Ryan Moats
AT&T

December 1999

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

Status of this Memo

Abstract

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

Table of Contents

1. Introduction

This draft presents a LDAPv3 [1,2] schema for the DMTF CIM Core model. Associations are mapped using a combination of auxiliary classes and DIT structure rules. All attribute, object class, and name form OIDs are place holders, and syntax OIDs in definitions have been replaced by names for clarity. Further, structure rule

identifiers are place holders and should be replaced as dictated by local implementations.

This document is a product of the DMTF LDAP WG.

2. LDAP Mapping Considerations

2.1 Differences from the Core CIM Model

This mapping differs from the core CIM model in several ways. The hierarchy of classes beginning with CIM_StatisticalInformation has not been mapped. The CIM_DependencyContaxt association is not mapped. Finally, the classes CIM_Service, CIM_System, CIM_ServiceAccessPoint, and CIM_LogicalDevice have been mapped to structural classes to support basic DIT containment rules.

2.2 cimAssociationInstance

This object class is used when attaching multiple association auxiliary classes to a structural object class would lead to attribute confusion or collision. Instead, the association auxiliary classes attach to cimAssociationInstance classes that are DIT contained by the original structural class.

```
( <oid-at74> NAME 'cimAssociationName'
  DESC 'The associations' name'
  SYNTAX string SINGLE-VALUE
)

( <oid-at75> NAME 'cimAssociationTypeName'
  DESC 'The associations' type name. It can support storing extra
        information about the association type'
  SYNTAX string SINGLE-VALUE
)

( <oid-oc44> NAME 'cimAssociationInstance'
  SUP top
  MUST (orderedCimModelPath)
  MAY (cimAssociationTypeName)
)

( <oid-nf9> NAME 'cimAssociationInstanceNameForm'
  OC cimAssociationInstance
  MUST (orderedCimModelPath)
)

( <sr9> NAME 'cimAssociationInstanceStructureRule'
```

```
        FORM cimAssociationInstanceNameForm
        SUP <sr1> <sr2> <sr3> <sr4> <sr5> <sr6> <sr7> <sr8>
      )
```

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

```
      ( <oid-oc44> NAME 'cimAssociationInstanceContentRule'
        DESC 'The auxiliary classes that may be attached to
              cimAssociationInstance'
        AUX (cim22CollectedMSEsAuxClass $ cim22CollectedCollectionsAuxClass
$
             cim22LogicalIdentityAuxClass $
             cim22ConfigurationComponentAuxClass $
             cim22ElementConfigurationAuxClass $
             cim22CollectionConfigurationAuxClass $
             cim22ElementSettingAuxClass $ cim22DefaultSettingAuxClass $
             cim22SettingContextAuxClass $ cim22CollectionSettingAuxClass $
             cim22DependencyAuxClass $ cim22ServiceAccessBySAPAuxClass $
             cim22ServiceServiceDependencyAuxClass $
             cim22ServiceSAPDependencyAuxClass $
             cim22SAPSAPDependencyAuxClass $
             cim22ProvidesServiceToElementAuxClass $ cim22RealizesAuxClass $
             cim22ComponentAuxClass $ cim22SystemComponentAuxClass $
             cim22ServiceComponentAuxClass $ cim22ProductParentChildAuxClass
$
             cim22CompatibleProductAuxClass $
             cim22ProductProductDependencyAuxClass $
             cim22ProductSupportAuxClass $ cim22ProductFRUAuxClass $
             cim22ProductPhysicalElementsAuxClass $
             cim22FRUPhysicalElementsAuxClass $
             cim22FRUIncludesProductAuxClass)
      )
```

2.3 cimOtherIdentifyingInfoInstance

CIM defines the concept of an ordered array, which LDAP does not support. In the core model, indexed arrays are only used in two cases to tie two properties together. In the LDAP mapping, these attributes are replaced with separate instances of cimOtherIdentifyingInfoInstance that each contain a single pair of property values and are DIT contained by the parent class. The attribute cimOtherIdentifyingInfo is already defined below and reused here. The attribute arrayIndex is defined here as the RDN for this class.

```
      ( <oid-at34> NAME 'arrayIndex'
        DESC 'the index of this child'
```

```
        SYNTAX integer
      )
```

```
( <oid-at35> NAME 'cimIdentifyingDescriptions'
  DESC 'A free-form string providing explanations and
        details behind the entries in the OtherIdentifyingInfo
        array.'
  SYNTAX string
)

( <oid-oc45> NAME 'cimOtherIdentifyingInfoInstance'
  DESC 'helper class to tie indexed arrays in core model together'
  SUP top
  MUST (arrayIndex)
  MAY (cimOtherIdentifyingInfo $ cimIdentifyingDescriptions)
)

( <oid-nf10> NAME 'cimOtherIdentifyingInfoInstanceNameForm'
  OC cimOtherIdentifyingInfoInstance
  MUST (arrayIndex)
)

( <sr10> NAME 'cimOtherIdentifyingInfoInstanceStructureRule'
  FORM cimOtherIdentifyingInfoInstanceNameForm
  SUP <sr3> <sr11>
)
```

2.4 Naming considerations

To support naming in the LDAP mapping of the core schema, the attribute orderedCimModelPath is defined. It is used to provide a simple RDN for directory implementations. The name form rules that follow show that this attribute should be used as the RDN for structural classes.

```
( <oid-at1> NAME 'orderedCimModelPath'
  DESC 'The model path for the instance. Used as an RDN'
  SYNTAX string SINGLE-VALUE
)
```

The value of this attribute is constructed by ordering the CIM keys of the object in the US-ASCII collation order of the property types.

To support an existing naming plan, an implementor may specify additional/replacement nameform rules.

2.5 Inheritance Assumptions

This schema mapping is based on the assumption that name form rules structural rules, and content rules are inherited by both auxiliary and structural sub-classes. For implementations where this is not

the case, extra rules will need to be specified.

On the other hand, the attributes contained in auxiliary classes in this mapping are based on the assumption that inheritance does not occur between auxiliary classes (even though such inheritance is shown in the auxiliary class definitions). For implementations that do support this type of inheritance, the schema definitions will be simpler.

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.

While this approach was chosen to minimize the number of DN pointers stored in the schema, some pointer dereferencing is necessary. While not explicitly stated, all DN pointers are assumed to support the extended matching rule defined in [3].

3.1 cim22ManagedSystemElement

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.

```
( <oid-at2> NAME 'cimCaption'
  DESC 'The Caption property is a short textual description
       (one-line string) of the object.'
  SYNTAX string{64} SINGLE-VALUE
)

( <oid-at3> NAME 'cimDescription'
  DESC 'The Description property provides a textual description of
        the object.'
  SYNTAX string SINGLE-VALUE
)

( <oid-at4> NAME 'cimInstallDate'
  DESC 'A datetime value indicating when the object was
        installed.'
  SYNTAX generalizedTime SINGLE-VALUE
)

( <oid-at5> NAME 'cimName'
```

```
        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 string{256} SINGLE-VALUE
      )

      ( <oid-at6> NAME 'cimStatus'
        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" shows that the
              Element is functioning, but needs attention. Examples of
              "Stressed" states are overload, overheated, etc. The
              condition "Pred Fail" (failure predicted) shows that
              an Element is functioning properly but predicting a
              failure soon. An example is a SMART-enabled hard
              drive. Non-operational statuses can also be specified.
              These are "Error", "NonRecover", "Starting",
              "Stopping" and "Service". "NonRecover" shows that a
              non-recoverable error has occurred. "Service" describes an
              Element being configured, maintained or 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
              another state.'
        SYNTAX string{10} SINGLE-VALUE
      )

      ( <oid-oc1> NAME 'cim22ManagedSystemElement'
        DESC 'cim22ManagedSystemElement 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 top ABSTRACT
        MUST (orderedCimModelPath)
        MAY (cimCaption $ cimDescription $ cimInstallDate $ cimName $
             cimStatus)
      )
```

3.2 cim22CollectionOfMSEs

This object allows the grouping of cim22ManagedSystemElement 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 cim22RedundancyGroup (which is correctly subclassed from cim22LogicalElement).

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 cim22ManagedSystemElements. 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.

```
( <oid-at7> NAME 'cimCollectionID'
  DESC 'The identification of the Collection object. When
        subclassed, the CollectionID property can be overridden to
        be a Key property.'
  SYNTAX string{256} SINGLE-VALUE
)

( <oid-oc2> NAME 'cim22CollectionOfMSEs'
  DESC 'The CollectionOfMSEs object allows the grouping of
        ManagedSystemElements 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. Therefore, it is incorrect to
        subclass groups that have state/status from
        CollectionOfMSEs - an example is cim22RedundancyGroup (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 way, and then
        querying the ElementSetting and ElementConfiguration
        associations to determine if the Collection set is
        completely covered.'
```

```
        SUP top ABSTRACT
        MAY (cimCollectionID $ cimCaption $ cimDescription)
      )
```

3.3 cim22CollectedMSEsAuxClass

This auxiliary class represents a generic association used to establish the members of the grouping object, cim22CollectionOfMSEs.

```
   ( <oid-at8> NAME 'cimCollectionRef'
     DESC 'The grouping or "bag" object that represents the Collection.'
     SYNTAX DN
   )

   ( <oid-at9> NAME 'cimMemberRef'
     DESC 'The members of the Collection.'
     SYNTAX DN
   )

   ( <oid-oc3> NAME 'cim22CollectedMSEsAuxClass'
     DESC 'cimCollectedMSEs is a generic association used to
           establish the members of the grouping object,
           CollectionOfMSEs.  Attribute cimCollectionRef points to
           cim22CollectionOfMSEs and cimMemberRef points to
           cim22ManagedSystemElement.'
     SUP top AUXILIARY
     MAY (cimCollectionRef $ cimMemberRef)
   )
```

3.4 cim22CollectedCollectionsAuxClass

This association represents that a cim22CollectionOfMSEs may itself be contained in another cim22CollectionOfMSEs object.

```
   ( <oid-at10> NAME 'cimCollectionInCollectionRef'
     DESC 'The "collected" Collection.'
     SYNTAX DN
   )

   ( <oid-oc4> NAME 'cim22CollectedCollectionsAuxClass'
     DESC 'cim22CollectedCollections is an aggregation association
           representing that a CollectionOfMSEs may itself be
           contained in a CollectionOfMSEs. Both attributes point to
           cim22CollectionOfMSEs.'
     SUP top AUXILIARY
     MAY (cimCollectionRef $ cimCollectionInCollectionRef)
   )
```

3.5 cim22PhysicalElement

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.

```
   ( <oid-at11> NAME 'cimCreationClassName'
     DESC 'CreationClassName shows 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 string{256} SINGLE-VALUE
   )

   ( <oid-at12> NAME 'cimTag'
     DESC 'An arbitrary string that uniquely identifies the
           PhysicalElement 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
           high in the object hierarchy 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 exists - and may even be inserted
           into a different scoping container. Therefore, the key for
           PhysicalElement is an arbitrary string and is defined
           independently of any placement or location-oriented
hierarchy.'
     SYNTAX string{256} SINGLE-VALUE
   )

   ( <oid-at13> NAME 'cimManufacturer'
     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
           cim22Product.'
     SYNTAX string{256} SINGLE-VALUE
   )
```

```
      ( <oid-at14> NAME 'cimModel'
```

```
        DESC 'The name by which the PhysicalElement is generally known.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at15> NAME 'cimSKU'
        DESC 'The stock keeping unit number for this PhysicalElement.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at16> NAME 'cimSerialNumber'
        DESC 'A manufacturer-allocated number used to identify the
              PhysicalElement.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at17> NAME 'cimVersion'
        DESC 'A string indicating the version of the PhysicalElement.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at18> NAME 'cimPartNumber'
        DESC 'The part number assigned by the organization responsible
              for producing or manufacturing the PhysicalElement.'
        SYNTAX string{256} SINGLE-VALUE
      )

      ( <oid-at19> NAME 'cimOtherIdentifyingInfo'
        DESC 'OtherIdentifyingInfo captures additional data, beyond asset
              tag information, that could be used to identify a
              PhysicalElement. 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 string SINGLE-VALUE
      )

      ( <oid-at20> NAME 'cimPoweredOn'
        DESC 'Boolean indicating that the PhysicalElement is powered on
              (TRUE), or is currently off (FALSE).'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-oc5> NAME 'cim22PhysicalElement'
        DESC 'Subclasses of cim22PhysicalElement 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
            PhysicalElements. 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 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 cim22ManagedSystemElement ABSTRACT
      MUST (cimTag $ cimCreationClassName)
      MAY (cimManufacturer $ cimModel $ cimSKU $ cimSerialNumber $
           cimVersion $ cimPartNumber $ cimOtherIdentifyingInfo $
           cimPoweredOn)
   )
```

3.6 cim22LogicalElement

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.

```
   ( <oid-oc6> NAME 'cim22LogicalElement'
     DESC 'cim22LogicalElement is a base class for all the components
           of a System that represent abstract system components, such
           as Files, Processes, or system capabilities as Logical
           Devices.'
     SUP cim22ManagedSystemElement ABSTRACT
   )
```

3.7 cim22LogicalIdentityAuxClass

This auxiliary class represents an abstract and generic association, showing that two cim22LogicalElements 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 cim22ManagedSystemElement. 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.

```
   ( <oid-at21> NAME 'cimSystemElementRef'
      DESC 'SystemElement represents one aspect of the
```

```
               LogicalElement.'
          SYNTAX DN
      )

      ( <oid-at22> NAME 'cimSameElementRef'
        DESC 'SameElement represents an alternate aspect of the System
              entity.'
        SYNTAX DN
      )

      ( <oid-oc7> NAME 'cim22LogicalIdentityAuxClass'
        DESC 'cim22LogicalIdentityAuxClass 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
              ManagedSystemElement. 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 scenario 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. Both attributes point to
              cim22LogicalElement objects.'
        SUP top AUXILIARY
        MAY (cimSystemElementRef $ cimSameElementRef)
      )
```

3.8 cim22Configpuration

This object allows the grouping of sets of parameters (defined in cim22Setting 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.

```
   ( <oid-oc8> NAME 'cim22Configuration'
```

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 top
      MUST (cimName $ orderedCimModelPath)
      MAY (cimCaption $ cimDescription)
   )

   ( <oid-nf1> NAME 'cim22ConfigurationNameForm'
     OC cim22Configuration
     MUST (orderedCimModelPath)
   )

   ( <sr1> NAME 'cim22ConfigurationStructureRule'
     FORM cim22ConfigurationNameForm
   )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Configuration.

```
   ( <oid-oc8> NAME 'cim22ConfigurationContentRule'
     DESC 'The auxiliary classes that may be attached to
           cim22Configuration'
     AUX (cim22ConfigurationComponentAuxClass $
          cim22ElementConfigurationAuxClass $
          cim22CollectionConfigurationAuxClass $
          cim22SettingContextAuxClass)
   )
```

3.9 cim22ConfigurationComponentAuxClass

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

```
   ( <oid-at23> NAME 'cimConfigGroupRef'
     SYNTAX DN
   )
```

```
      ( <oid-at24> NAME 'cimConfigComponentRef'
        DESC 'A Configuration that is part of a "higher-level"
              Configuration.'
        SYNTAX DN
      )

      ( <oid-oc9> NAME 'cim22ConfigurationComponentAuxClass'
        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. Both attributes point to cim22Configuration
              objects.'
        SUP top AUXILIARY
        MAY (cimConfigGroupRef $ cimConfigComponentRef)
      )
```

3.10 cim22ElementConfigurationAuxClass

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.

```
      ( <oid-at25> NAME 'cimElementRef'
        DESC 'The ManagedSystemElement.'
        SYNTAX DN
      )

      ( <oid-at26> NAME 'cimConfigurationRef'
        DESC 'The Configuration object that groups the Settings and
              dependencies associated with the ManagedSystemElement.'
        SYNTAX DN
      )

      ( <oid-oc10> NAME 'cim22ElementConfigurationAuxClass'
        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. Attribute
              cimElementRef points to cim22ManagedSystemElement and
attribute
              cimConfigurationRef points to cim22Configuration.'
        SUP top AUXILIARY
```

```
        MAY (cimElementRef $ cimConfigurationRef)
```

```
    )
```

3.11 cim22CollectionConfigurationAuxClass

This auxiliary class relates a cim22Configuration object to one or more cim22CollectionOfMSEs objects. The cim22Configuration object represents a certain behavior, or a desired functional state for the associated collection.

```
    ( <oid-oc11> NAME 'cim22CollectionConfigurationAuxClass'
      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. Attribute
            cimCollectionRef points to cim22CollectionOfMSEs and
            attribute cimConfigurationRef points to cim22Configuration.'
      SUP top AUXILIARY
      MAY (cimCollectionRef $ cimConfigurationRef)
    )
```

3.12 cim22Setting

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.

```
    ( <oid-at27> NAME 'cimSettingID'
      DESC 'The identifier by which the Setting object is known.'
      SYNTAX string{256} SINGLE-VALUE
    )

    ( <oid-oc12> NAME 'cim22Setting'
      DESC 'The Setting class represents configuration-related and
            operational parameters for one or more
            ManagedSystemElement(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 top ABSTRACT
```

```
        MAY (cimSettingID $ cimCaption $ cimDescription)
      )
```

3.13 cim22ElementSettingAuxClass

This auxiliary class represents the association between managed system elements and the setting class(es) defined for them.

```
   ( <oid-at28> NAME 'cimSettingRef'
     DESC 'The Setting object associated with the ManagedSystemElement.'
     SYNTAX DN
   )

   ( <oid-oc13> NAME 'cim22ElementSettingAuxClass'
     DESC 'ElementSetting represents the association between
           ManagedSystemElements and the Setting class(es) defined for
           them. Attribute cimElementRef points to
           cim22ManagedSystemElement and attribute cimSettingRef points
           to cim22Setting.'
     SUP top AUXILIARY
     MAY (cimElementRef $ cimSettingRef)
   )
```

3.14 cim22DefaultSettingAuxClass

This auxiliary class represents the association between a cim22ManagedSystemElement and the single cim22Setting class that is defined to be the default setting for this element.

```
   ( <oid-oc14> NAME 'cim22DefaultSettingAuxClass'
     DESC 'DefaultSetting represents the association between a
           ManagedSystemElement and the single Setting class that is
           defined to be the default setting for this
           Element. Attribute cimElementRef points to
           cim22ManagedSystemElement and attribute cimSettingRef points
           to cim22Setting.'
     SUP cim22ElementSettingAuxClass AUXILIARY
     MAY (cimElementRef $ cimSettingRef)
   )
```

3.15 cim22SettingContextAuxClass

This auxiliary class associates 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.

```
   ( <oid-at29> NAME 'cimContextRef'
```

```
        DESC 'The Configuration object that aggregates the Setting.'
        SYNTAX DN
      )

      ( <oid-oc15> NAME 'cim22SettingContextAuxClass'
        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. Here, 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.  Attribute cimContextRef points to
              cim22Configuration and attribute cimSettingRef points to
              cim22Setting.'
        SUP top AUXILIARY
        MAY (cimContextRef $ cimSettingRef)
      )
```

3.16 cim22CollectionSettingAuxClass

This auxiliary class represents the association between a cim22CollectionOfMSEs class and the cim22Setting class(es) defined for them.

```
      ( <oid-oc16> NAME 'cim22CollectionSettingAuxClass'
        DESC 'CollectionSetting represents the association between a
              CollectionOfMSEs class and the Setting class(es) defined
              for them. Attribute cimCollectionRef points to
              cim22CollectionOfMSEs and attribute cimSettingRef points to
              cim22Setting.'
        SUP top AUXILIARY
        MAY (cimCollectionRef $ cimSettingRef)
      )
```

3.17 cim22System

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.

```
      ( <oid-at30> NAME 'cimNameFormat'
        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 string{64} SINGLE-VALUE
      )

      ( <oid-at31> NAME 'cimPrimaryOwnerContact'
        DESC 'A string that provides information on how the primary
              system owner can be reached (e.g. phone number, email
              address, ...).'
        SYNTAX string{256} SINGLE-VALUE
      )

      ( <oid-at32> NAME 'cimPrimaryOwnerName'
        DESC 'The name of the primary system owner.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at33> NAME 'cimRoles'
        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 string
      )

      ( <oid-oc17> NAME 'cim22System'
        DESC 'A cim22System 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 cim22LogicalElement
        MUST (cimCreationClassName $ cimName)
        MAY (cimNameFormat $ cimPrimaryOwnerContact $
             cimPrimaryOwnerName $ cimRoles)
      )

      ( <oid-nf2> NAME 'cim22SystemNameForm'
```

```
      OC cim22System
      MUST (orderedCimModelPath)
    )

    ( <sr2> NAME 'cim22SystemStructureRule'
      FORM cim22SystemNameForm
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22System.

```
    ( <oid-oc17> NAME 'cim22SystemContentRule'
      DESC 'The auxiliary classes that may be attached to cim22System'
      AUX (cim22SystemComponentAuxClass $ cim22CollectedMSEsAuxClass $
           cim22LogicalIdentityAuxClass $ cim22ElementSettingAuxClass $
           cim22DependencyAuxClass $ cim22ComponentAuxClass $
           cim22ElementConfigurationAuxClass $
           cim22ProvidesServiceToElementAuxClass)
    )
```

3.18 cim22ComputerSystem

This class is derived from cim22System 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).

```
    ( <oid-at36> NAME 'cimDedicated'
      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).'
      SYNTAX integer
    )

    ( <oid-oc18> NAME 'cim22ComputerSystem'
      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 cim22System
      MAY (cimDedicated)
    )
```

```
( <oid-nf11> NAME 'cim22ComputerSystemNameForm'
  OC cim22ComputerSystem
  MUST (orderedCimModelPath)
)

( <sr11> NAME 'cim22ComputerSystemStructureRule'
  FORM cim22ComputerSystemNameForm
)
```

3.19 cim22LogicalDevice

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.

```
( <oid-at37> NAME 'cimSystemCreationClassName'
  DESC 'The scoping System's CreationClassName.'
  SYNTAX string{256} SINGLE-VALUE
)

( <oid-at38> NAME 'cimSystemName'
  DESC 'The scoping System's Name.'
  SYNTAX string{256} SINGLE-VALUE
)

( <oid-at39> NAME 'cimDeviceID'
  DESC 'An address or other identifying information to uniquely
        name the LogicalDevice.'
  SYNTAX string{64} SINGLE-VALUE
)

( <oid-at40> NAME 'cimPowerManagementSupported'
  DESC 'Boolean indicating that the Device can be power managed -
        ie, put into a power save state. This boolean does not
        show 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 boolean SINGLE-VALUE
)

( <oid-at41> NAME 'cimPowerManagementCapabilities'
  DESC 'Shows 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" shows 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) shows that the
            SetPowerState method is supported. "Power Cycling
            Supported" (6) shows that the SetPowerState method can
            be invoked with the PowerState input variable set to 5
            ("Power Cycle"). "Timed Power On Supported" (7) shows
            that the SetPowerState method can be invoked with the
            PowerState input variable set to 5 ("Power Cycle") and the
            Time parameter set to a specific date and time, or
            interval, for power-on.'
      SYNTAX integer
    )

    ( <oid-at42> NAME 'cimAvailability'
      DESC 'The availability and status of the Device. For example, the
            Availability property shows 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") shows that
            the Device is known to be in a power save mode, but its
            exact status in this mode is unknown; 14 ("Power Save - Low
            Power Mode") shows that the Device is in a power save
            state but still functioning, and may exhibit degraded
            performance; 15 ("Power Save - Standby") describes that the
            Device is not functioning but could be brought to full
            power "quickly"; and value 17 ("Power Save - Warning")
            shows that the Device is in a warning state, though
            also in a power save mode.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at43> NAME 'cimStatusInfo'
      DESC 'StatusInfo is a string indicating whether the LogicalDevice
            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"), should be used.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at44> NAME 'cimLastErrorCode'
      DESC 'LastErrorCode captures the last error code reported by the
```

```
            LogicalDevice.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at45> NAME 'cimErrorDescription'
        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 string SINGLE-VALUE
      )

      ( <oid-at46> NAME 'cimErrorCleared'
        DESC 'ErrorCleared is a boolean property indicating that the
            error reported in LastErrorCode is now cleared.'
        SYNTAX boolean SINGLE-VALUE
      )

      ( <oid-oc19> NAME 'cim22LogicalDevice'
        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 cim22LogicalElement
        MUST (cimSystemCreationClassName $ cimSystemName $
            cimCreationClassName $ cimDeviceID)
        MAY (cimPowerManagementSupported $ cimAvailability $
            cimPowerManagementCapabilities $ cimStatusInfo $
            cimLastErrorCode $ cimErrorDescription $ cimErrorCleared)
      )

      ( <oid-nf3> NAME 'cim22LogicalDeviceNameForm'
        OC cim22LogicalDevice
        MUST (orderedCimModelPath)
      )

      ( <sr3> NAME 'cim22LogicalDeviceStructureRule'
        FORM cim22LogicalDeviceNameForm
        SUP <sr2>
      )
```

The following content rule specifies the auxiliary classes that may

be attached to cim22LogicalDevice.

```
    ( <oid-oc19> NAME 'cim22LogicalDeviceContentRule'
      DESC 'The auxiliary classes that may be attached to
cim22LogicalDevice'
      AUX (cim22RealizesAuxClass $ cim22LogicalIdentityAuxClass $
           cim22CollectedMSEsAuxClass $
           cim22ElementConfigurationAuxClass $
           cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
           cim22ProvidesServiceToElementAuxClass $
           cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
    )
```

3.20 cim22Service

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.

```
    ( <oid-at47> NAME 'cimStartMode'
      DESC 'StartMode is a string value indicating whether the Service
            is automatically started by a System, Operating System,
            etc. or only started on request.'
      SYNTAX string{10} SINGLE-VALUE
    )

    ( <oid-at48> NAME 'cimStarted'
      DESC 'Started is a boolean indicating whether the Service has
            been started (TRUE), or stopped (FALSE).'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-oc20> NAME 'cim22Service'
      DESC 'A cim22Service 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 cim22LogicalElement
      MUST (cimCreationClassName $ cimName $ cimSystemName $
            cimSystemCreationClassName)
      MAY (cimStartMode $ cimStarted)
    )

    ( <oid-nf4> NAME 'cim22ServiceNameForm'
```

OC cim22Service

```
      MUST (orderedCimModelPath)
    )

    ( <sr4> NAME 'cim22ServiceStructureRule'
      FORM cim22ServiceNameForm
      SUP <sr2>
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Service.

```
    ( <oid-oc20> NAME 'cim22ServiceContentRule'
      DESC 'The auxiliary classes that may be attached to cim22Service'
      AUX (cim22ServiceAccessBySAPAuxClass $
           cim22ServiceServiceDependencyAuxClass $
           cim22ServiceSAPDependencyAuxClass $
           cim22ProvidesServiceToElementAuxClass $
           cim22ServiceComponentAuxClass $
           cim22LogicalIdentityAuxClass $ cim22CollectedMSEsAuxClass $
           cim22ElementConfigurationAuxClass $
           cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
           cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
    )
```

3.21 cim22ServiceAccessPoint

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.

```
    ( <oid-oc21> NAME 'cim22ServiceAccessPoint'
      DESC 'cim22ServiceAccessPoint represents the ability to use or
            invoke a Service. Access points represent that a Service is
            made available to other entities for use.'
      SUP cim22LogicalElement
      MUST (cimCreationClassName $ cimName $ cimSystemName $
            cimSystemCreationClassName)
    )

    ( <oid-nf5> NAME 'cim22ServiceAccessPointNameForm'
      OC cim22ServiceAccessPoint
      MUST (orderedCimModelPath)
    )

    ( <sr5> NAME 'cim22ServiceAccessPointStructureRule'
      FORM cim22ServiceAccessPointNameForm
      SUP <sr2>
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22ServiceAccessPoint.

```
( <oid-oc21> NAME 'cim22ServiceAccessPointContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22ServiceAccessPoint'
  AUX (cim22ServiceAccessBySAPAuxClass $
       cim22ServiceSAPDependencyAuxClass $
       cim22SAPSAPDependencyAuxClass $
       cim22LogicalIdentityAuxClass $ cim22CollectedMSEsAuxClass $
       cim22ElementConfigurationAuxClass $
       cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
       cim22ProvidesServiceToElementAuxClass $
       cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
)
```

3.22 cim22DependencyAuxClass

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

```
( <oid-at49> NAME 'cimAntecedentRef'
  DESC 'Antecedent represents the independent object in this
        association.'
  SYNTAX DN
)

( <oid-at50> NAME 'cimDependentRef'
  DESC 'Dependent represents the object dependent on the
        Antecedent.'
  SYNTAX DN
)

( <oid-oc22> NAME 'cim22DependencyAuxClass'
  DESC 'cimDependency is a generic association used to establish
        dependency relationships between objects. Both attributes
        point to cim22ManagedSystemElement objects.'
  SUP top AUXILIARY
  MAY (cimAntecedentRef $ cimDependentRef)
)
```

3.23 cim22ServiceAccessBySAPAuxClass

This association identifies the access points for a service. For example, a printer may be accessed by Netware, MacIntosh or Windows service access points, potentially hosted on different systems.

```
( <oid-oc23> NAME 'cim22ServiceAccessBySAPAuxClass'
```

```
        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
              ServiceAccessPoints, potentially hosted on different
              Systems. Attribute cimAntecedentRef points to cim22Service
              and attribute cimDependentRef points to
              cim22ServiceAccessPoint.'
        SUP cim22DependencyAuxClass AUXILIARY
        MAY (cimAntecedentRef $ cimDependentRef)
      )
```

3.24 cim22ServiceServiceDependencyAuxClass

This is 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.

```
   ( <oid-at51> NAME 'cimTypeOfDependency'
     DESC '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.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-oc24> NAME 'cim22ServiceServiceDependencyAuxClass'
     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
           on 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 use the
           SAPs of this Service. This usage dependency is modeled via
           the ServiceSAPDependency association. Both attributes
           point to cim22Service objects.'
     SUP cim22DependencyAuxClass AUXILIARY
     MAY (cimAntecedentRef $ cimDependentRef $ cimTypeOfDependency)
   )
```

3.25 cim22ServiceSAPDependencyAuxClass

This class is 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.

```
( <oid-oc25> NAME 'cim22ServiceSAPDependencyAuxClass'
  DESC 'ServiceSAPDependency is an association between a
        Service and a ServiceAccessPoint indicating 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. Attribute
        cimAntecedentRef points to cim22ServiceAccessPoint and
        attribute cimDependentRef points to cim22Service.'
  SUP cim22DependencyAuxClass AUXILIARY
  MAY (cimAntecedentRef $ cimDependentRef)
)
```

3.26 cim22SAPSAPDependencyAuxClass

This class is 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.

```
( <oid-oc26> NAME 'cim22SAPSAPDependencyAuxClass'
  DESC 'SAPSAPDependency is an association between a
        ServiceAccessPoint and another ServiceAccessPoint
        indicating that the latter is required in order for the
        former ServiceAccessPoint 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 ProtocolEndpoints, to send the print request.
        Both attributes point to cim22ServiceAccessPoint objects.'
  SUP cim22DependencyAuxClass AUXILIARY
  MAY (cimAntecedentRef $ cimDependentRef)
)
```

3.27 cim22ProvidesServiceToElementAuxClass

This association is used to describe that cim22ManagedSystemElements may be dependent on the functionality of one or more Services.

```
( <oid-oc27> NAME 'cim22ProvidesServiceToElementAuxClass'
  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 AlertOnLAN
            Services to signal an incomplete or erroneous boot, and
            hardware-related errors. Attribute cimAntecedentRef points
            to cim22Service and attribute cimDependentRef points to
            cim22ManagedSystemElement.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.28 cim22RealizesAuxClass

This association defines the mapping between a logical device and the physical component that implements the device.

```
    ( <oid-oc28> NAME 'cim22RealizesAuxClass'
      DESC 'Realizes is the association that defines the mapping
            between a Logical Device and the physical component that
            implements the Device. Attribute cimAntecedentRef points to
            cim22PhysicalElement and attribute cimDependentRef points to
            cim22LogicalDevice.'
      SUP cim22DependencyAuxClass AUXILIARY
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.29 cim22ComponentAuxClass

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

```
    ( <oid-at52> NAME 'cimGroupComponentRef'
      DESC 'The parent element in the association.'
      SYNTAX DN
    )

    ( <oid-at53> NAME 'cimPartComponentRef'
      DESC 'The child element in the association.'
      SYNTAX DN
    )

    ( <oid-oc29> NAME 'cim22ComponentAuxClass'
      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. Both attributes point to
            cim22ManagedSystemElement objects.'
      SUP top AUXILIARY
```

```
        MAY (cimGroupComponentRef $ cimPartComponentRef)
      )
```

3.30 cim22SystemComponentAuxClass

This class is a specialization of the cim22ComponentAuxClass association that establishes part of relationships between a system and the managed system elements of which it is composed.

```
   ( <oid-oc30> NAME 'cim22SystemComponentAuxClass'
     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. Attribute
           cimGroupComponentRef points to cim22System and attribute
           cimPartComponentRef points to cim22ManagedSystemElement.'
     SUP cim22ComponentAuxClass AUXILIARY
     MAY (cimGroupComponentRef $ cimPartComponentRef)
   )
```

3.31 cim22ServiceComponentAuxClass

This auxiliary class models a set of subordinate services that are aggregated together to form a higher-level service.

```
   ( <oid-oc31> NAME 'cim22ServiceComponentAuxClass'
     DESC 'The ServiceComponent aggregation models a set of
           subordinate Services that are aggregated together to form a
           higher-level service. Both attributes point to cim22Service
           objects.'
     SUP cim22ComponentAuxClass AUXILIARY
     MAY (cimGroupComponentRef $ cimPartComponentRef)
   )
```

3.32 cim22Product

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.

```
   ( <oid-at54> NAME 'cimIdentifyingNumber'
     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 string{64} SINGLE-VALUE
   )
```

```
    ( <oid-at55> NAME 'cimVendor'
      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 CIM Solution Exchange Standard.'
      SYNTAX string{256} SINGLE-VALUE
    )

    ( <oid-at56> NAME 'cimSKUNumber'
      DESC 'Product SKU (stock keeping unit) information.'
      SYNTAX string{64} SINGLE-VALUE
    )

    ( <oid-oc32> NAME 'cim22Product'
      DESC 'cim22Product 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 that 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 cim22Product.'
      SUP top
      MUST (cimIdentifyingNumber $ cimName $ cimVendor $ cimVersion $
            orderedCimModelPath)
      MAY (cimCaption $ cimDescription $ cimSKUNumber)
    )

    ( <oid-nf6> NAME 'cim22ProductNameForm'
      OC cim22Product
      MUST (orderedCimModelPath)
    )

    ( <sr6> NAME 'cim22ProductStructureRule'
      FORM cim22ProductNameForm
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Product.

```
    ( <oid-oc32> NAME 'cim22ProductContentRule'
      DESC 'The auxiliary classes that may be attached to cim22Product'
      AUX (cim22ProductParentChildAuxClass $
           cim22CompatibleProductAuxClass $
           cim22ProductProductDependencyAuxClass $
           cim22ProductSupportAuxClass $  cim22ProductFRUAuxClass $
           cim22ProductPhysicalElementsAuxClass $
           cim22FRUIncludesProductAuxClass)
    )
```

3.33 cim22ProductParentChildAuxClass

The association defines a parent child hierarchy among products. For example, a product may come bundled with other products.

```
( <oid-at57> NAME 'cimParentRef'
  DESC 'The parent Product in the association.'
  SYNTAX DN
)

( <oid-at58> NAME 'cimChildRef'
  DESC 'The child Product in the association.'
  SYNTAX DN
)

( <oid-oc33> NAME 'cim22ProductParentChildAuxClass'
  DESC 'The ProductParentChild association defines a parent
        child hierarchy among Products. For example, a Product may
        come bundled with other Products. Both attributes point to
        cim22Product objects.'
  SUP top AUXILIARY
  MAY (cimParentRef $ cimChildRef)
)
```

3.34 cim22CompatibleProductAuxClass Association

This 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.

```
( <oid-at59> NAME 'cimProductRef'
  DESC 'The Product for which compatible offerings are defined.'
  SYNTAX DN
)

( <oid-at60> NAME 'cimCompatibleProductRef'
  DESC 'The compatible Product.'
  SYNTAX DN
)

( <oid-at61> NAME 'cimCompatibilityDescription'
  DESC 'CompatibilityDescription is a free-form string defining
        how the two referenced Products interoperate or are
        compatible, any limitations to compatibility, etc.'
  SYNTAX string SINGLE-VALUE
)
```

```
( <oid-oc34> NAME 'cim22CompatibleProductAuxClass'
  DESC 'CompatibleProduct is an association between Products
        that 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. The
        string property, CompatibilityDescription, defines how the
        Products interoperate or are compatible, any limitations
        regarding interoperability or installation, ...  Both
        reference attributes point to cim22Product objects.'
  SUP top AUXILIARY
  MAY (cimProductRef $ cimCompatibleProductRef $
       cimCompatibilityDescription)
)
```

3.35 cim22ProductProductDependencyAuxClass

This association is 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.

```
( <oid-at62> NAME 'cimRequiredProductRef'
  DESC 'The required Product.'
  SYNTAX DN
)

( <oid-at63> NAME 'cimDependentProductRef'
  DESC 'The Product that is dependent on another Product.'
  SYNTAX DN
)

( <oid-oc35> NAME 'cim22ProductProductDependencyAuxClass'
  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.
        Both reference attributes point to cim22Product objects.'
  SUP top AUXILIARY
  MAY (cimRequiredProductRef $ cimDependentProductRef $
       cimTypeOfDependency)
)
```

3.36 cim22SupportAccess

This class defines how to obtain help for a product.

```
    ( <oid-at64> NAME 'cimSupportAccessId'
      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 string{256} SINGLE-VALUE
    )

    ( <oid-at65> NAME 'cimCommunicationInfo'
      DESC 'CommunicationInfo provides the details of the
            CommunicationMode. For example, if the CommunicationMode is
            "Phone", CommunicationInfo specifies the phone number to be
            called.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-at66> NAME 'cimCommunicationMode'
      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.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at67> NAME 'cimLocale'
      DESC 'Locale defines the geographic region and/or language
            dialect to which this Support resource pertains.'
      SYNTAX string{64} SINGLE-VALUE
    )

    ( <oid-oc36> NAME 'cim22SupportAccess'
      DESC 'The cim22SupportAccess association defines how to obtain
            help for a Product.'
      SUP top
      MUST (cimSupportAccessId $ orderedCimModelPath)
      MAY (cimCommunicationInfo $ cimCommunicationMode $ cimLocale $
           cimDescription)
    )

    ( <oid-nf7> NAME 'cim22SupportAccessNameForm'
      OC cim22SupportAccess
      MUST (orderedCimModelPath)
    )

    ( <sr7> NAME 'cim22SupportAccessStructureRule'
      FORM cim22SupportAccessNameForm
    )
```

The following content rule specifies the auxiliary classes that may

be attached to cim22SupportAccessProduct.

```
( <oid-oc36> NAME 'cim22SupportAccessContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22SupportAccess'
  AUX (cim22ProductSupportAuxClass)
)
```

3.37 cim22ProductSupportAuxClass

This auxiliary class represents 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.

```
( <oid-at68> NAME 'cimSupportRef'
  DESC 'Support for the Product.'
  SYNTAX DN
)

( <oid-oc37> NAME 'cim22ProductSupportAuxClass'
  DESC 'cimProductSupport 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 help for
        multiple Products. Attribute cimProductRef points to
        cim22Product and attribute cimSupportRef points to
        cim22SupportAccess.'
  SUP top AUXILIARY
  MAY (cimProductRef $ cimSupportRef)
)
```

3.38 cim22FRU

This class is a 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'.

```
( <oid-at69> NAME 'cimFRUNumber'
  DESC 'FRU ordering information.'
  SYNTAX string{64} SINGLE-VALUE
)
```

```
( <oid-at70> NAME 'cimRevisionLevel'
  DESC 'The FRU's revision level.'
  SYNTAX string{64} SINGLE-VALUE
)

( <oid-oc38> NAME 'cim22FRU'
  DESC 'The cimFRU class is a vendor-defined collection of
        Products and/or PhysicalElements 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". '
  SUP top
  MUST (cimFRUNumber $ cimIdentifyingNumber $ cimVendor $
        orderedCimModelPath)
  MAY (cimCaption $ cimDescription $ cimName $ cimRevisionLevel)
)

( <oid-nf8> NAME 'cim22FRUNameForm'
  OC cim22FRU
  MUST (orderedCimModelPath)
)

( <sr8> NAME 'cim22FRUStructureRule'
  FORM cim22FRUNameForm
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22FRU.

```
( <oid-oc38> NAME 'cim22FRUContentRule'
  DESC 'The auxiliary classes that may be attached to cim22FRU'
  AUX (cim22ProductFRUAuxClass $ cim22FRUPhysicalElementsAuxClass $
       cim22FRUIncludesProductAuxClass)
)
```

3.39 cim22ProductFRUAuxClass

This auxiliary class provides information regarding what product components have been or are being replaced and uses the previously defined attribute cimProductRefs.

```
( <oid-at71> NAME 'cimFRURef'
  DESC 'The FRU.'
  SYNTAX DN
)

( <oid-oc39> NAME 'cim22ProductFRUAuxClass'
  DESC 'cimProductFRU 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. Attribute cimProductRef points to
            cim22Product and attribute cimFRURef points to cim22FRU.'
      SUP top AUXILIARY
      MAY (cimProductRef $ cimFRURef)
    )
```

3.40 cim22ProductPhysicalElementsAuxClass

Shows the physical elements that make up a product. It uses the previously defined cimProductsRefs attribute and defines cimPhysicalElementRefs.

```
  ( <oid-at72> NAME 'cimComponentRef'
    DESC 'The PhysicalElement that is a part of the Product.'
    SYNTAX DN
  )

  ( <oid-oc40> NAME 'cim22ProductPhysicalElementsAuxClass'
    DESC 'Shows the PhysicalElements that make up a
          Product. Attribute cimProductRef points to
          cim22Product and attribute cimComponentRef points to
          cim22PhysicalElement.'
    SUP top AUXILIARY
    MAY (cimProductRef $ cimComponentRef)
  )
```

3.41 cim22FRUPhysicalElementsAuxClass

This auxiliary class shows the physical elements that make up a FRU and uses previously defined attributes.

```
  ( <oid-oc41> NAME 'cim22FRUPhysicalElementsAuxClass'
    DESC 'Shows the PhysicalElements that make up a FRU. Attribute
          cimFRURef points to cim22FRU and attribute cimComponentRef
          points to cim22PhysicalElement.'
    SUP top AUXILIARY
    MAY (cimFRURef $ cimComponentRef)
  )
```

3.42 cim22FRUIncludesProductAuxClass

This auxiliary class shows that a FRU may be composed of other product(s). It uses previously defined attributes.

```
( <oid-oc42> NAME 'cim22FRUIncludesProductAuxClass'
  DESC 'Shows that a FRU may be composed of other
        Product(s). Attribute cimFRURef points to cim22FRU and
        attribute cimComponentRef points to cim22Product.'
  SUP top AUXILIARY
  MAY (cimFRURef $ cimComponentRef)
)
```

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

[4] CIM, "CIM Core Model, v2.2".

5. Acknowledgement

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

6. Editor's Address

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