Internet-Draft
draft-moats-dmtf-physical-ldap-01.txt
Expires in six months

Ryan Moats
Gerald Maziarski
AT&T
John Strassner
cisco Systems
December 1999

LDAP Schema for the DMTF Physical CIM v2.2 Model
Filename: draft-moats-dmtf-physical-ldap-01.txt

Status of this Memo

Abstract

This draft presents a LDAP schema for the DMTF CIM Physical model version 2.2 [4].

1. Introduction

This draft presents a LDAPv3 [1,2] schema for the DMTF CIM Physical model. Associations are mapped using a combination of auxiliary classes and DIT structure rules. Where auxiliary classes are used, name form and DIT content rules are specified.

This document is not a product of the DMTF, and represents the view of the authors.

2. LDAP Mapping Considerations

There are several special considerations in mapping the physical model from CIM to LDAP. They are discussed in this section.

2.1 Modifications to cimAssociationInstance

The core mapping [5] defined cimAssociationInstance as a helper class. To support the auxiliary classes, the following classes should be added to cimAssociationInstance's content rule:

```
cim22PhysicalElementLocationAuxClass
cim22ElementCapacityAuxClass
cim22ParticipatesInSetAuxClass
cim22ContainerAuxClass
cim22ChassisInRackAuxClass
cim22PackageInChassisAuxClass
cim22DockedAuxClass
cim22CardOnCardAuxClass
cim22DeviceServicesLocationAuxClass
cim22PackagedComponentAuxClass
cim22MemoryOnCardAuxClass
cim22MemoryWithMediaAuxClass
cim22PhysicalMediaInLocationAuxClass
cim22ElementsLinkedAuxClass
cim22ConnectedToAuxClass
cim22SlotInSlotAuxClass
cim22AdjacentSlotsAuxClass
cim22PackageInConnectorAuxClass
cim22PackageInSlotAuxClass
cim22CardInSlotAuxClass
cim22LinkHasConnectorAuxClass
cim22ConnectorOnPackageAuxClass
cim22AdapterActiveConnectionAuxClass
cim22ComputerSystemPackageAuxClass
cim22LibraryPackageAuxClass
cim22PackageCoolingAuxClass
cim22PackageTempSensorAuxClass
cim22PackageAlarmAuxClass
```

Also, the following structure rules defined here need to be added to the structure rule for cimAssociationInstance: <sr12>, <sr13>, <sr14>, <sr15>, <sr16>, <sr17>, <sr18>, <sr19>, and <sr20>.

2.2 cimServicePhilosophyInstance

The class cimPhysicalFrame defines two linked indexed arrays: ServicePhilosophy and ServiceDescription. In the LDAP mapping, these

are replaced with separate instances of cimServicePhilosophyInstance, DIT contained by cimPhysicalFrame.

```
( <oid-at98> NAME 'cimServicePhilosophy'
  DESC 'ServicePhilosophy is an enumerated, integer value that
        indicates whether the Frame is serviced from the top
        (value=2), front (3), back (4) or side (5), whether it has
        sliding trays (6) or removable sides (7), and/or whether
        the Frame is moveable (8), for example, having rollers.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-at99> NAME 'cimServiceDescriptions'n
  DESC 'A free-form strings providing more detailed explanations
        for this entries.
  SYNTAX string SINGLE-VALUE
)

( <oid-oc92> NAME 'cimServicePhilosophyInstance'
  DESC 'helper class to tie ServicePhilosophy and
        ServiceDescriptions in PhysicalFrame together'
  SUP top
  MUST (arrayIndex)
  MAY (cimServicePhilosophy $ cimServiceDescription)
)

( <oid-nf21> NAME 'cimServicePhilosophyInstanceNameForm'
  OC cimServicePhilosophyInstance
  MUST (arrayIndex)
)

( <sr21> NAME 'cimServicePhilosophyInstanceStructureRule'
  FORM cimServicePhilosophyInstanceNameForm
  SUP <sr17>
)
```

2.3 cimChassisTypeInstance

The class cimChassis defines two linked indexed arrays: ChassisTypes and TypeDescriptions. In the LDAP mapping, these are replaced with separate instances of cimChassisTypeInstance, DIT contained by cimChassis.

```
( <oid-at111> NAME 'cimChassisTypes'
  DESC 'An enumerated, integer value indicating the type of
        Chassis.'
  SYNTAX integer SINGLE-VALUE
)
```

```
      ( <oid-at112> NAME 'cimTypeDescriptions'
        DESC 'A free-form strings providing more information on the
              ChassisTypes array entries.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-oc93> NAME 'cimChassisTypeInstance'
        DESC 'helper class to tie ChassisType and
              TypeDescriptions in Chassis together'
        SUP top
        MUST (arrayIndex)
        MAY (cimChassisType $ cimTypeDescription)
      )

      ( <oid-nf22> NAME 'cimChassisTypeInstanceNameForm'
        OC cimChassisTypeInstance
        MUST (arrayIndex)
      )

      ( <sr22> NAME 'cimChassisTypeInstanceStructureRule'
        FORM cimChassisTypeInstanceNameForm
        SUP <sr18>
      )
```

2.4 cimMediaTypesSupportedInstance

The class cimStorageMediaLocation defines two linked indexed arrays: MediaTypesSupported and MediaSizesSupported. In the LDAP mapping, these are replaced with separate instances of cimMediaTypeSupportedInstance, DIT contained by cimStorageMediaLocation.

```
      ( <oid-at125> NAME 'cimMediaTypesSupported'
        DESC 'Certain StorageMediaLocations may only be able to accept a
              limited set of PhysicalMedia MediaTypes. This property defines
              the types of Media that are acceptable for placement in the
              Location.
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at126> NAME 'cimMediaSizesSupported'
        DESC 'The size (in inches) of the particular MediaTypes that may
              be placed in the Location.'
        SUP cim22Float32 SINGLE-VALUE
      )

      ( <oid-oc94> NAME 'cimMediaTypesSupportedInstance'
        DESC 'helper class to tie MediaTypesSupported and
```

```
            MediaSizesSupported in StorageMediaLocation together'
      SUP top
      MUST (arrayIndex)
      MAY (cimMediaTypesSupported $ cimMediaSizesSupported)
    )

    ( <oid-nf23> NAME 'cimMediaTypesSupportedInstanceNameForm'
      OC cimMediaTypesSupportedInstance
      MUST (arrayIndex)
    )

    ( <sr23> NAME 'cimMediaTypesSupportedInstanceStructureRule'
      FORM cimMediaTypesSupportedInstanceNameForm
      SUP <sr19>
    )
```

2.5 cimPhysicalLabelsInstance

The class cimPhysicalMedia defines three linked indexed arrays: PhysicalLabels, LabelStates, and LabelFormats. In the LDAP mapping, these are replaced with separate instances of cimPhysicalLabelsInstance, DIT contained by cimPhysicalMedia.

```
    ( <oid-at143> NAME 'cimPhysicalLabels'
      DESC '"labels" on the PhysicalMedia. The format of the labels and
            their state (readable, unreadable, upside-down) are
            indicated in LabelFormats and LabelStates properties.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-at144> NAME 'cimLabelStates'
      DESC 'An enumerated integer describing the state of a label on a
            PhysicalMedia. The Label is listed in the PhysicalLabels
            property.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at145> NAME 'cimLabelFormats'
      DESC 'An  enumerated integer describing the format of a label on
            a PhysicalMedia. The Labels is listed in the PhysicalLabels
            property.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-oc95> NAME 'cimPhysicalLabelsInstance'
      DESC 'helper class to tie PhysicalLabels, LabelStates, and
            LabelFormats in PhsyicalMedia together'
```

```
        SUP top
        MUST (arrayIndex)
        MAY (cimPhysicalLabels $ cimLabelStates $ cimLabelFormats)
      )

      ( <oid-nf24> NAME 'cimPhysicalLabelsInstanceNameForm'
        OC cimPhysicalLabelsInstance
        MUST (arrayIndex)
      )

      ( <sr24> NAME 'cimPhysicalLabelsInstanceStructureRule'
        FORM cimPhysicalLabelsInstanceNameForm
        SUP <sr20>
      )
```

2.6 Floating point attributes: cimFloat32 and cimFloat64

Several attributes in this schema inherit from the attributes cimFloat32 and cimFloat64, defined in [6]. Interested readers are directed there for information about these attribute definition.

3. Class Definitions

For efficiency in the LDAP representation, associations are specified as a combination of auxiliary classes and DIT structure rules. Attribute definitions for each class are presented with the object class. Other definitions are also provided when necessary.

This approach minimizes the number of DN pointers stored in the schema, but some pointer dereferencing is necessary. While not explicitly stated in the definitions below, we assume that all attributes with DN support the matching rule defined in [3]. Attribute names for DN pointers also follow the convention that a single pointer's name ends in "Ref", while an array of pointers' name ends in "Refs".

Note: all attribute, object class, and name form OIDs are place holders, and syntax OIDs in definitions have been replaced by names for clarity.

There are some classes that aren't included in this mapping: Since MediaTransferDevice isn't included in the device model, DeviceServiceLocation isn't included here. MediaPhysicalStatInfo isn't included here because it is filled with nothing but counters. Without StorageExtents, the associations RealizesExtent, RealizesPExtent, RealizesDiskPartition, RealizesAggregatePExtent, and RealizesTapePartition do not make sense to include here. Finally, the PackageTempSensor association is not included because there are

no TemperatureSensors in the schema.

3.1 cim22Location

Locations are the position and address of a PhysicalElement.

```
( <oid-at76> NAME 'cimPhysicalPosition'
  DESC 'Position is a free-form string indicating the placement of
        PhysicalElement. It can specify slot information on a
        HostingBoard, mounting site in a Cabinet, or latitude and
        longitude information, for example, from a GPS. It is part
        of the key of the Location object.'
  SYNTAX string{256} SINGLE-VALUE
)

( <oid-at77> NAME 'cimAddress'
  DESC 'Address is a free-form string indicating a street, building
        or other type of address for the PhysicalElement's
        Location.'
  SYNTAX string{1024} SINGLE-VALUE
)

( <oid-oc46> NAME 'cim22Location'
  DESC 'The Location class specifies the position and address of a
        PhysicalElement.'
  SUP top
  MUST (orderedCimModelPath $ cimName $ cimPhysicalPosition)
  MAY (cimAddress)
)

( <oid-nf12> NAME 'cim22LocationNameForm'
  OC cim22Location
  MUST (orderedCimModelPath)
)

( <sr12> NAME 'cim22LocationStructureRule'
  FORM cim22LocationNameForm
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22Location.

```
( <oid-oc46> NAME 'cim22LocationContentRule'
  DESC 'The auxiliary classes that may be attached to cim22Location'
  AUX (cim22PhysicalElementLocationAuxClass)
)
```

3.2 cim22PhysicalElementLocationAuxClass

This class associates a physical element with a cim22Location object.

```
( <oid-at78> NAME 'cimPhysicalLocationRef'
  DESC 'The PhysicalElement's Location.'
  SYNTAX DN
)

( <oid-oc47> NAME 'cim22PhysicalElementLocationAuxClass'
  DESC 'PhysicalElementLocation associates a PhysicalElement with a
        Location object for inventory or replacement
        purposes. Attribute cimElementRef points to
       cim22PhysicalElement and attribute cimPhysicalLocationRef
points
       to cim22Location. '
  SUP top AUXILIARY
  MAY (cimElementRef $ cimPhysicalLocationRef)
)
```

3.3 cim22PhysicalCapacity

This class describes a physical element's requirements.

```
( <oid-oc48> NAME 'cim22PhysicalCapacity'
  DESC 'PhysicalCapacity is an abstract class describing a
         PhysicalElement's minimum/maximum requirements and
         ability to support different types of hardware. For
         example, minimum and maximum memory requirements can be
         modeled as a subclass of cim22PhysicalCapacity.'
  SUP top ABSTRACT
  MAY (cimName $ cimCaption $ cimDescription)
)
```

3.4 cim22ElementCapacityAuxClass

This class associates a cim22PhysicalCapacity object with one or more cim22PhysicalElements.

```
( <oid-at79> NAME 'cimCapacityRef'
  DESC 'PhysicalCapacity describes the minimum and maximum
        requirements, and ability to support different types of
        hardware for a PhysicalElement.'
  SYNTAX DN
)

( <oid-oc49> NAME 'cim22ElementCapacityAuxClass'
  DESC 'ElementCapacity associates a PhysicalCapacity object with
        one or more PhysicalElements. It serves to associate a
```

```
            description of min/max hardware requirements or
            capabilities (stored as a kind of PhysicalCapacity), with
            the PhysicalElements being described. Attribute
            cimCapacityRef points to cim22PhysicalCapacity. Attribute
            cimElementRef points to cim22PhysicalElement.'
      SUP top AUXILIARY
      MAY (cimCapacityRef $ cimElementRef)
    )
```

3.5 cim22MemoryCapacity

Physical elements are limited in what memory can be installed. Instances of this class store information on what memory is currently installed.

```
    ( <oid-at80> NAME 'cimMemoryType'
      DESC 'The type of memory. This is a part of the object
            key. Values correspond to the list of possible memory types
            in the PhysicalMemory class.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at81> NAME 'cimMinimumMemoryCapacity'
      DESC 'Minimum amount of memory, in Kbytes, that is needed for
            the associated PhysicalElement to operate correctly.
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at82> NAME 'cimMaximumMemoryCapacity'
      DESC 'Maximum amount of memory, in Kbytes, that can be supported
            by the associated PhysicalElement.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-oc50> NAME 'cim22MemoryCapacity'
      DESC 'MemoryCapacity describes the type of Memory that can be
            installed on a PhysicalElement and its minimum/maximum
            configurations. Information on what memory is currently
            "installed", versus an Element's min/max requirements, is
            located in instances of the PhysicalMemory class.'
      SUP cim22PhysicalCapacity
      MUST (orderedCimModelPath $ cimMemoryType)
      MAY (cimMinimumMemoryCapacity $ cimMaximumMemoryCapacity)
    )

    ( <oid-nf13> NAME 'cim22MemoryCapacityNameForm'
      OC cim22MemoryCapacity
      MUST (orderedCimModelPath)
```

```
      )

      ( <sr13> NAME 'cim22MemoryCapacityStructureRule'
        FORM cim22MemoryCapacityNameForm
      )
```

3.6 cim22ConfigurationCapacity

Capacity includes the number of power supplies, fans, disk drives, etc. that can be connected to or placed on/into a physical element (and the number that must be connected/added/removed at a time). cim22ElementCapacityAuxClass identifies the physical element whose configuration is described.

This class does NOT represent the tradeoffs required of one resource for another. It simply represents capacities. For a StorageLibrary, there are only 2 valid configurations - 9 TapeDrives with 88 Slots, or 3 TapeDrives with 264 Slots. it only conveys that 'up to' 9 Drives and 'up to' 264 slots are available and supported.

```
   ( <oid-at83> NAME 'cimObjectType'
     DESC 'The type of object (power supply, fan, disk drive, ...)
           whose capacities are indicated. This information is part of
           the class' key.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-at84> NAME 'cimOtherTypeDescription'
     DESC 'A string describing the object type - used when the
           ObjectType property is set to 0
           ("Other"). OtherTypeDescription should be set to NULL when
           ObjectType is any value other than 0.'
     SYNTAX string{64} SINGLE-VALUE
   )

   ( <oid-at85> NAME 'cimMinimumCapacity'
     DESC 'Minimum number of Elements of type, ObjectType, that must
           be installed.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-at86> NAME 'cimMaximumCapacity'
     DESC 'Maximum number of Elements of type, ObjectType, that may be
           installed.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-at87> NAME 'cimIncrement'
```

```
        DESC 'Increment in which Elements must be added or removed.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-oc51> NAME 'cim22ConfigurationCapacity'
        DESC 'ConfigurationCapacity provides information on the minimum
              and maximum numbers of power supplies, fans, disk drives,
              etc. that can be connected to or placed on/into a
              PhysicalElement (and the number that must be
              connected/added/removed at a time). The PhysicalElement
              whose configuration is described is identified using the
              ElementCapacity association, inherited from
              PhysicalCapacity. The object whose capacities are indicated
              (ie, the power supply or fan) is identified in the
              ObjectType property of this class. Since the same min/max
              configurations can apply to multiple instances, this class
              is not defined as "weak". Examples of the use of the
              ConfigurationCapacity class are to describe that a "control
              unit" Chassis may be connected to (at most) 4 other I/O
              chassis, or to describe what a StorageLibrary's cabinet may
              contain. Continuing the latter example, a particular
              StorageLibrary's cabinet might hold a minimum of 3 and a
              maximum of 9 TapeDrives, and a minimum of 88 and a maximum
              of 264 StorageMediaLocations ("Slots"). This information
              would be described in two instances of
              ConfigurationCapacity, both associated to the
              StorageLibrary's PhysicalPackage.  This class does NOT
              represent the tradeoffs that are likely to be required of
              one resource for another. It simply represents
              capacities. In the case of the StorageLibrary, there may be
              only 2 valid configurations - 9 TapeDrives with 88 Slots,
              or 3 TapeDrives with 264 Slots. This class only conveys
              that "up to" 9 Drives and "up to" 264 slots may be
              available and are supported.'
        SUP cim22PhysicalCapacity
        MUST (orderedCimModelPath $ cimObjectType)
        MAY (cimOtherTypeDescription $ cimMinimumCapacity $
             cimMaximumCapacity $ cimIncrement)
      )

      ( <oid-nf14> NAME 'cim22ConfigurationCapacityNameForm'
        OC cim22ConfigurationCapacity
        MUST (orderedCimModelPath)
      )

      ( <sr14> NAME 'cim22ConfigurationCapacityStructureRule'
        FORM cim22ConfigurationCapacityNameForm
      )
```

3.7 cim22ReplacementSet

A replacement set is a group of physical elements that must be replaced or FRUed together. For example, when replacing a memory card, the component memory chips could be removed and replaced as well.

```
( <oid-oc52> NAME 'cim22ReplacementSet'
  DESC 'The ReplacementSet class aggregates PhysicalElements that
        must be "replaced" or "FRUed" together. For example, when
        replacing a memory card, the component memory chips could
        be removed and replaced as well. Or, a set of memory chips
        may be specified to be replaced or upgraded together using
        this association.'
  SUP top
  MUST (cimName $ orderedCimModelPath)
  MAY (cimDescription)
)

( <oid-nf15> NAME 'cim22ReplacementSetNameForm'
  OC cim22ReplacementSet
  MUST (orderedCimModelPath)
)

( <sr15> NAME 'cim22ReplacementSetStructureRule'
  FORM cim22ReplacementSetNameForm
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22ReplacementSet.

```
( <oid-oc52> NAME 'cim22ReplacementSetContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22ReplacementSet'
  AUX (cim22ParticipatesInSetAuxClass)
)
```

3.8 cim22ParticipatesInSetAuxClass

This class shows which physical elements should be replaced together.

```
( <oid-oc53> NAME 'cim22ParticipatesInSetAuxClass'
  DESC 'ParticipatesInSet indicates which PhysicalElements should
        be replaced together. Attribute cimSetRef points to
        cim22ReplacementSet and attribute cimElementRef points to
        cim22PhysicalElement.'
  SUP top AUXILIARY
  MAY (cimSetRef $ cimElementRef)
```

```
    )
```

3.9 cim22PhysicalPackage

A physical package contains or hosts components. Examples are a rack enclosure or an adapter Card.

```
    ( <oid-at89> NAME 'cimRemovable'
      DESC 'A PhysicalPackage is Removable if it is designed to be
            taken in and out of the physical container in which it is
            normally found, without impairing the function of the
            overall packaging. A Package can still be Removable if
            power must be "off" in order to perform the removal. If
            power can be "on" and the Package removed, then the Element
            is both Removable and HotSwappable. For example, an extra
            battery in a laptop is Removable, as is a disk drive
            Package inserted using SCA connectors. However, the latter
            is also HotSwappable. A laptop's display is not Removable,
            nor is a non-redundant power supply. Removing these
            components would impact the function of the overall
            packaging or is impossible due to the tight integration of
            the Package.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at90> NAME 'cimReplaceable'
      DESC 'A PhysicalPackage is Replaceable if it is possible to
            replace (FRU or upgrade) the Element with a physically
            different one. For example, some ComputerSystems allow the
            main Processor chip to be upgraded to one of a higher clock
            rating. In this case, the Processor is said to be
            Replaceable. Another example is a power supply Package
            mounted on sliding rails. All Removable packages are
            inherently Replaceable.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at91> NAME 'cimHotSwappable'
      DESC 'A PhysicalPackage is HotSwappable if it is possible to
            replace the Element with a physically different but
            equivalent one while the containing Package has power
            applied to it (ie, is "on"). For example, a disk drive
            Package inserted using SCA connectors is both Removable and
            HotSwappable. All HotSwappable packages are inherently
            Removable and Replaceable.'
      SYNTAX boolean SINGLE-VALUE
    )
```

```
      ( <oid-at92> NAME 'cimHeight'
        DESC 'The height of the PhysicalPackage in inches.'
        SUP CimFloat32 SINGLE-VALUE
      )

      ( <oid-at93> NAME 'cimDepth'
        DESC 'The depth of the PhysicalPackage in inches.'
        SUP CimFloat32 SINGLE-VALUE
      )

      ( <oid-at94> NAME 'cimWidth'
        DESC 'The width of the PhysicalPackage in inches.'
        SUp CimFloat32 SINGLE-VALUE
      )

      ( <oid-at95> NAME 'cimWeight'
        DESC 'The weight of the PhysicalPackage in pounds.'
        SUP CimFloat32 SINGLE-VALUE
      )

      ( <oid-oc54> NAME 'cim22PhysicalPackage'
        DESC 'The PhysicalPackage class represents PhysicalElements that
              contain or host other components. Examples are a Rack
              enclosure or an adapter Card.'
        SUP cim22PhysicalElement
        MAY (cimRemovable $ cimReplaceable $ cimHotSwappable $
             cimHeight $ cimDepth $ cimWidth $ cimWeight)
      )

      ( <oid-nf16> NAME 'cim22PhysicalPackageNameForm'
        OC cim22PhysicalPackage
        MUST (orderedCimModelPath)
      )

      ( <sr16> NAME 'cim22PhysicalPackageStructureRule'
        FORM cim22PhysicalPackageNameForm
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22PhysicalPackage.

```
      ( <oid-oc54> NAME 'cim22PhysicalPackageContentRule'
        DESC 'The auxiliary classes that may be attached to
              cim22PhysicalPackage'
        AUX (cim22ContainerAuxClass $ cim22PackageInChassisAuxClass $
             cim22PackagedComponentAuxClass $
             cim22PackageInConnectorAuxClass $
             cim22PackageInSlotAuxClass $ cim22ConnectorOnPackageAuxClass $
```

```
            cim22ComputerSystemPackageAuxClass $
            cim22LibraryPackageAuxClass $ cim22PackageCoolingAuxClass $
            cim22PackageTempSensorAuxClass $ cim22PackageAlarmAuxClass $
            cim22ProductPhysicalElementsAuxClass $
            cim22FRUPhysicalElementsAuxClass $
            cim22PhysicalElementLocationAuxClass $
            cim22ElementCapacityAuxClass $
            cim22ParticipatesInSetAuxClass $ cim22ElementsLinkedAuxClass $
            cim22CollectedMSEsAuxClass $
            cim22ElementConfigurationAuxClass $
            cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
            cim22ProvidesServiceToElementAuxClass $
            cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
    )
```

3.10 cim22ContainerAuxClass

This class represents the relationship between a contained and a containing PhysicalElement.

```
    ( <oid-at96> NAME 'cimLocationWithinContainer'
      DESC 'A free-form string representing the positioning of the
            PhysicalElement within the PhysicalPackage. This string
            could supplement or be used in place of instantiating the
            cimLocation object.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-oc55> NAME 'cim22ContainerAuxClass'
      DESC 'The Container association represents the relationship
            between a contained and a containing PhysicalElement. A
            containing object must be a PhysicalPackage. Attribute
            cimGroupComponentRef points to cim22PhysicalPackage and
            attribute cimPartComponentRef points to
            cim22PhysicalElement.'
      SUP cim22ComponentAuxClass AUXILIARY
      MAY (cimGroupComponentRef $ cimPartComponentRef $
           cimLocationWithinContainer)
    )
```

3.11 cim22PhysicalFrame

A physical frame is a generic frame enclosure.

```
    ( <oid-at97> NAME 'cimCableManagementStrategy'
      DESC 'CableManagementStrategy is a free-form string that contains
            information on how the various cables are connected and
            bundled for the Frame. With many networking,
```

```
            storage-related and power cables, cable management can be a
            complex and challenging endeavor. This string property
            contains information to aid in assembly and service of the
            Frame.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-at100> NAME 'cimLockPresent'
      DESC 'Boolean indicating whether the Frame is protected with a
            lock.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at101> NAME 'cimAudibleAlarm'
      DESC 'Boolean indicating whether the Frame is equipped with an
            audible alarm.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at102> NAME 'cimVisibleAlarm'
      DESC 'Boolean indicating that the equipment includes a visible
            alarm.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at103> NAME 'cimSecurityBreach'
      DESC 'SecurityBreach is an enumerated, integer-valued property
            indicating whether a physical breach of the Frame was
            attempted but unsuccessful (value=4) or attempted and
            successful (5). Also, the values, "Unknown", "Other" or
            "No Breach", can be specified.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at104> NAME 'cimBreachDescription'
      DESC 'BreachDescription is a free-form string providing more
            information if the SecurityBreach property indicates that a
            breach or some other security-related event occurred.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-at105> NAME 'cimIsLocked'
      DESC 'Boolean indicating that the Frame is currently locked.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-oc56> NAME 'cim22PhysicalFrame'
      SUP cim22PhysicalPackage
```

```
        MAY (cimCableManagementStrategy $ cimLockPresent $
             cimAudibleAlarm $ cimVisibleAlarm $ cimSecurityBreach $
             cimBreachDescription $ cimIsLocked)
      )

      ( <oid-nf17> NAME 'cim22PhysicalFrameNameForm'
        OC cim22PhysicalFrame
        MUST (orderedCimModelPath)
      )

      ( <sr17> NAME 'cim22PhysicalFrameStructureRule'
        FORM cim22PhysicalFrameNameForm
      )
```

3.12 cim22Rack

Racks are enclosures in which chassis are placed. Typically they are nothing more than the enclosure, and the chassis packages all functioning componentry.

```
      ( <oid-at106> NAME 'cimTypeOfRack'
        DESC 'Enumeration indicating the type of Rack.  Information such
              as "Telco" rack (value=2) or standard 19 inch rack (1) can
              be specified. The country for which the Rack is
              manufactured is defined in the the CountryDesignation
              property.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at107> NAME 'cimCountryDesignation'
        DESC 'Designation of the country for which the Rack is
              designed. Country code strings are as defined by ISO/IEC
              3166. The rack type is specified in the TypeOfRack
              property.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-oc57> NAME 'cim22Rack'
        DESC 'A Rack is a PhysicalFrame that represents an enclosure in
              which Chassis are placed. Typically a Rack is nothing more
              than the enclosure, and all the functioning componentry is
              packaged in the Chassis, loaded in the Rack.'
        SUP cim22PhysicalFrame
        MAY (cimTypeOfRack $ cimCountryDesignation)
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Rack.

```
      ( <oid-oc57> NAME 'cim22RackContentRule'
        DESC 'The auxiliary classes that may be attached to cim22Rack'
        AUX (cim22ChassisInRackAuxClass)
      )
```

3.13 cim22Chassis

Chassis enclose other elements and provide definable functionality.

```
      ( <oid-at108> NAME 'cimNumberOfPowerCords'
        DESC 'Integer indicating the number of power cords which must be
              connected to the Chassis, for all the componentry to
              operate.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at109> NAME 'cimCurrentRequiredOrProduced'
        DESC 'Current required by the Chassis at 120V. If power is
              provided by the Chassis (as in the case of a UPS), this
              property may indicate the amperage produced, as a negative
              number.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at110> NAME 'cimHeatGeneration'
        DESC 'Amount of heat generated by the Chassis in BTU/hour.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-oc58> NAME 'cim22Chassis'
        DESC 'The Chassis class represents the PhysicalElements that
              enclose other Elements and provide definable functionality,
              such as a desktop, processing node, UPS, disk or tape
              storage, or a combination of these.'
        SUP cim22PhysicalFrame
        MAY (cimNumberOfPowerCords $ cimCurrentRequiredOrProduced $
             cimHeatGeneration)
      )

      ( <oid-nf18> NAME 'cim22ChassisNameForm'
        OC cim22Chassis
        MUST (orderedCimModelPath)
      )

      ( <sr18> NAME 'cim22ChassisStructureRule'
        FORM cim22ChassisNameForm
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Chassis.

```
( <oid-oc58> NAME 'cim22ChassisContentRule'
  DESC 'The auxiliary classes that may be attached to cim22Chassis'
  AUX (cim22ChassisInRackAuxClass $ cim22PackageInChassisAuxClass $
       cim22DockedAuxClass)
)
```

3.14 cim22ChassisInRackAuxClass

This class makes explicit the 'containing' relationship between the Rack and the Chassis.

```
( <oid-at113> NAME 'cimBottomU'
  DESC 'An integer indicating the lowest or "bottom" U in which the
        Chassis is mounted. A "U" is a standard unit of measure for
        the height of a Rack or rack-mountable component. It is
        equal to 1.75 inches or 4.445 cm.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-oc59> NAME 'cim22ChassisInRackAuxClass'
  DESC 'Racks, as simple enclosures, contain Chassis that provide
        the physical componentry realizing processing nodes,
        storage devices, UPSs, etc. The ChassisInRack association
        makes explicit the "containing" relationship between the
        Rack and the Chassis.Attribute cimGroupComponentRef points
        to cim22Rack and attribute cimPartComponentRef points to
        cim22Chassis.'
  SUP cim22ContainerAuxClass AUXILIARY
  MAY (cimGroupComponentRef $ cimPartComponentRef $
       cimLocationWithinContainer $ cimBottomU)
)
```

3.15 cim22PackageInChassisAuxClass

This class makes the containment relationship between a chassis and other packages explicit.

```
( <oid-oc60> NAME 'cim22PackageInChassisAuxClass'
  DESC 'A Chassis can contain other Packages, such as other Chassis
        and Cards. The PackageInChassis association makes explicit
        this relationship. Attribute cimGroupComponentRef points to
        cim22Chassis and attribute cimPartComponentRef points to
        cim22PhysicalPackage.'
  SUP cim22ContainerAuxClass AUXILIARY
  MAY (cimGroupComponentRef $ cimPartComponentRef $
```

```
            cimLocationWithinContainer)
      )
```

3.16 cim22DockedAuxClass

This class makes explicit the relationship between a laptop, a type of chassis, which docks in another type of chassis, a docking station.

```
   ( <oid-oc61> NAME 'cim22DockedAuxClass'
     DESC 'A laptop, a type of Chassis, may be docked in another type
           of Chassis, a Docking Station. This is the relationship
           represented by the Docked association. Because this is such
           a typical relationship, it is explicitly described. Both
           attributes point to cim22Chassis objects.'
     SUP cim22DependencyAuxClass AUXILIARY
     MAY (cimAntecedentRef $ cimDependentRef)
   )
```

3.17 cim22Card

This class represents a type of physical container that can be plugged into another Card or HostingBoard, or is itself a HostingBoard/Motherboard in a Chassis. It includes any package capable of carrying signals and providing a mounting point for PhysicalComponents, such as Chips, or other PhysicalPackages, such as other Cards.

```
   ( <oid-at114> NAME 'cimHostingBoard'
     DESC 'Boolean indicating that this Card is a Motherboard or, more
           generically, a baseboard in a Chassis.'
     SYNTAX boolean SINGLE-VALUE
   )

   ( <oid-at115> NAME 'cimSlotLayout'
     DESC 'SlotLayout is a free-form string that describes the slot
           positioning, typical usage, restrictions, individual slot
           spacings or any other pertinent information for the slots
           on a Card.'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-at116> NAME 'cimRequiresDaughterBoard'
     DESC 'Boolean indicating that at least one daughterboard or
           auxiliary Card is required in order to function properly.'
     SYNTAX boolean SINGLE-VALUE
   )
```

```
      ( <oid-at117> NAME 'cimSpecialRequirements'
        DESC 'Boolean indicating that this Card is physically unique
              from other Cards of the same type and therefore requires a
              special Slot. For example, a double-wide Card requires two
              Slots. Another example is where a certain Card may be used
              for the same general function as other Cards but requires a
              special Slot (e.g., extra long), whereas the other Cards
              can be placed in any available Slot. If set to TRUE, then
              the corresponding property, RequirementsDescription, should
              specify the nature of the uniqueness or purpose of the
              Card.'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-at118> NAME 'cimRequirementsDescription'
        DESC 'A free-form string describing the way(s) in which this Card
              is physically unique from other Cards. This property only
              has meaning when the corresponding boolean property,
              SpecialRequirements, is set to TRUE.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-at119> NAME 'cimOperatingVoltages'
        DESC 'Operating voltages required by the Card.'
        SYNTAX
      )

      ( <oid-oc62> NAME 'cim22Card'
        DESC 'The Card class represents a type of physical container that
              can be plugged into another Card or HostingBoard, or is
              itself a HostingBoard/Motherboard in a Chassis. The
              cim22Card class includes any package capable of carrying
              signals and providing a mounting point for
              PhysicalComponents, such as Chips, or other
              PhysicalPackages, such as other Cards.'
        SUP cim22PhysicalPackage
        MAY (cimHostingBoard $ cimSlotLayout $ cimRequiresDaughterBoard $
             cimSpecialRequirements $ cimRequirementsDescription $
             cimOperatingVoltages)
      )

   The following content rule specifies the auxiliary classes that may
   be attached to cim22Card.

      ( <oid-oc62> NAME 'cim22CardContentRule'
        DESC 'The auxiliary classes that may be attached to cim22Card'
        AUX (cim22CardOnCardAuxClass $ cim22MemoryOnCardAuxClass $
             cim22CardInSlotAuxClass)
```

```
      )
```

3.18 cim22SystemBusCard

System bus cards require additional attributes, detailing the card's bus type and data width, which dictate the type of slot into which the card can be inserted. For example, attributes can define that a card is a PCI, 64 bit adapter.

```
   ( <oid-at120> NAME 'cimBusType'
     DESC 'An enumerated integer describing the System bus type for
           this Card. It indicates the type of Slot into which the
           Card can plug.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-at121> NAME 'cimBusWidth'
     DESC 'System bus width (in bits) required by this Card. If
           "unknown", enter 0. If "other" than the values, 8, 16, 32,
           64 or 128, enter 1.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-oc63> NAME 'cim22SystemBusCard'
     DESC 'The SystemBusCard class represents additional information
           for a cimCard, detailing the Card's bus type and data
           width. These properties dictate the type of Slot into which
           the Card can be inserted. For example, using the properties
           of this class, one can define that a Card is a PCI, 64 bit
           adapter.'
     SUP cim22Card
     MAY (cimBusType $ cimBusWidth)
   )
```

3.19 cim22CardOnCardAuxClass

Cards may be plugged into Motherboards/baseboards, are daughtercards of an adapter, or support special Card-like modules. This auxiliary class describes these relationships.

```
      ( <oid-at122> NAME 'cimMountOrSlotDescription'
        DESC 'A string describing and identifying how the Card is mounted
              on or plugged into the "other" Card. Slot information could
              be included in this field and may be sufficient for certain
              management purposes. If so, this avoids creating
              instantiations of Connector/Slot objects just to model the
              relationship of Cards to HostingBoards or other
              adapters. On the other hand, if Slot and Connector
```

```
            information is available, this field could be used to
            provide more detailed mounting or slot insertion data.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-oc64> NAME 'cim22CardOnCardAuxClass'
      DESC 'Cards may be plugged into Motherboards/baseboards, are
            daughtercards of an adapter, or support special Card-like
            modules. These relationships are described by the
            CardOnCard association. Both reference attributes point to
            cim22Card objects.'
      SUP cim22ContainerAuxClass AUXILIARY
      MAY (cimGroupComponentRef $ cimPartComponentRef $
           cimLocationWithinContainer $ cimMountOrSlotDescription)
    )
```

3.20 cim22StorageMediaLocation

A storage media location holds media and goes beyond being just a location used by a storage library.

```
    ( <oid-at123> NAME 'cimLocationType'
      DESC 'The type of Location. For example, whether this is an
            individual Media "Slot" (value=2), a MediaAccessDevice
            (value=4) or a "Magazine" (value=3) is indicated in this
            property.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at124> NAME 'cimLocationCoordinates'
      DESC 'LocationCoordinates represent the physical location of the
            the StorageMediaLocation instance. The property is defined
            as a free-form string to allow the location information to
            be described in vendor-unique terminology.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-at127> NAME 'cimMediaCapacity'
      DESC 'A StorageMediaLocation may hold more than one PhysicalMedia
            - for example, a Magazine. This property indicates the
            PhysicalMedia capacity of the Location.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-oc65> NAME 'cim22StorageMediaLocation'
      DESC 'StorageMediaLocation is a PhysicalElement where
            PhysicalMedia may be placed. This class describes an entity
            that holds Media and is not just a "place" (as is conveyed
```

```
            by the cim22Location object). This class is typically used
            in the context of a StorageLibrary. Examples of
            StorageMediaLocations are MediaAccessDevices,
            InterLibraryPorts or "slots" in a Library panel.'
      SUP cim22PhysicalPackage
      MAY (cimLocationType $ cimLocationCoordinates $ cimMediaCapacity)
    )

    ( <oid-nf19> NAME 'cim22StorageMediaLocationNameForm'
      OC cim22StorageMediaLocation
      MUST (orderedCimModelPath)
    )

    ( <sr19> NAME 'cim22StorageMediaLocationStructureRule'
      FORM cim22StorageMediaLocationNameForm
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22StorageMediaLocation.

```
    ( <oid-oc65> NAME 'cim22StorageMediaLocationContentRule'
      DESC 'The auxiliary classes that may be attached to
            cim22StorageMediaLocation'
      AUX (cim22DeviceServicesLocationAuxClass $
           cim22PhysicalMediaInLocationAuxClass)
    )
```

3.21 cim22PhysicalComponent

A physical component either can not or does not need to be decomposed into its constituent parts. For example, an ASIC can not be further decomposed and a tape for data storage does not need to be decomposed. Any element that is not a link, connector, or package is subclassed from this class.

```
    ( <oid-oc66> NAME 'cim22PhysicalComponent'
      DESC 'The PhysicalComponent class represents any low-level or
            basic Component within a Package. A Component object either
            can not or does not need to be decomposed into its
            constituent parts. For example, an ASIC (or Chip) can not
            be further decomposed. A tape for data storage
            (PhysicalMedia) does not need to be decomposed. Any
            PhysicalElement that is not a Link, Connector, or Package
            is a descendent (or member) of this class. For example, the
            UART chipset on an internal modem Card would be a subclass
            (if additional properties or associations are defined) or
            an instance of PhysicalComponent.'
      SUP cim22PhysicalElement
```

```
        MAY (cimRemovable $ cimReplaceable $ cimHotSwappable)
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22PhysicalComponent.

```
   ( <oid-oc66> NAME 'cim22PhysicalComponentContentRule'
     DESC 'The auxiliary classes that may be attached to
           cim22PhysicalComponent'
     AUX (cim22PackagedComponentAuxClass $ cim22RealizesAuxClass $
          cim22ProductPhysicalElementsAuxClass $
          cim22FRUPhysicalElementsAuxClass $
          cim22PhysicalElementLocationAuxClass $
          cim22ElementCapacityAuxClass $
          cim22ParticipatesInSetAuxClass $ cim22ContainerAuxClass $
          cim22ElementsLinkedAuxClass $ cim22CollectedMSEsAuxClass $
          cim22ElementConfigurationAuxClass $
          cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
          cim22ProvidesServiceToElementAuxClass $
          cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
   )
```

3.22 cim22PackagedComponentAuxClass

As a physical package typically contains a component, this class makes this relationship explicit. The word, 'typically', is used because a Component may be removed from, or not yet inserted into, its containing Package (ie, the Removable boolean is TRUE). Therefore, a Component may not always be associated with a container.

```
   ( <oid-oc67> NAME 'cim22PackagedComponentAuxClass'
     DESC 'A Component is typically contained by a PhysicalPackage,
           such as a Chassis or Card. The PackagedComponent
           association makes this relationship explicit. In the first
           sentence, the word, "typically", is used. This is because a
           Component may be removed from, or not yet inserted into,
           its containing Package (ie, the Removable boolean is
           TRUE). Therefore, a Component may not always be associated
           with a container. Attribute cimGroupComponentRef points to
           cim22PhysicalPackage and attribute cimPartComponentRef points
           to cim22PhysicalComponent.'
     SUP cim22ContainerAuxClass AUXILIARY
     MAY (cimGroupComponentRef $ cimPartComponentRef $
          cimLocationWithinContainer)
   )
```

3.23 cim22Chip

A chip is of IC hardware, including ASICs, processors, and memory chips.

```
( <oid-at128> NAME 'cimFormFactor'
  DESC 'The implementation form factor for the Chip.  For example,
        values such as SIMM (7), TSOP (9) or PGA (10) can be
        specified.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-oc68> NAME 'cim22Chip'
  DESC 'The Chip class represents any type of integrated circuit
        hardware, including ASICs, processors, memory chips, etc.'
  SUP cim22PhysicalComponent
  MAY (cimFormFactor)
)
```

3.24 cim22PhysicalMemory

Physical memory are low level memory devices.

```
( <oid-at129> NAME 'cimTotalWidth'
  DESC 'Total width, in bits, of the PhysicalMemory, including
        check or error correction bits. If there are no error
        correction bits, the value in this property should match
        that specified for DataWidth.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-at130> NAME 'cimDataWidth'
  DESC 'Data width of the PhysicalMemory, in bits. A data width of
        0 and a TotalWidth of 8 would indicate that the Memory is
        solely used to provide error correction bits.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-at131> NAME 'cimSpeed'
  DESC 'The speed of the PhysicalMemory, in nanoseconds.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-at132> NAME 'cimCapacity'
  DESC 'The total capacity of this PhysicalMemory, in bytes.'
  SYNTAX integer SINGLE-VALUE
)
```

```
( <oid-at133> NAME 'cimBankLabel'
  DESC 'A string identifying the physically labeled bank where the
        Memory is located - for example, "Bank 0" or "Bank A".'
  SYNTAX string{64} SINGLE-VALUE
)

( <oid-at134> NAME 'cimPositionInRow'
  DESC 'Specifies the position of the PhysicalMemory in a
        "row". For example, if it takes two 8-bit memory devices to
        form a 16-bit row, then a value of "2"means that this
        Memory is the second device. 0 is an invalid value for this
        property.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-at135> NAME 'cimInterleavePosition'
  DESC 'The position of this PhysicalMemory in an interleave. 0
        indicates non-interleaved. 1 indicates the first position,
        2 the second position and so on. For example, in a 2:1
        interleave, a value of "1" would indicate that the Memory
        is in the "even" position.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-oc69> NAME 'cim22PhysicalMemory'
  DESC 'PhysicalMemory is a subclass of cim22Chip, representing low
        level memory devices - SIMMS, DIMMs, raw memory chips,
        etc.'
  SUP cim22Chip
  MAY (cimFormFactor $ cimMemoryType $ cimTotalWidth $
       cimDataWidth $ cimSpeed $ cimCapacity $ cimBankLabel $
       cimPositionInRow $ cimInterleavePosition)
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22PhysicalMemory.

```
( <oid-oc69> NAME 'cim22PhysicalMemoryContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22PhysicalMemory'
  AUX (cim22MemoryOnCardAuxClass $ cim22MemoryWithMediaAuxClass)
)
```

3.25 cim22MemoryOnCardAuxClass

Hosting boards, adapter Cards, etc., can hold physical memory. Therefore, this class represents that relationship.

```
      ( <oid-oc70> NAME 'cim22MemoryOnCardAuxClass'
        DESC 'PhysicalMemory can be located on HostingBoards, adapter
              Cards, etc. This association explicitly defines this
              relationship of Memory to Cards.  Attribute
              cimGroupComponentRef points to cim22Card. Attribute
              cimPartComponentRef points to cim22PhysicalMemory.'
        SUP cim22PackagedComponentAuxClass
        MAY (cimGroupComponentRef $ cimPartComponentRef $
             cimLocationWithinContainer)
      )
```

3.26 cim22PhysicalMedia

Physical media are any type of documentation or storage medium, typically removable media. However, this class can also model 'sealed' media where dmtfPackagedComponentAuxClass associates the media with the physical package.

```
      ( <oid-at136> NAME 'cimMediaType'
        DESC 'Specifies the type of the PhysicalMedia, as an enumerated
              integer. The MediaDescription property is used to provide
              more explicit definition of the Media type, whether it is
              pre-formatted, compatability features, etc.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at137> NAME 'cimMediaDescription'
        DESC 'Additional detail related to the MediaType enumeration. For
              example, if value 3 ("QIC Cartridge") is specified, this
              property could indicate whether the tape is wide or 1/4
              inch, whether it is pre-formatted, whether it is Travan
              compatible, etc.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-at138> NAME 'cimWriteProtectOn'
        DESC 'Boolean specifying whether the Media is currently write
              protected by some kind of physical mechanism, such as a
              protect tab on a floppy diskette.'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-at139> NAME 'cimCleanerMedia'
        DESC 'Boolean indicating that the PhysicalMedia is used for
              cleaning purposes and not data storage.'
        SYNTAX boolean SINGLE-VALUE
      )
```

```
      ( <oid-at140> NAME 'cimMediaSize'
        DESC 'Size of the Media in inches. For example, "3.5" would be
              entered for a 3.5 inch disk, or "12" would be entered for a
              12 inch optical disk. On the other hand, "0.5" would be
              defined for a 1/2 inch tape.'
        SUP cim22Float32 SINGLE-VALUE
      )

      ( <oid-at141> NAME 'cimMaxMounts'
        DESC 'For removable Media, the maximum number of times that the
              Media can be mounted before it should be retired. For
              cleaner Media, this is the maximum number of Drive cleans
              that can be performed. For nonremovable Media, such as hard
              disks, this property is not applicable and should be set to
              0.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at142> NAME 'cimDualSided'
        DESC 'Boolean indicating that the Media has two recording sides
              (TRUE) or only a single side (FALSE). Examples of dual
              sided Media include DVD-ROM and some optical
              disks. Examples of single sided Media are tapes and
              CD-ROM.'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-oc71> NAME 'cim22PhysicalMedia'
        DESC 'The PhysicalMedia class represents any type of
              documentation or storage medium, such as tapes, CDROMs,
              etc. This class is typically used to locate and manage
              Removable Media (versus Media sealed with the
              MediaAccessDevice, as a single Package, as is the case with
              hard disks). However, "sealed" Media can also be modeled
              using this class, where the Media would then be associated
              with the PhysicalPackage using the PackagedComponent
relationship.'
        SUP cim22PhysicalComponent
        MAY (cimCapacity $ cimMediaType $ cimMediaDescription $
             cimWriteProtectOn $ cimCleanerMedia $ cimMediaSize $
             cimMaxMounts $ cimDualSided $ cimPhysicalLabels $
             cimLabelStates $ cimLabelFormats)
      )

      ( <oid-nf20> NAME 'cim22PhysicalMemoryNameForm'
        OC cim22PhysicalMemory
        MUST (orderedCimModelPath)
      )
```

```
    ( <sr20> NAME 'cim22PhysicalMemoryStructureRule'
      FORM cim22PhysicalMemoryNameForm
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22PhysicalMedia.

```
    ( <oid-oc71> NAME 'cim22PhysicalMediaContentRule'
      DESC 'The auxiliary classes that may be attached to
            cim22PhysicalMedia'
      AUX (cim22MemoryWithMediaAuxClass $
           cim22PhysicalMediaInLocationAuxClass)
    )
```

3.27 cim22MemoryWithMediaAuxClass

This class shows that memory is associated with a physical media and its cartridge and provides identification and also stores user-specific data.

```
    ( <oid-oc72> NAME 'cim22MemoryWithMediaAuxClass'
      DESC 'MemoryWithMedia indicates that Memory is associated with a
            PhysicalMedia and its cartridge. The Memory provides media
            identification and also stores user-specific
            data. Attribute cimAntecedentRef points to
            cim22PhysicalMemory and attribute cimDependentRef points to
            cim22PhysicalMedia.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.28 cim22PhysicalMediaInLocationAuxClass

Within a storage library, all media should be accounted for, and be present in some storage location. In addition, one can determine if a location is empty or full based on whether this auxiliary class is attached to a cim22StorageMediaLocation object.

```
    ( <oid-oc73> NAME 'cim22PhysicalMediaInLocationAuxClass'
      DESC 'Within a StorageLibrary, all Media should be accounted for,
            and be present in some Storage Location. This relationship
            is made explicit by the PhysicalMediaInLocation
            association. In addition, one can determine if a Location is
            empty or full based on whether this association exists for
            the StorageMediaLocation. Attribute cimAntecedentRef points
            to cim22StorageMediaLocation and attribute cimDependentRef
            points to cim22PhysicalMedia.'
      SUP cim22DependencyAuxClass AUXILIARY
```

```
        MAY (cimAntecedentRef $ cimDependentRef)
      )
```

3.29 cim22PhysicalTape

This class represents data for a tape Media, including information on the length and whether it must be unloaded from BOT.

```
   ( <oid-at146> NAME 'cimTapeLength'
     DESC 'The physical length of the Tape in feet.'
     SUP cim22Float32 SINGLE-VALUE
   )

   ( <oid-at147> NAME 'cimUnloadAnywhere'
     DESC 'Boolean set to TRUE if the Tape can be unloaded at any
           position on the Media. It is set to FALSE if the tape must
           be at a certain position for unload - such as at the
           beginning of tape (BOT) area, or at mid-tape point for
           TapeDrives with mid-tape load.'
     SYNTAX boolean SINGLE-VALUE
   )

   ( <oid-oc74> NAME 'cim22PhysicalTape'
     DESC 'The PhysicalTape class represents additional data for a
           Tape Media. Information on the tape length and whether it
           must be unloaded from BOT are properties of this class.'
     SUP cim22PhysicalMedia
     MAY (cimTapeLength $ cimUnloadAnywhere)
   )
```

3.30 cim22PhysicalLink

Physical links are the cabling together of physical elements, including cables and links. Rather than model the numerous physical cables within a physical package or network, this class is intended for those cases where the cables or links are either critical components or important assets.

```
   ( <oid-at149> NAME 'cimMaxLength'
     DESC 'The maximum length of the PhysicalLink in feet.'
     SUP cim22Float64 SINGLE-VALUE
   )

   ( <oid-at150> NAME 'cimLength'
     DESC 'The current length of the PhysicalLink in feet. For some
           connections, especially wireless technologies, this
           property may not be applicable and should be left
           uninitialized.'
```

```
        SUP cim22Float64 SINGLE-VALUE
      )

      ( <oid-at151> NAME 'cimWired'
        DESC 'Boolean indicating whether the PhysicalLink is an actual
              cable (TRUE) or a wireless connection (FALSE).'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-oc75> NAME 'cim22PhysicalLink'
        DESC 'The PhysicalLink class represents the cabling of
              PhysicalElements together. For example, serial or Ethernet
              cables, and infrared Links would be subclasses (if
              additional properties or associations are defined) or
              instances of PhysicalLink. In many cases, the numerous
              physical cables within a PhysicalPackage or Network will
              not be modeled. However, where these cables or Links are
              critical components, or are tagged assets of the company,
              these objects can be instantiated using this class or one
              of its descendent classes.'
        SUP cim22PhysicalElement
        MAY (cimMaxLength $ cimLength $ cimWired $ cimMediaType)
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22PhysicalLink.

```
      ( <oid-oc75> NAME 'cim22PhysicalLinkContentRule'
        DESC 'The auxiliary classes that may be attached to
              cim22PhysicalLink'
        AUX (cim22ElementsLinkedAuxClass $
             cim22LinkHasConnectorAuxClass $ cim22RealizesAuxClass $
             cim22ProductPhysicalElementsAuxClass $
             cim22FRUPhysicalElementsAuxClass $
             cim22PhysicalElementLocationAuxClass $
             cim22ElementCapacityAuxClass $
             cim22ParticipatesInSetAuxClass $ cim22ContainerAuxClass $
             cim22CollectedMSEsAuxClass $
             cim22ElementConfigurationAuxClass $
             cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
             cim22ProvidesServiceToElementAuxClass $
             cim22ComponentAuxClass $ cim22SystemComponentAuxClass)

      )
```

3.31 cim22ElementsLinkedAuxClass

This class shows which physical elements are cabled together by a physical link.

```
( <oid-oc76> NAME 'cim22ElementsLinkedAuxClass'
  DESC 'The ElementsLinked association indicates which
        PhysicalElements are cabled together by a
        PhysicalLink. Attribute cimAntecedentRef points to
        cim22PhysicalLink and attribute cimDependentRef points to
        cim22PhysicalElement.'
  SUP cim22DependencyAuxClass AUXILIARY
  MAY (cimAntecedentRef $ cimDependentRef)
)
```

3.32 cim22PhysicalConnector

This class represents any physical element that is used to connect to other elements. Any object that can be used to connect and transmit signals or power between two or more physical elements is a descendant of this class. For example, slots and D-shell connectors are types of physical connectors.

```
( <oid-at152> NAME 'cimConnectorPinout'
  DESC 'A free-form string describing the pin configuration and
        signal usage of a PhysicalConnector.'
  SYNTAX string SINGLE-VALUE
)

( <oid-at153> NAME 'cimConnectorType'
  DESC 'An array of integers defining the type of
        PhysicalConnector. An array is specified to allow the
        description of "combinations" of Connector information. For
        example, one array entry could specify RS-232 (value=25),
        another DB-25 (value=23) and a third entry define the
        Connector as "Male" (value=2).'
  SYNTAX integer
)

( <oid-oc77> NAME 'cim22PhysicalConnector'
  DESC 'The PhysicalConnector class represents any PhysicalElement
        that is used to connect to other Elements. Any object that
        can be used to connect and transmit signals or power
        between two or more PhysicalElements is a descendant (or
        member) of this class. For example, Slots and D-shell
        connectors are types of PhysicalConnectors.'
  SUP cim22PhysicalElement
  MAY (cimConnectorPinout $ cimConnectorType)
```

```
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22PhysicalConnector.

```
    ( <oid-oc77> NAME 'cim22PhysicalConnectorContentRule'
      DESC 'The auxiliary classes that may be attached to
            cim22PhysicalConnector'
      AUX (cim22ConnectedToAuxClass $ cim22PackageInConnectorAuxClass $
           cim22LinkHasConnectorAuxClass $
           cim22ConnectorOnPackageAuxClass $
           cim22AdapterActiveConnectionAuxClass $ cim22RealizesAuxClass $
           cim22ProductPhysicalElementsAuxClass $
           cim22FRUPhysicalElementsAuxClass $
           cim22PhysicalElementLocationAuxClass $
           cim22ElementCapacityAuxClass $
           cim22ParticipatesInSetAuxClass $ cim22ContainerAuxClass $
           cim22ElementsLinkedAuxClass $ cim22CollectedMSEsAuxClass $
           cim22ElementConfigurationAuxClass $
           cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
           cim22ProvidesServiceToElementAuxClass $
           cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
    )
```

3.33 cim22ConnectedToAuxClass

This class shows that two or more physical connectors are connected.

```
    ( <oid-oc78> NAME 'cim22ConnectedToAuxClass'
      DESC 'The ConnectedTo association indicates that two or more
            PhysicalConnectors are connected together. Both attributes
            point to cim22PhysicalConnector objects.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.34 cim22Slot

A slot represents connectors into which packages are inserted. For example, a physical package that is a disk drive may be inserted into a SCA slot. As another example, a card may be inserted into a 16-, 32-, or 64-bit expansion slot on a hosting board.

```
    ( <oid-at154> NAME 'cimSupportsHotPlug'
      DESC 'Boolean indicating whether the Slot supports hot-plug of
            adapter Cards.'
      SYNTAX boolean SINGLE-VALUE
    )
```

```
      ( <oid-at155> NAME 'cimHeightAllowed'
        DESC 'Maximum height of an adapter Card that can be inserted into
              the Slot, in inches./
        SUP cim22Float32 SINGLE-VALUE
      )

      ( <oid-at156> NAME 'cimLengthAllowed'
        DESC 'Maximum length of an adapter Card that can be inserted into
              the Slot, in inches.'
        SUP cim22Float32 SINGLE-VALUE
      )

      ( <oid-at157> NAME 'cimMaxDataWidth'
        DESC 'Maximum bus width of adapter Cards that can be inserted
              into this Slot, in bits. If the value is "unknown", enter
              0. If the value is other than 8, 16, 32, 64 or 128, enter
              1.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at158> NAME 'cimVccMixedVoltageSupport'
        DESC 'An array of enumerated integers indicating the Vcc voltage
              supported by this Slot.'
        SYNTAX integer
      )

      ( <oid-at159> NAME 'cimVppMixedVoltageSupport'
        DESC 'An array of enumerated integers indicating the Vpp voltage
              supported by this Slot.'
        SYNTAX integer
      )

      ( <oid-at160> NAME 'cimThermalRating'
        DESC 'Maximum thermal dissipation of the Slot in milliwatts.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at161> NAME 'cimSpecialPurpose'
        DESC 'Boolean indicating that this Slot is physically unique and
              may hold special types of hardware, e.g. a graphics
              processor slot. If set to TRUE, then the property,
              SpecialPurposeDescription (a string), should specify the
              nature of the uniqueness or purpose of the Slot.'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-at162> NAME 'cimPurposeDescription'
        DESC 'A free-form string describing that this Slot is physically
```

```
            unique and may hold special types of hardware. This
            property only has meaning when the corresponding boolean
            property, SpecialPurpose, is set to TRUE.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-at163> NAME 'cimNumber'
      DESC 'The Number property indicates the physical slot number,
            which can be used as an index into a system slot table,
            whether or not that slot is physically occupied.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-oc79> NAME 'cim22Slot'
      DESC 'The Slot class represents Connectors into which Packages
            are inserted. For example, a PhysicalPackage that is a
            DiskDrive may be inserted into an SCA "Slot". As another
            example, a Card (subclass of PhysicalPackage) may be
            inserted into a 16-, 32-, or 64-bit expansion "Slot" on a
            HostingBoard. PCI or PCMCIA Type III Slots are examples of
            the latter.'
      SUP cim22PhysicalConnector
      MAY (cimSupportsHotPlug $ cimHeightAllowed $ cimLengthAllowed $
           cimMaxDataWidth $ cimThermalRating $
           cimVccMixedVoltageSupport $ cimVppMixedVoltageSupport $
           cimSpecialPurpose $ cimPurposeDescription $ cimNumber)
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Slot.

```
    ( <oid-oc79> NAME 'cim22SlotContentRule'
      DESC 'The auxiliary classes that may be attached to cim22Slot'
      AUX (cim22SlotInSlotAuxClass $ cim22AdjacentSlotsAuxClass $
           cim22PackageInSlotAuxClass $ cim22CardInSlotAuxClass)
    )
```

3.35 cim22SlotInSlotAuxClass

This class represents the ability of an adapter to extend a slot structure, which enables the slot to support cards that would otherwise be incompatible by interfacing to the slot provided by the adapter. This has many practical uses.

```
    ( <oid-oc80> NAME 'cim22SlotInSlotAuxClass'
      DESC 'Slots are special types of Connectors into which adapter
            Cards are typically inserted. The SlotInSlot relationship
            represents the ability of a special adapter to extend the
```

```
            existing Slot structure to enable otherwise incompatible
            Cards to be plugged into a Frame or HostingBoard. The
            adapter effectively creates a new Slot and can be thought
            of (conceptually) as a Slot in a Slot. This enables Cards
            that would otherwise be physically and/or electrically
            incompatible with the existing Slots to be supported, by
            interfacing to the Slot provided by the adapter. This has
            many practical uses. For example, networking boards are
            very expensive. As new hardware becomes available, Chassis
            and even Card configurations change. To protect the
            investment of their customers, networking vendors will
            manufacture special adapters that enable old Cards to fit
            into new Chassis or HostingBoards and/or new Cards to fit
            into old. This is done using a special adapter that fits
            over one or more existing Slots and presents a new Slot
            into which the Card can plug. Both attributes point to
            cim22Slot objects.'
      SUP cim22ConnectedToAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.36 cim22AdjacentSlotsAuxClass

This class describes the layout of slots on a hosting board or adapter card and includes the distance between the slots and whether they are 'shared'.

```
   ( <oid-at164> NAME 'cimSlotARef'
     DESC 'One of the adjacent Slots.'
     SYNTAX DN
   )

   ( <oid-at165> NAME 'cimSlotBRef'
     DESC 'The "other" adjacent Slot.'
     SYNTAX DN
   )

   ( <oid-at166> NAME 'cimDistanceBetweenSlots'
     DESC 'The distance, in inches, between adjacent Slots.'
     SUP cim22Float32 SINGLE-VALUE
   )

   ( <oid-at167> NAME 'cimSharedSlots'
     DESC 'Slots can be located in close proximity on Hosting Boards
           or other Cards, such that if one of these Slots is
           populated by an adapter Card, the other Slot must be left
           empty. This relationship is indicated by the SharedSlots
           boolean set to TRUE.'
```

```
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-oc81> NAME 'cim22AdjacentSlotsAuxClass'
        DESC 'AdjacentSlots describes the layout of Slots on a
              HostingBoard or adapter Card. Information like the distance
              between the Slots and whether they are "shared" (if one is
              populated, then the other Slot can not be used), is
              conveyed as properties of the association. Both reference
              attributes point to cim22Slot objects.'
        SUP top AUXILIARY
        MAY (cimSlotARef $ cimSlotBRef $ cimDistanceBetweenSlots $
             cimSharedSlots)
      )
```

3.37 cim22PackageInConnectorAuxClass

This class represents the relationship between cards that are into system connectors for power and/or to transfer data. For example, it would be used to describe the insertion of a daughter card onto another card.

```
      ( <oid-oc82> NAME 'cim22PackageInConnectorAuxClass'
        DESC 'Adapter cards and other "packaging" are plugged into System
              Connectors for power and/or to transfer data. This
              relationship is defined by PackageInConnector. For example,
              it would be used to describe the insertion of a
              daughtercard onto another Card. Various subclasses of
              PackageInConnector are also defined. PackageInSlot and its
              subclass, CardInSlot, are two examples of
              subclasses. Attribute cimAntecedentRef points to
              cim22PhysicalConnector and attribute cimDependentRef points
              to cim22PhysicalPackage.'
        SUP cim22DependencyAuxClass AUXILIARY
        MAY (cimAntecedentRef $ cimDependentRef)
      )
```

3.38 cim22PackageInSlotAuxClass

Complex networking devices often are based on chassis, which allow for enhancement and/or augmentation of their base functionality adding new chassis devices, similar to adding cards. This auxiliary class models this capability.

```
      ( <oid-oc83> NAME 'cim22PackageInSlotAuxClass'
        DESC 'Complex networking devices often are Chassis-based. These
              Chassis allow for enhancement and/or augmentation of their
              base functionality by accepting additional Chassis devices,
```

```
            similar to accepting functionality in the form of adding
            Cards. This association models this capability.. Attribute
            cimAntecedentRef points to cim22Slot and attribute
            cimDependentRef points to cim22PhysicalPackage.'
      SUP cim22PackageInConnectorAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.39 cim22CardInSlotAuxClass

Slots are special types of connectors into which cards are inserted. This relationship of a Card in a Slot is made explicit using this class.

```
   ( <oid-oc84> NAME 'cim22CardInSlotAuxClass'
     DESC 'Slots are special types of Connectors into which adapter
           Cards are inserted. This relationship of a Card in a Slot
           is made explicit using the CardInSlot
           association. Attribute cimAntecedentRef points to cim22Slot
           and attribute cimDependentRef points to cim22Card.'
     SUP cim22PackageInSlotAuxClass AUXILIARY
     MAY (cimAntecedentRef $ cimDependentRef)
   )
```

3.40 cim22LinkHasConnectorAuxClass

Cables and links use physical connectors to connect physical elements, which this class explicitly defines.

```
   ( <oid-oc85> NAME 'cim22LinkHasConnectorAuxClass'
     DESC 'Cables and Links utilize PhysicalConnectors to actually
           "connect" PhysicalElements. This association explicitly
           defines this relationship of Connectors for
           PhysicalLinks. Attribute cimGroupComponentRef points to
           cim22PhysicalLink and attribute cimPartComponentRef points to
           cim22PhysicalConnector.'
     SUP cim22ComponentAuxClass AUXILIARY
     MAY (cimGroupComponentRef $ cimPartComponentRef)
   )
```

3.41 cim22ConnectorOnPackageAuxClass

Physical packages contain connectors and other physical elements, which this class makes explicit.

```
   ( <oid-oc86> NAME 'cim22ConnectorOnPackageAuxClass'
     DESC 'PhysicalPackages contain Connectors as well as other
           PhysicalElements. The ConnectorOnPackage association makes
```

```
            explicit the containment relationship between Connectors
            and Packages. Attribute cimGroupComponentRef points to
            cim22PhysicalPackage and attribute cimPartComponentRef points
            to cim22PhysicalConnector.'
      SUP cim22ContainerAuxClass AUXILIARY
      MAY (cimGroupComponentRef $ cimPartComponentRef $
           cimLocationWithinContainer)
   )
```

3.42 cim22AdapterActiveConnectionAuxClass

This class shows that a network adapter uses a physical connector for output to the network. This relationship is important when the adapter can choose from one of several connectors.

```
   ( <oid-oc87> NAME 'cim22AdapterActiveConnectionAuxClass'
     DESC 'The AdapterActiveConnection relationship indicates that a
           NetworkAdapter is using the referenced PhysicalConnector to
           output to the network. This relationship is important when
           the Adapter can choose to output from one of several
           Connectors. The Connectors may be associated with the
           NetworkAdapter in a Realizes relationship - but this is not
           required. This association provides additional information
           (i.e., "in use for communication") different than
           Realizes. Attribute cimAntecedentRef points to
           cim22PhysicalConnector and attribute cimDependentRef points
           to cim22NetworkAdapter.'
     SUP cim22DependencyAuxClass AUXILIARY
     MAY (cimAntecedentRef $ cimDependentRef)
   )
```

3.43 cim22ComputerSystemPackageAuxClass

Similar to the way that physical element 'realize' logical devices, physical packages 'realize' unitary computer systems. This class explicitly defines this relationship.

```
   ( <oid-at168> NAME 'cimPlatformGUID'
     DESC 'A Gloabally Unique Identifier for the System's Package.'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-oc88> NAME 'cim22ComputerSystemPackageAuxClass'
     DESC 'Similar to the way that LogicalDevices are "Realized" by
           PhysicalElements, UnitaryComputerSystems are realized in
           one or more PhysicalPackages. The ComputerSystemPackage
           association explicitly defines this relationship. Attribute
           cimAntecedentRef points to cim22PhysicalPackage and attribute
```

```
            cimDependentRef points to cim22UnitaryComputerSystem.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef $ cimPlatformGUID)
    )
```

3.44 cim22LibraryPackageAuxClass

Similar to the way that physical element 'realize' logical devices, physical packages 'realize' storage libraries. This class explicitly defines this relationship.

```
    ( <oid-oc89> NAME 'cim22LibraryPackageAuxClass'
      DESC 'Similar to the way that LogicalDevices are "Realized" by
            PhysicalElements, a StorageLibrary can be realized in one
            or more PhysicalPackages. The LibraryPackage association
            explicitly defines this relationship. Attribute
            cimAntecedentRef points to cim22PhysicalPackage and attribute
            cimDependentRef points to cim22StorageLibrary.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.45 cim22PackageCoolingAuxClass

Often, a package includes a cooling device to assist in the cooling of the Package in general. This relationship is described by this class.

```
    ( <oid-oc90> NAME 'cim22PackageCoolingAuxClass'
      DESC 'Often, a CoolingDevice is installed in a Package such as a
            Chassis or a Rack, not for a specific Device, but to assist
            in the cooling of the Package in general. This relationship
            is described by the PackageCooling association. Attribute
            cimAntecedentRef points to cim22CoolingDevice. Attribute
            cimDependentRef points to cim22PhysicalPackage.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.46 cim22PackageAlarmAuxClass

Often, an alarm device is installed as part of a package, not to track any particular logical device or physical component, but the package itself. This relationship is described by this class.

```
    ( <oid-oc91> NAME 'cim22PackageAlarmAuxClass'
      DESC 'Often, an AlarmDevice is installed as part of a Package,
            not to indicate issues with any particular LogicalDevice or
```

```
            PhysicalComponent, but with the Package's environment in
            general, its security state or its overall health. This
            relationship is described by the PackageAlarm
            association. Attribute cimAntecedentRef points to
            cim22AlarmDevice and attribute cimDependentRef points to
            cim22PhysicalPackage.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

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, Decemeber 1997.

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

[3] Ryan Moats, Gerald Maziarski, John Strassner, "Extensible Match Rule to Dereference Pointers", Internet Draft (work in progress), June 1999.

[4] DMTF, "CIM Physcial Model, v2.2".

[5] Ryan Moats, Gerald Maziarski, John Strassner, "LDAP Schema for the DMTF Core CIM v2.2 Model", Internet Draft (work in progress), December 1999.

[6] Doug Wood, "Directory string representation for floating point values", Internet Draft (work in progress), December 1999.

5. Author's Addresses

Ryan Moats
15621 Drexel Circle
Omaha, NE 68135
USA
E-mail: jayhawk@att.com

Jerry Maziarski
Room C3-3Z01
200 S. Laurel Ave.
Middletown, NJ 07748
USA

E-mail: gfm@qsun.att.com

John Strassner
Cisco Systems, Bldg 1
170 West Tasman Drive
San Jose, CA 95134
E-mail: johns@cisco.com