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

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

LDAP Schema for the DMTF Application CIM v2.1 Model
Filename: draft-moats-dmtf-application-ldap-01.txt

Status of this Memo

Abstract

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

1. Introduction

This draft presents a LDAPv3 [1,2] schema for the DMTF CIM Application 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. 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:

cim22DirectorySpecificationFileAuxClass
cim22ActionSequenceAuxClass
cim22SoftwareFeatureSoftwareElementsAuxClass
cim22ToDirectorySpecificationAuxClass
cim22FromDirectorySpecificationAuxClass
cim22ToDirectoryActionAuxClass
cim22FromDirectoryActionAuxClass
cim22SoftwareFeatureSAPImplementationAuxClass
cim22ApplicationSystemSoftwareFeatureAuxClass
cim22InstalledSoftwareElementAuxClass

Also, the following structure rules defined here need to be added to the structure rule for cimAssociationInstance: <sr25>, <sr26>, <sr27>, <sr28>,

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Note: all attribute, object class, and name form OIDs are place holders, and syntax OIDs in definitions have been replaced by names for clarity.

3.1 cim22ApplicationSystem

This class represents an application or a software system that supports a particular business function and that can be managed as independent units. The cim22SoftwareFeature class allows such a system to be decomposed into its functional pieces. The software features for a particular application or software system are located using cim22ApplicationSystemSoftwareFeatureAuxClass.

```
( <oid-oc96> NAME 'cim22ApplicationSystem'
  DESC 'The cim22ApplicationSystem class is used to represent an
        application or a software system that supports a particular
        business function and that can be managed as an independent
        units. Such a system can be decomposed into its functional
        components using the cim22SoftwareFeature class. The
        software features for a particular application or software
        system are located using the
        cim22ApplicationSystemSoftwareFeature association.'
  SUP cim22System
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22ApplicationSystem.

```
( <oid-oc96> NAME 'cim22ApplicationSystemContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22ApplicationSystem'
  AUX (cim22ApplicationSystemSoftwareFeatureAuxClass)
)
```

3.2 cim22SoftwareElement

This class decomposes a cim22SoftwareFeature object into a set of individually manageable or deployable parts for a particular platform. A software element's underlying hardware architecture and operating system uniquely identifies its platform. As such, to understand the details of how the functionality of a particular software feature is provided on a particular platform, the cim22SoftwareElement objects referenced by cim22SoftwareFeatureSoftwareElementAuxClass are organized in disjoint sets based on the targetOperatingSystem property. A cim22SoftwareElement object captures the management details of a part or component in one of four states characterized by the SoftwareElementState property.

```
( <oid-at171> NAME 'cimSoftwareElementState'
  DESC ' The SoftwareElementState is defined in this model to
         identify various states of a software elements life
         cycle. - A software element in the deployable state
         describes the details necessary to successful distribute
         it and the details (conditions and actions) required to
         create a software element in the installable state (i.e,
         the next state). - A software element in the installable
         state describes the details necessary to successfully
         install it and the details (conditions and actions
         required to create a software element in the executable
```

state (i.e., the next state). - A software element in the

```
            executable state describes the details necessary to
            successfully start it and the details (conditions and
            actions required to create a software element in the
            running state (i.e., the next state). - A software element
            in the running state describes the details necessary to
            monitor and operate on a start element.  May be used as an
            RDN.  Allowed values are: "Deployable", "Installable",
            "Executable", "Running".'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at172> NAME 'cimSoftwareElementID'
      DESC ' This is an identifier for this software element and is
            designed to be used with other keys to
            create a unique representation of this SoftwareElement.
            May be used as an RDN.'
      SYNTAX string{256} SINGLE-VALUE
    )

    ( <oid-at173> NAME 'cimTargetOperatingSystem'
      DESC ' The Target Operating System property allows the provider
            to specify the operating system environment. The value of
            this property does not ensure binary executable. Two other
            pieces of information are needed. First, the version of
            the OS needs to be specified. using the OS Version
            Check. The second piece of information is the architecture
            the OS runs on. This information is capture with the
            ArchitectureCheck class. The combination of these
            constructs allows the provider to clearly identify the
            level of OS required for a particular software element.
            May be used as an RDN.  Allowed values are: "Unknown",
            "Other", "MACOS", "ATTUNIX", "DGUX", "DECNT", "Digital
            Unix", "OpenVMS", "HPUX", "AIX", "MVS", "OS400", "OS/2",
            "JavaVM", "MSDOS", "WIN3x", "WIN95", "WIN98", "WINNT",
            "WINCE", "NCR3000", "NetWare", "OSF", "DC/OS", "Reliant
            UNIX", "SCO UnixWare", "SCO OpenServer", "Sequent",
            "IRIX", "Solaris", "SunOS", "U6000", "ASERIES",
            "TandemNSK", "TandemNT", "BS2000", "LINUX", "Lynx",
            "XENIX", "VM/ESA", "Interactive UNIX", "BSDUNIX",
            "FreeBSD", "NetBSD", "GNU Hurd", "OS9", "MACH Kernel",
            "Inferno", "QNX", "EPOC", "IxWorks", "VxWorks", "MiNT",
            "BeOS", "HP MPE", "NextStep", "PalmPilot", "Rhapsody".'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-at174> NAME 'cimOtherTargetOS'
      DESC ' The otherTargetOS property records the manufacturer and
```

operating system type for a software element when the

```
               targetOperatingSystem property has a value of 1 ("Other")
               For all other values of TargetOperatingSystem, the
               OtherTargetOS property is to be NULL.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at175> NAME 'cimBuildNumber'
        DESC 'The internal identifier for this compilation of this
              software element.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at176> NAME 'cimCodeSet'
        DESC 'Array defining the character sets'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at177> NAME 'cimIdentificationCode'
        DESC ' The value of this property is the manufacturer's
               identifier for this software element. Often this will be a
               stock keeping unit (SKU) or a part number.'
        SYNTAX string{64} SINGLE-VALUE
      )

      ( <oid-at178> NAME 'cimLanguageEdition'
        DESC 'The value of this property identifies the language edition
              of this software element. The language codes defined in ISO
              639 should be used. Where the software element represents
              multi-lingual or international version of a product, the
              string multilingual should be used.'
        SYNTAX string{32} SINGLE-VALUE
      )

      ( <oid-oc97> NAME 'cim22SoftwareElement'
        DESC 'The cim22SoftwareElement class is used to decompose a
              cim22SoftwareFeature object into a set of individually
              manageable or deployable parts for a particular platform. A
              software element's platform is uniquely identified by its
              underlying hardware architecture and operating system (for
              example Sun Solaris on Sun Sparc or Windows NT on
              Intel). As such, to understand the details of how the
              functionality of a particular software feature is provided
              on a particular platform, the cim22SoftwareElement objects
              referenced by cim22SoftwareFeatureSoftwareElement
              associations are organized in disjoint sets based on the
              TargetOperatingSystem property. A cim22SoftwareElement
              object captures the management details of a part or
```

component in one of four states characterized by the

```
             SoftwareElementState property.'
       SUP cim22LogicalElement
       MAY (cimName $ cimVersion $ cimSoftwareElementState $
            cimSoftwareElementID $ cimTargetOperatingSystem $
            cimOtherTargetOS $ cimManufacturer $ cimBuildNumber $
            cimSerialNumber $ cimCodeSet $ cimIdentificationCode $
            cimLanguageEdition)
    )

    ( <oid-nf27> NAME 'cim22SoftwareElementNameForm'
      OC cim22SoftwareElement
      MUST (orderedCimModelPath)
    )

    ( <sr27> NAME 'cim22SoftwareElementStructureRule'
      FORM cim22SoftwareElementNameForm
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22SoftwareElement.

```
    ( <oid-oc97> NAME 'cim22SoftwareElementContentRule'
      DESC 'The auxiliary classes that may be attached to
            cim22SoftwareElement'
      AUX (cim22SoftwareFeatureSoftwareElementsAuxClass $
           cim22InstalledSoftwareElementAuxClass $
           cim22LogicalIdentityAuxClass $ cim22CollectedMSEsAuxClass $
           cim22ElementConfigurationAuxClass $
           cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
           cim22ProvidesServiceToElementAuxClass $
           cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
    )
```

3.3 cim22SoftwareFeature

This class defines a particular function or capability of a product or application system. It captures the level of granularity that is significant to a consumer or user of a product rather than the units that reflect how the product is built or packaged, which is captured using the cim22SoftwareElement class. When a software feature can exist on multiple platforms or operating systems (for example, a client component of a three tiered client/server applications might run on Solaris, Windows NT, and Windows 95), a software feature is a collection of all the software elements for these different platforms. Here, the users of the model must be aware of this since typically they will be interested in a sub-collection of the software elements required for a particular platform. Software Features are

always defined in the context of a cim22Product class using DIT

```
   structure rules since features are delivered through products.
   Optionally, software features from one or more products can be
   organized into application systems using the
   cim22ApplicationSystemSoftwareFeatureAuxClass object.

      ( <oid-at179> NAME 'cimProductName'
        DESC 'Commonly used Product name'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-oc98> NAME 'cim22SoftwareFeature'
        DESC 'The cim22SoftwareFeature class defines a particular function
              or capability of a product or application system. This
              class in intended to capture the level of granularity that
              is meaningful to a consumer or user of a product rather
              than the units that reflect how the product is built or
              packaged. The latter detailed is captured using a
              cim22SoftwareElement class. When a software feature can
              exist on multiple platforms or operating systems (for
              example, a client component of a three tiered client/server
              applications might run on Solaris, Windows NT, and Windows
              95), a software feature is a collection of all the software
              elements for these different platforms. In this case, the
              users of the model must be aware of this situation since
              typically they will be interested in a sub-collection
              of the software elements required for a particular
              platform. Software Features are always defined in the
              context of a cim22Product class using the
              cim22ProductSoftwareFeature association since features are
              delivered through products. Optionally, software features
              from one or more products can be organized into application
              systems using the cim22ApplicationSystemSoftwareFeature
              association.'
        SUP cim22LogicalElement
        MUST (cimIdentifyingNumber $ cimProductName $ cimVendor $
              cimVersion)
        MAY (cimName)
      )

      ( <oid-nf28> NAME 'cim22SoftwareFeatureNameForm'
        OC cim22SoftwareFeature
        MUST (orderedCimModelPath)
      )

      ( <sr28> NAME 'cim22SoftwareFeatureStructureRule'
        FORM cim22SoftwareFeatureNameForm
        SUP <sr6>
```

)

The following content rule specifies the auxiliary classes that may be attached to cim22SoftwareFeature.

```
( <oid-oc98> NAME 'cim22SoftwareFeatureContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22SoftwareFeature'
  AUX (cim22SoftwareFeatureSoftwareElementsAuxClass $
       cim22SoftwareFeatureServiceImplementationAuxClass $
       cim22SoftwareFeatureSAPImplementationAuxClass $
       cim22ApplicationSystemSoftwareFeatureAuxClass $
       cim22LogicalIdentityAuxClass $ cim22CollectedMSEsAuxClass $
       cim22ElementConfigurationAuxClass $
       cim22ElementSettingAuxClass $ cim22DependencyAuxClass $
       cim22ProvidesServiceToElementAuxClass $
       cim22ComponentAuxClass $ cim22SystemComponentAuxClass)
)
```

3.4 cim22Check

A Check is a condition or characteristic that is expected to be true in an environment defined or scoped by an instance of a cim22ComputerSystem. They are associated with a particular software element and are organized into one of two groups using the Phase property. Conditions that are expected to be satisfied when a software element is in a particular environment are known as in-state conditions. Conditions that need to be satisfied to transition the current software element to its next state are known as next-state conditions.

```
( <oid-at180> NAME 'cimCheckID'
  DESC 'An identifier used in conjunction with other keys to
        uniquely identify the check'
  SYNTAX string SINGLE-VALUE
)

( <oid-at181> NAME 'cimCheckMode'
  DESC 'The CheckMode property is used to indicate whether the
        condition is expected to exist or not exist in the
        environment. When the value is True, the condition is
        expected to exist (e.g., a file is expected to be on a
        system) so invoke() is expected to return True. When the
        value is False, the condition is not expect to exist (e.g.,
        a file is not to be on a system) so invoke is expected to
        return false'
  SYNTAX boolean SINGLE-VALUE
)
```

```
      ( <oid-oc99> NAME 'cim22Check'
```

```
        DESC 'A Check is a condition or characteristic that is expected
              to be true in an environment defined or scoped by an
              instance of a cim22ComputerSystem. The checks associated
              with a particular software element are organized into one
              of two groups using the Phase property of the
              cim22SoftwareElementChecks association. Conditions that are
              expected to be satisfied when a software element is in a
              particular environment are known as in-state
              conditions. Conditions that need to be satisfied in order
              to transition the current software element to its next
              state are known as next-state conditions A
              cim22ComputerSystem object represents the environment in
              which cim22SoftwareElements are already installed or in
              which cim22SoftwareElements will be installed. For the case
              in which a software element is already installed, the
              cim22InstalledSoftwareElement association is used to
              identify the cim22ComputerSystem object that represents the
              "environment". When a software elements is being
              distributed and installed on a different computer system,
              the cim22ComputerSystem object for the targeted system is
              the environment.'
        SUP top
        MUST (orderedCimModelPath)
        MAY (cimName $ cimVersion $ cimSoftwareElementState $
             cimSoftwareElementID $ cimTargetOperatingSystem $
             cimCheckID $ cimDescription $ cimCaption $ cimCheckMode)
      )

      ( <oid-nf25> NAME 'cim22CheckNameForm'
        OC cim22Check
        MUST (orderedCimModelPath)
      )

      ( <sr25> NAME 'cim22CheckStructureRule'
        FORM cim22CheckNameForm
        SUP <sr27>
      )
```

3.5 cim22DirectorySpecification

A directory specification captures the major directory structure of a software element and organizes the files of a software element into manageable units that can be relocated on a computer system.

```
      ( <oid-at182> NAME 'cimDirectoryType'
        DESC 'The DirectoryType property characterizes the type of
              directory being described.'
```

```
        SYNTAX integer SINGLE-VALUE
```

```
    )

    ( <oid-at183> NAME 'cimDirectoryPath'
      DESC 'The DirectoryPath property is used to capture the name of a
            directory. The value supplied by an application provider is
            actually a default or recommended path name. The value can
            be changed for a particular environment.'
      SYNTAX string SINGLE-VALUE
    )

    ( <oid-oc100> NAME 'cim22DirectorySpecification'
      DESC 'The cim22DirectorySpecification class captures the major
            directory structure of a software element. This class is
            used to organize the files of a software element into
            manageable units that can be relocated on a computer
            system.'
      SUP cim22Check
      MAY (cimDirectoryType $ cimDirectoryPath)
    )
```

The following content rule specifies the auxiliary classes that may be attached to cim22DirectorySpecification.

```
    ( <oid-oc100> NAME 'cim22DirectorySpecificationContentRule'
      DESC 'The auxiliary classes that may be attached to
            cim22DirectorySpecification'
      AUX (cim22DirectorySpecificationFileAuxClass $
           cim22ToDirectorySpecificationAuxClass $
           cim22FromDirectorySpecificationAuxClass)
    )
```

3.6 cim22ArchitectureCheck

Architecture checks specify the hardware platform a software element can run on. Its attributes are compared with the corresponding processor attributes. As long as there is at least one processor that satisfies the details of the condition, the check is satisfied.

```
    ( <oid-at184> NAME 'cimArchitectureType'
      DESC 'The ArchitectureType property identifies a particular type
            of architecture or architecture family that is required to
            properly execute a particular software element. The intent
            is to capture the details about the machine instructions
            exploited by the executables of the software element.'
      SYNTAX integer SINGLE-VALUE
    )

    ( <oid-oc101> NAME 'cim22ArchitectureCheck'
```

```
        DESC 'The Architecture Check specifies the hardware platform a
              software element can run on. The details of this checks are
              compared with the corresponding details found in a
              cim22Processor object referenced by a
              cim22ComputerSystemProcessor association for the
              cim22ComputerSystem object that describes the
              environment. As long as there is at least one cim22Processor
              that satisfies the details of the condition, the check is
              satisfied. In other words, all the processors on the
              relevant computer system do not need to satisfy the
              condition. There needs to be at least one.'
        SUP cim22Check
        MAY (cimArchitectureType)
      )
```

3.7 cim22MemoryCheck

Memory checks specify the minimum amount of memory that a system must have available. When an operating system has more free physical memory than the value specified in MemorySize, the condition is satisfied.

```
   ( <oid-at185> NAME 'cimMemorySize'
     DESC 'The amount of memory that needs to exist on a computer
           system for a software element to executing properly.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-oc102> NAME 'cim22MemoryCheck'
     DESC 'The MemoryCheck specifies a condition for the minimum
           amount of memory that needs to be available on a
           system. The amount is specified in the MemorySize
           property. The details of this checks are compared with the
           value of the FreePhysicalMemory property of the
           cim22OperatingSystem object referenced by an InstalledOS
           association for the cim22ComputerSystem object that
           describes the environment. When the value of
           FreePhyscalMemory property is greater than or equal to the
           value specified in MemorySize, the condition is satisfied.'
     SUP cim22Check
     MAY (cimMemorySize)
   )
```

3.8 cim22DiskSpaceCheck

Disk space checks specify the minimum amount of disk space that a system must have available. When a file system have more available

space than the value specified in availableDiskSpace, the condition

is satisfied.

```
( <oid-at186> NAME 'cimAvailableDiskSpace'
  SYNTAX integer SINGLE-VALUE
)

( <oid-oc103> NAME 'cim22DiskSpaceCheck'
  DESC 'The Disk Space Check the amount of disk space the needs to
        be available on the system. The amount is specified in the
        AvailableDiskSpace property. The details of this checks are
        compared with the value of the AvailableSpace property of
        the cim22FileSystem object associated with the
        cim22ComputerSystem object that describes the
        environment. When the value of AvailableSpace property is
        greater than or equal to the value specified in
        AvailableDiskSpace, the condition is satisfied.'
  SUP cim22Check
  MAY (cimAvailableDiskSpace)
)
```

3.9 cim22SwapSpaceCheck

Swap space checks specify the minimum amount of swap space that a system must have available. When a system have more swap space available that the value in swapSpaceSize, the condition is satisfied.

```
( <oid-at187> NAME 'cimSwapSpaceSize'
  DESC 'The SwapSpaceSize property specifies the minimum number of
        Kilo bytes of swap space that needs to be available on the
        target system.'
  SYNTAX integer SINGLE-VALUE
)

( <oid-oc104> NAME 'cim22SwapSpaceCheck'
  DESC 'The Swap Space Check specifies the amount of swap space
        that needs to be available on the system. The amount is
        specified in the SwapSpaceSize property. The details of
        this checks are compared with the corresponding details
        found in a cim22OperatingSystem object referenced by
        InstalledOS association for the cim22ComputerSystem object
        describing the environment. When the value of
        TotalSwapSpaceSize property is greater than or equal to the
        value specified in SwapSpacesize, the condition is
        satisfied.'
  SUP cim22Check
  MAY (cimSwapSpaceSize)
```

)

3.10 cim22OSVersionCheck

This class specifies the versions of the OS that can support this software element. This check can be for a specific, minimum, maximum or a range of releases of an OS. To specify a specific version the minimum and maximum versions must be the same. To specify a minimum, the minimum version needs only be specified. To specify a maximum version, the maximum version needs only be specified. To specify a range both minimum and maximum version need to be specified.

```
( <oid-at188> NAME 'cimMinimumVersion'
  DESC 'Minimum version of required operating system. The value is
        encoded as <major>.<minor>.<revision> or
        <major>.<minor><letter revision> '
  SYNTAX string SINGLE-VALUE
)

( <oid-at189> NAME 'cimMaximumVersion'
  DESC 'Maximum version of required operating system. The value is
        encoded as <major>.<minor>.<revision> or
        <major>.<minor><letter revision> '
  SYNTAX string SINGLE-VALUE
)

( <oid-oc105> NAME 'cim22OSVersionCheck'
  DESC 'The OS Version Check class specifies the versions of the OS
        that can support this software element. This check can be
        for a specific, minimum, maximum or a range of releases of
        an OS. To specify a specific version the minimum and
        maximum versions must be the same. To specify a minimum,
        the minimum version needs only be specified. To specify a
        maximum version, the maximum version needs only be
        specified. To specify a range both minimum and maximum
        version need to be specified. The type of operating system
        is specified in the TargetOperatingSystem property of the
        owning SoftwareElement. The details of this checks are
        compared with the corresponding details found in a
        cim22OperatingSystem object referenced by InstalledOS
        association for the cim22ComputerSystem object that
        describes the environment. As long as there is at least one
        cim22OperatingSystem that satisfies the details of the
        condition, the check is satisfied. In other words, all the
        operating systems on the relevant computer system do not
        need to satisfy the condition. There needs to be at least
        one. Also, note the the OSType property of the
        cim22OperatingSystem class must match the type of the
        TargetOperatingSystem property.'
```

```
        SUP cim22Check
```

```
        MAY (cimMinimumVersion $ cimMaximumVersion)
      )
```

3.11 cim22SoftwareElementVersionCheck

This class specifies a type of software element that must exist in the environment. This check can be for a specific, minimum, maximum or a range of versions. To specify a specific version the lower and upper versions must be the same. To specify a minimum the lower version needs only be specified. To specify a maximum version the upper version needs only be specified. To specify a range both upper and lower version need to be specified. The details of this checks are compared with the corresponding details found in a cim22SoftwareElement object referenced by an InstalledSoftwareElement association for the cim22ComputerSystem object. As long as there is at least one cim22SoftwareElement that satisfies the details of the condition, the check is satisfied.

```
   ( <oid-at190> NAME 'cimSoftwareElementName'
     DESC 'The name of the software element being checked.'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-at191> NAME 'cimLowerSoftwareElementVersion'
     DESC 'The minimum version of a software elements being checked.'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-at192> NAME 'cimUpperSoftwareElementVersion'
     DESC 'The maximum version of a software elements being checked.'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-at193> NAME 'cimSoftwareElementStateDesired'
     DESC 'The state of the software element being checked.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-at194> NAME 'cimTargetOperatingSystemDesired'
     DESC 'The target operating system of the software element being
           checked.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-oc106> NAME 'cim22SoftwareElementVersionCheck'
     DESC 'The Software Element Version Check class specifies a type
           of software element that must exist in the
           environment. This check can be for a specific, minimum,
```

```
            maximum or a range of versions. To specify a specific
            version the lower and upper versions must be the same. To
            specify a minimum the lower version needs only be
            specified. To specify a maximum version the upper version
            needs only be specified. To specify a range both upper and
            lower version need to be specified. The details of this
            checks are compared with the corresponding details found in
            a cim22SoftwareElement object referenced by an
            InstalledSoftwareElement association for the
            cim22ComputerSystem object. As long as there is at least one
            cim22SoftwareElement that satisfies the details of the
            condition, the check is satisfied. In other words, all the
            software elements on the relevant computer system do not
            need to satisfy the condition. There needs to be at least
            one.'
      SUP cim22Check
      MAY (cimSoftwareElementName $ cimLowerSoftwareElementVersion $
           cimUpperSoftwareElementVersion $
           cimSoftwareElementStateDesired $
           cimTargetOperatingSystemDesired)
   )
```

3.12 cim22FileSpecification

A cim22FileSpecification object identifies a file that is either to be on or off the system. cim22DirectorySpecificationFileAuxClass identifies the directory the file is to be located in.

```
   ( <oid-at195> NAME 'cimFileName'
     DESC 'Either the name of the file or the name of the file with a
           directory prefix.'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-at196> NAME 'cimCreateTimeStamp'
     DESC 'The creation date and time of the file.'
     SYNTAX generalizedTime SINGLE-VALUE
   )

   ( <oid-at197> NAME 'cimFileSize'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-at198> NAME 'cimCheckSum'
     DESC 'The File Checksum property is a checksum calculated as the
           16-bit sum of the first 32 bytes of the file.'
     SYNTAX integer SINGLE-VALUE
```

)

```
      ( <oid-at199> NAME 'cimCRC1'
        DESC 'The File CRC 1 property is the CRC value calculated using
              the middle 512K bytes.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at200> NAME 'cimCRC2'
        DESC 'The File CRC 2 is the CRC value for the middle 512K bytes
              with a offset modulo 3 to the start of the file of zero.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-at201> NAME 'cimMD5Checksum'
        DESC 'The MD5 algorithm is a well-known algorithm for computing a
              128-bit checksum for any file or object. The likelihood of
              two different files producing the same MD5 checksum is very
              small (about 1 in 2^64), and as such, the MD5 checksum of a
              file can be used to construct a reliable content identifier
              that is very likely to uniquely identify the file. The
              reverse is also true. If two files have the same MD5
              checksum, it is very likely that the files are
              identical. For purposes of MOF specification of the MD5
              property, the MD5 algorithm always generates a 32 character
              string. For example: The string abcdefghijklmnopqrstuvwxyz
              generates the string c3fcd3d76192e4007dfb496cca67e13b. See
              http://www. rsa.com/pub/rfc1321.txt for details on the
              implementation of the MD5 algorithm.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-oc107> NAME 'cim22FileSpecification'
        DESC 'A cim22FileSpecification identifies a file that is either to
              be on or off the system. The file is to be located in the
              directory identified by the DirectorySpecificationFile
              associations. When the invoke() method is used, it is
              expected that it will use the combination of information
              provided to check for the file existence. Therefore, any of
              the properties with a NULL value are not checked. So, if
              only the Name and the MD5 properties have values, they are
              the only ones consider by the invoke() method.'
        SUP cim22Check
        MAY (cimFileName $ cimCreateTimeStamp $ cimFileSize $
             cimCheckSum $ cimCRC1 $ cimCRC2 $ cimMD5Checksum)
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22FileSpecification.

```
    ( <oid-oc107> NAME 'cim22FileSpecificationContentRule'
      DESC 'The auxiliary classes that may be attached to
            cim22FileSpecification'
      AUX (cim22DirectorySpecificationFileAuxClass)
    )
```

3.13 cim22VersionCompatibilityCheck

This class specifies whether it is permissible to create the next state of a software element.

```
    ( <oid-at202> NAME 'cimAllowDownVersion'
      DESC 'The AllowDownVersion property indicates that this software
            element can transition to its next state even if a higher
            or later version of the software element already exists in
            the environment.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at203> NAME 'cimAllowMultipleVersions'
      DESC 'The AllowMultipleVersions option controls the ability to
            configure multiple versions of a product on a system.'
            SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-at204> NAME 'cimReinstall'
      DESC 'The Reinstall property indicates that this software element
            can transition to its next state even if a software element
            of the same version already exists in the environment.'
      SYNTAX boolean SINGLE-VALUE
    )

    ( <oid-oc108> NAME 'cim22VersionCompatibilityCheck'
      DESC 'The VersionCompatibilityCheck class specifies whether it is
            permissible to create the next state of a software
            element.'
      SUP cim22Check
      MAY (cimAllowDownVersion $ cimAllowMultipleVersions $
           cimReinstall)
    )
```

3.14 cim22Action

This class represents an operation that is part of a process to either create a SoftwareElement in its next state or to eliminate the SoftwareElement in the current state.

```
    ( <oid-at205> NAME 'cimActionID'
```

```
        DESC 'The ActionID property is a unique identifier assigned to a
              particular action for a software element.'
        SYNTAX string{256} SINGLE-VALUE
      )

      ( <oid-at206> NAME 'cimDirection'
        DESC 'The Direction property is used to indicate whether a
              particular Actionobject is part of a sequence of actions to
              transition the currentsoftware element to its next state,
              such as Install or to remove the current software element,
              such as Uninstall.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-oc109> NAME 'cim22Action'
        DESC 'A cim22Action is an operation that is part of a process to
              either create a SoftwareElement in its next state or to
              eliminate the SoftwareElement in the current state.'
        SUP top
        MUST (orderedCimModelPath)
        MAY (cimName $ cimVersion $ cimSoftwareElementState $
             cimSoftwareElementID $ cimTargetOperatingSystem $
             cimActionID $ cimDirection $ cimCaption $ cimDescription)
      )

      ( <oid-nf26> NAME 'cim22ActionNameForm'
        OC cim22Action
        MUST (orderedCimModelPath)
      )

      ( <sr26> NAME 'cim22ActionStructureRule'
        FORM cim22ActionNameForm
        SUP <sr27>
      )
```

The following content rule specifies the auxiliary classes that may be attached to cim22Action.

```
      ( <oid-oc109> NAME 'cim22ActionContentRule'
        DESC 'The auxiliary classes that may be attached to cim22Action'
        AUX (cim22ActionSequenceAuxClass)
      )
```

3.15 cim22DirectoryAction

This class manages directories to be managed. Creation of directories is handled by cim22CreateDirectoryAction and removal is handled by cim22RemoveDirectoryAction.

```
( <oid-at207> NAME 'cimDirectoryName'
  SYNTAX string SINGLE-VALUE
)

( <oid-oc110> NAME 'cim22DirectoryAction'
  DESC 'The DirectoryAction is an class that is used for
        directories to be managed. Creation of directories is
        handled by the CreateDirectoriesAction and removal is
        handled by the RemoveDirectory action.'
  SUP cim22Action
  MAY (cimDirectoryName)
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22DirectoryAction.

```
( <oid-oc110> NAME 'cim22DirectoryActionContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22DirectoryAction'
  AUX (cim22ToDirectoryActionAuxClass $
       cim22FromDirectoryActionAuxClass)
)
```

3.16 cim22CreateDirectoryAction

This class creates empty directories for cim22SoftwareElement objects to be installed locally.

```
( <oid-oc111> NAME 'cim22CreateDirectoryAction'
  DESC 'The CreateDirectory action creates empty directories for
        SoftwareElements to be installed locally.'
  SUP cim22DirectoryAction
)
```

3.17 cim22RemoveDirectoryAction

This class removes directories for cim22SoftwareElement objects.

```
( <oid-at208> NAME 'cimMustBeEmpty'
  SYNTAX boolean SINGLE-VALUE
)

( <oid-oc112> NAME 'cim22RemoveDirectoryAction'
  DESC 'The RemoveDirectoryAction removes directories for
        SoftwareElements.'
  SUP cim22DirectoryAction
  MAY (cimMustBeEmpty)
)
```

3.18 cim22FileAction

This class allows an implementor to locate files that already exist on the users machine, and move or copy those files to a new location.

```
( <oid-oc113> NAME 'cim22FileAction'
  DESC 'The cim22FileAction allows the author to locate files that
        already exist on the users machine, and move or copy those
        files to a new location.'
  SUP cim22Action
)
```

The following content rule specifies the auxiliary classes that may be attached to cim22FileAction.

```
( <oid-oc113> NAME 'cim22FileActionContentRule'
  DESC 'The auxiliary classes that may be attached to
        cim22FileAction'
  AUX (cim22FromDirectorySpecificationAuxClass $
       cim22FromDirectoryActionAuxClass)
)
```

3.19 cim22CopyFileAction

This class specifies files that exist on a computer system, and to move or copy those files to a new location.

```
( <oid-at209> NAME 'cimSource'
  SYNTAX string SINGLE-VALUE
)

( <oid-at210> NAME 'cimDestination'
  SYNTAX string SINGLE-VALUE
)

( <oid-at211> NAME 'cimDeleteAfterCopy'
  SYNTAX boolean SINGLE-VALUE
)

( <oid-oc114> NAME 'cim22CopyFileAction'
  DESC 'The cim22CopyFileAction specifies files that exist on a
        computer system, and to move or copy those files to a new
        location. The to/from information for the copy is specified
        using either the ToDirectorySpecification/
        FromDirectorySpecification or the ToDirectoryAction/
        FromDirectoryAction associations. The first set is
        used when the source and/or the target are to exist before
        any actions are taken. The second set is used when the
```

```
            source and/or target are created as a part of a previous
            action. In the latter case, the action to create the
            directory must occur prior to the CopyFileAction object.'
      SUP cim22FileAction
      MAY (cimSource $ cimDestination $ cimDeleteAfterCopy)
   )
```

The following content rule specifies the auxiliary classes that may be attached to cim22CopyFileAction.

```
   ( <oid-oc114> NAME 'cim22CopyFileActionContentRule'
     DESC 'The auxiliary classes that may be attached to
           cim22CopyFileAction'
     AUX (cim22ToDirectorySpecificationAuxClass $
          cim22ToDirectoryActionAuxClass)
   )
```

3.20 cim22RemoveFileAction

This class uninstalls files, specified in the file attribute.

```
   ( <oid-at212> NAME 'cimFile'
     SYNTAX string SINGLE-VALUE
   )

   ( <oid-oc115> NAME 'cim22RemoveFileAction'
     DESC 'The RemoveFileAction uninstalls files.'
     SUP cim22FileAction
     MAY (cimFile)
   )
```

3.21 cim22RebootAction

This class causes a reboot of the system where the software element is installed.

```
   ( <oid-oc116> NAME 'cim22RebootAction'
     DESC 'The RebootAction Causes a reboot of the system where the
           SoftwareElement is installed.'
     SUP cim22Action
   )
```

3.22 cim22ExecuteProgram

This class causes files to be executed on the system where the cim22SoftwareElement object is installed.

```
   ( <oid-at213> NAME 'cimProgramPath'
```

```
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-at214> NAME 'cimCommandLine'
        DESC 'A string that is invocable on a system command line.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-oc117> NAME 'cim22ExecuteProgram'
        DESC 'The ExecuteProgram causes files to be executed on the
              system where the SoftwareElement is installed.'
        SUP cim22Action
        MAY (cimProgramPath $ cimCommandLine)
      )
```

3.23 cim22SettingCheck

This class specifies information needed to check a particular setting file for a specific entry that contains a value that is equal to, or contains, the value specified. All compares are case insensitive.

```
      ( <oid-at215> NAME 'cimSectionKey'
        DESC 'Key of section containing the settings to be checked.'
        SYNTAX string{256} SINGLE-VALUE
      )

      ( <oid-at216> NAME 'cimEntryName'
        DESC 'Name of the Entry to be checked'
        SYNTAX string{256} SINGLE-VALUE
      )

      ( <oid-at217> NAME 'cimEntryValue'
        DESC 'Value to be checked that is associated with the named
              entry.'
        SYNTAX string SINGLE-VALUE
      )

      ( <oid-at218> NAME 'cimCheckType'
        DESC 'This specifies the way the setting value should be compared.'
        SYNTAX integer SINGLE-VALUE
      )

      ( <oid-oc118> NAME 'cim22SettingCheck'
        DESC 'This class specifies information needed to check a
              particular setting file for a specific entry that contains
              a value that is equal to, or contains, the value
              specified. All compares are assumed to be case insensitive.'
        SUP cim22Check
```

```
        MAY (cimSectionKey $ cimEntryName $ cimEntryValue $
             cimCheckType $ cimFileName)
      )
```

3.24 cim22ModifySettingAction

This class specifies the information to be used to change a particular setting file for a specific entry with a specific value. All additions are case sensitive while removes are case insensitive.

```
   ( <oid-at219> NAME 'cimActionType'
     DESC 'Type of action to be performed on the specified setting
           entry. Create - Creates the specified entry. Delete -
           Deletes the specified entry. Append - Append to the end of
           the specified entry. Remove - Remove the value from the
           specified entry.'
     SYNTAX integer SINGLE-VALUE
   )

   ( <oid-oc119> NAME 'cim22ModifySettingAction'
     DESC 'This class specifies the information to be used to modify a
           particular setting file for a specific entry with a
           specific value. The value specified is created as a new
           entry or appends to, replaces, removes from, or deletes the
           specified entry. All additions are assumed to be case
           sensitive. Removes are assumed to be case insensitive.'
     SUP cim22Action
     MAY (cimSectionKey $ cimEntryName $ cimEntryValue $ cimFileName $
          cimActionType)
   )
```

3.25 cim22DirectorySpecificationFileAuxClass

This class identifies the directory that contains the file being specified by referencing the cim22DirectorySpecification class.

```
   ( <oid-at220> NAME 'cimDirectorySpecificationRef'
     SYNTAX DN
   )

   ( <oid-at221> NAME 'cimFileSpecificationRef'
     SYNTAX DN
   )

   ( <oid-oc120> NAME 'cim22DirectorySpecificationFileAuxClass'
     DESC 'The cim22DirectorySpecificationFile association identifies
           the directorythat contains the file being specified by
           referencing the cim22DirectorySpecification class.'
```

```
        SUP top AUXILIARY
        MAY (cimDirectorySpecificationRef $ cimFileSpecificationRef)
      )
```

3.26 cim22ActionSequenceAuxClass

cim22ActionSequenceAuxClass defines a series of operations that either transitions the software element, referenced by cim22SoftwareElementActionsAuxClass, to its next state or removes the software element from its current environment. The action classes participating in this association must have the same value for the direction property since they are either part of a sequence to transition a software element into its next state or to uninstall a software element. The next-state actions and uninstall actions associated with a particular software element must be a continuous sequence. Since the action sequence is an association the loops on the action class with roles for the 'prior' action and 'next' action in a sequence, the need for a continuous sequence implies: (1)Within the set of next-state or uninstall actions, there is one and only one action that does not have an instance of the ActionSequence association referencing it in the 'next' role. This is the first action in the sequence. (2) Within the set of next-state or uninstall actions, there is one and only one action that does not have an instance of the ActionSequence association referencing it in the 'prior' role. This is the last action in the sequence. (3) All other actions within the set of next-state and uninstall actions must participate in two instances of the ActionSequence association, one in a prior role and one in the next role.

```
   ( <oid-at222> NAME 'cimNextRef'
     SYNTAX DN
   )

   ( <oid-at223> NAME 'cimPriorRef'
     SYNTAX DN
   )

   ( <oid-oc121> NAME 'cim22ActionSequenceAuxClass'
     DESC 'The cim22ActionSequence association is used to define a
           series of operations that either transitions the software
           element, referenced by the cim22SoftwareElementActions
           association, to its next state or removes the software
           element from its current environment. The Action classes
           participating in this association must have the same value
           for the Direction property since they are either part of a
           sequence to transition a software element into its next
           state or to uninstall a software element. The next-state
```

actions and uninstall actions associated with a particular

```
            software element must be a continuous sequence. Since the
            ActionSequence is an association the loops on the Action
            class with roles for the "prior" action and "next" action
            in a sequence, the need for a continuous sequence implies:
            (1)Within the set of next-state or uninstall actions, there
            is one and only one action that does not have an instance
            of the ActionSequence association referencing it in the
            "next" role. This is the first action in the sequence. (2)
            Within the set of next-state or uninstall actions, there is
            one and only one action that does not have an instance of
            the ActionSequence association referencing it in the
            "prior" role. This is the last action in the sequence. (3)
            All other actions within the set of next-state and
            uninstall actions must participate in two instances of the
            ActionSequence association, one in a prior role and one in
            the next role. Both attributes point to cim22Action objects.'
      SUP top AUXILIARY
      MAY (cimNextRef $ cimPriorRef)
    )
```

3.27 cim22SoftwareFeatureSoftwareElementsAuxClass

This auxiliary class identifies the software elements that make up a particular software feature.

```
    ( <oid-oc122> NAME 'cim22SoftwareFeatureSoftwareElementsAuxClass'
      DESC 'The SoftwareFeatureSoftwareElements associations identifies
            the software elements that make up a particular software
            feature. Attribute cimGroupComponentRef points to
            cim22SoftwareFeature and attribute cimPartComponentRef points
            to cim22SoftwareElement.'
      SUP cim22ComponentAuxClass
      MAY (cimGroupComponentRef $ cimPartComponentRef)
    )
```

3.28 cim22ToDirectorySpecificationAuxClass

This auxiliary class identifies the target directory for the file action and assumes that the target directory already existed. This association cannot exist with a cim22ToDirectoryActionAuxClass since a file action can only involve a single target directory.

```
    ( <oid-at224> NAME 'cimDestinationDirectoryRef'
      SYNTAX DN
    )

    ( <oid-at225> NAME 'cimFileNameRef'
```

SYNTAX DN

```
)

( <oid-oc123> NAME 'cim22ToDirectorySpecificationAuxClass'
  DESC 'The ToDirectorySpecification association identifies the
        target directory for the file action. When this association
        is used, the assumption is that the target directory
        already existed. This association cannot exist with a
        ToDirectoryAction association since a file action can only
        involve a single target directory. Attribute
        cimDestinationDirectoryRef points to
        cim22DirectorySpecification and attribute cimFileNameRef
        points to cim22CopyFileAction.'
  SUP top AUXILIARY
  MAY (cimDestinationDirectoryRef $ cimFileNameRef)
)
```

3.29 cim22FromDirectorySpecificationAuxClass

This auxiliary class identifies the source directory for the file action and assumed that the source directory already existed. This association cannot exist with a cim22FromDirectoryActionAuxClass since a file action can only involve single source directory.

```
( <oid-at226> NAME 'cimSourceDirectoryRef'
  SYNTAX DN
)

( <oid-oc124> NAME 'cim22FromDirectorySpecificationAuxClass'
  DESC 'The FromDirectorySpecification association identifies the
        source directory for the file action. When this association
        is used, the assumption is that the source directory
        already existed. This association cannot exist with a
        FromDirectoryAction association since a file action can
        only involve single source directory. Attribute
        cimSourceDirectoryRef points to cim22DirectorySpecification
        and cimFileNameRef points to cim22FileAction.'
  SUP top AUXILIARY
  MAY (cimSourceDirectoryRef $ cimFileNameRef)
)
```

3.30 cim22ToDirectoryActionAuxClass

This auxiliary class identifies the target directory for the file action and assumes that a previous action creates the target directory This association cannot exist with a cim22ToDirectorySpecificationAuxClass since a file action can only involve a single target directory.

```
( <oid-oc125> NAME 'cim22ToDirectoryActionAuxClass'
  DESC 'The ToDirectoryAction association identifies the target
        directory for the file action. When this association is
        used, the assumption is that the target directory was
        created by a previous action. This association cannot exist
        with a ToDirectorySpecification association since a file
        action can only involve a single target
        directory. Attribute cimDestinationDirectoryRef points to
        cim22DirectoryAction and attribute cimFileNameRef
        points to cim22CopyFileAction.'
  SUP top AUXILIARY
  MAY (cimDestinationDirectoryRef $ cimFileNameRef)
)
```

3.31 cim22FromDirectoryActionAuxClass

This auxiliary class identifies the source directory for the file action and assumes that a previous action creates the source directory. It cannot exist with a cim22FromDirectorySpecificationAuxClass since a file action can only involve a single source directory.

```
( <oid-oc126> NAME 'cim22FromDirectoryActionAuxClass'
  DESC 'The FromDirectoryAction association identifies the source
        directory for the file action. When this association is
        used, the assumption is that the source directory was
        created by a previous action. This association cannot exist
        with a FromDirectorySpecification association since a file
        action can only involve a single source
        directory. Attribute cimSourceDirectoryRef points to
        cim22DirectoryAction and cimFileNameRef points to
        cim22FileAction.'
  SUP top AUXILIARY
  MAY (cimSourceDirectoryRef $ cimFileNameRef)
)
```

3.32 cim22SoftwareFeatureServiceImplementationAuxClass

This auxiliary class relates a service and how it is implemented in software and has a many-to-many cardinality. A service may be provided by more than one software feature, operating in conjunction and, any software feature may provide more than one service. If different implementations of a service exist, each of these implementations are individual instantiations of the cim22Service object. These individual instantiations would then have associations to the unique implementations.

```
      ( <oid-oc127> NAME
```

```
        'cim22SoftwareFeatureServiceImplementationAuxClass'
        DESC 'An association between a Service and how it is implemented
              in software. The cardinality of this association is
              many-to-many. A Service may be provided by more than one
              SoftwareFeature, operating in conjunction. And, any
              software Feature may provide more than one Service. When
              multiple SoftwareFeatures are associated with a single
              Service, it is assumed that these elements operate in
              conjunction to provide the Service. If different
              implementations of a Service exist, each of these
              implementations would result in individual instantiations
              of the Service object. These individual instantiations
              would then have associations to the unique
              implementations. Attribute cimAntecendentRef points to
              cim22SoftwareFeature and attribute cimDependentRef points to
              cim22Service.'
        SUP cim22DependencyAuxClass
        MAY (cimAntecedentRef $ cimDependentRef)
      )
```

3.33 cim22SoftwareFeatureSAPImplementationAuxClass

This auxiliary class represents the relationship between a service access point and how it is implemented in software. The cardinality of this association is many-to-many. A SAP may be provided by more than one SoftwareFeature, operating in conjunction. And, any SoftwareFeature may provide more than one ServiceAccessPoint. a When many SoftwareFeatures are associated with single SAP, it is assumed that these elements operate in conjunction to provide the AccessPoint. If different implementations of a SAP exist, each of these implementations would result in individual instantiations of the ServiceAccessPoint object. These individual instantiations would then have associations to the unique implementations.

```
   ( <oid-oc128> NAME 'cim22SoftwareFeatureSAPImplementationAuxClass'
     DESC 'An association between a ServiceAccessPoint and how it is
           implemented in software. The cardinality of this association
           is many-to-many. A SAP may be provided by more than one
           SoftwareFeature, operating in conjunction. And, any
           SoftwareFeature may provide more than one
           ServiceAccessPoint. a When many SoftwareFeatures are
           associated with single SAP, it is assumed that these
           elements operate in conjunction to provide the
           AccessPoint. If different implementations of a SAP exist,
           each of these implementations would result in individual
           instantiations of the ServiceAccessPoint object. These
           individual instantiations would then have associations to
```

the unique implementations. Attribute cimAntecendentRef

```
            points to cim22SoftwareFeature and attribute cimDependentRef
            points to cim22ServiceAccessPoint.'
      SUP cim22DependencyAuxClass
      MAY (cimAntecedentRef $ cimDependentRef)
    )
```

3.34 cim22ApplicationSystemSoftwareFeatureAuxClass

This auxiliary class identifies the software features, which may be part of different products, that make up a particular application system.

```
    ( <oid-oc129> NAME 'cim22ApplicationSystemSoftwareFeatureAuxClass'
      DESC 'The ApplicationSystemSoftwareFeature associations
            identifies the software features that make up a particular
            application system. The software features can be part of
            different products. Attribute cimGroupComponentRef points
            to cim22ApplicationSystem and attribute cimPartComponentRef
            points to cim22SoftwareFeature.'
      SUP cim22SystemComponentAuxClass
      MAY (cimGroupComponentRef $ cimPartComponentRef)
    )
```

3.35 cim22InstalledSoftwareElementAuxClass

This auxiliary class allows one to identify the computer system a particular software element is installed on.

```
    ( <oid-at227> NAME 'cimSoftwareRef'
      DESC 'References the software element that is installed.'
      SYNTAX DN
    )

    ( <oid-at228> NAME 'cimSystemRef'
      DESC 'References the computer system hosting a particular
            software element.'
      SYNTAX DN
    )

    ( <oid-oc130> NAME 'cim22InstalledSoftwareElementAuxClass'
      DESC 'The InstalledSoftwareElement association allows one to to
            identify the Computer System a particular Software element
            is installed on. Attribute cimSoftwareRef points to
            cim22SoftwareElement and attribute cimSystemRef points to
            cim22ComputerSystem.'
      SUP top AUXILIARY
      MAY (cimSoftwareRef $ cimSystemRef)
    )
```

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, "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 Application 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.

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