintained	3.50
1.3.6.1.4.1.412.100.2.2.216	dlmWhenSynced	3.50
1.3.6.1.4.1.412.100.2.2.217	dlmSynchronizedSyncedElementRef	3.50
1.3.6.1.4.1.412.100.2.2.218	dlmSynchronizedSystemElementRef	3.50
1.3.6.1.4.1.412.100.2.2.219	dlmSynchronizedHelperRef	3.50

B.3 Nameforms

OID	Nameform	Section
1.3.6.1.4.1.412.100.2.3.3.9	dlmOtherIdentifyingInfoInstanceNameForm	2.3.1
1.3.6.1.4.1.412.100.2.3.3.1	dlm1ConfigurationInstanceNameForm1	3.13
1.3.6.1.4.1.412.100.2.3.3.2	dlm1ProductInstanceNameForm1	3.15
1.3.6.1.4.1.412.100.2.3.3.3	dlm1SupportAccessInstanceNameForm1	3.16
1.3.6.1.4.1.412.100.2.3.3.4	dlm1FRUInstanceNameForm1	3.17
1.3.6.1.4.1.412.100.2.3.3.5	dlm1ServiceServiceDependencyInstanceNameForm1	3.32
1.3.6.1.4.1.412.100.2.3.3.6	dlm1CompatibleProductInstanceNameForm1	3.43

1.3.6.1.4.1.412.100.2.3.3.7	dlm1ProductProductDependencyInstanceNameForm1	3.44
1.3.6.1.4.1.412.100.2.3.3.8	dlm1SynchronizedInstanceNameForm1	3.50