

## Windows 2000 — Active Directory Objects

By Edward Voorhees – Contract Technical Services

### Active Directory Objects

Active Directory *objects* are the entities that make up a network. An object is a distinct, named set of attributes that represents something concrete, such as a user, a printer, or an application. When you create an Active Directory object, Active Directory generates values for some of the object's attributes, others you provide. For example, when you create a user object, Active Directory assigns the globally unique identifier (GUID), and you provide values for such attributes as the user's given name, surname, the logon identifier, and so on.

### Active Directory Schema

The *schema* is a description of the *object classes* (the various types of objects) and the *attributes* for those object classes. For each class of object, the schema defines:

- the attributes that object class must have,
- the additional attributes it may have, and
- the object class that can be its parent.

Every Active Directory object is an instance of an object class. Each attribute is defined only once and can be used in multiple classes. For example, the Description attribute is defined once but is used in many different classes.

The schema is stored in Active Directory. Schema definitions are themselves also stored as objects—Class Schema objects and Attribute Schema objects. This allows Active Directory to manage class and attribute objects in the same way that it manages other directory objects.

Applications that create or modify Active Directory objects use the schema to determine what attributes the object must or might have, and what those attributes can look like in terms of data structures and syntax constraints.

Objects are either container objects or leaf objects (also called non-container objects). A container object stores other objects and a leaf object does not. For example, a folder is a container object for files, which are leaf objects.

Each class of objects in the Active Directory schema has attributes that ensure:

- Unique identification of each object in a directory data store.
- For security principals (users, computers, or groups), compatibility with security identifiers (SIDs) used in the Windows NT 4.0 operating system and earlier.
- Compatibility with LDAP standards for directory object names.

## Schema Object Names

As stated earlier, classes and attributes are both schema objects. Any schema object can be referenced by each of the following types of names:

- LDAP display name.** The LDAP display name is globally unique for each schema object. The LDAP display name consists of one or more words combined, using initial caps for words after the first word. For example, `mailAddress` and `machinePasswordChangeInterval` are the LDAP display names for two schema attributes. Active Directory Schema and other Windows 2000 administrative tools display the LDAP display name of objects, and programmers and administrators use this name to reference the object programmatically.
- Common name.** The common name for schema objects is also globally unique. You specify the common name when creating a new object class or attribute in the schema—it is the relative distinguished name (RDN) of the object in the schema that represents the object class. For example, the common names of the two attributes mentioned in the preceding paragraph are `SMTP-Mail-Address` and `Machine-Password-Change-Interval`.
- Object identifier (OID).** A schema object's identifier is a number issued by an issuing authority such as the International Organization for Standardization (ISO) and the American National Standards Institute (ANSI). For example, the OID for the `SMTP-Mail-Address` attribute is `1.2.840.113556.1.4.786`. OIDs are guaranteed to be unique across all networks worldwide. Once you obtain a root OID from an issuing authority, you can use it to allocate additional OIDs. OIDs form a hierarchy.

For example, Microsoft has been issued the root OID of `1.2.840.113556`. Microsoft manages further branches from this root internally. One of the branches is used to allocate OIDs for Active Directory schema classes, and another for attributes. To continue the example, the OID in Active Directory is `1.2.840.113556.1.5.4`, which identifies the Builtin Domain class and can be parsed as shown in Table 1.

Object ID Number	Identifies
1	ISO ("root" authority) issued 1.2 to ANSI, then...
2	ANSI issued 1.2.840 to USA, then...
840	USA issued 1.2.840.113556 to Microsoft, then...
113556	Microsoft internally manages several object identifier branches under 1.2.840.113556 that include...
1	a branch called Active Directory that includes...
5	a branch called classes that includes...
4	a branch called Builtin Domain

**Table 1 - Object identifier**

## Characteristics of Object Classes

Each Active Directory™ object class is defined by a **classSchema** object in the schema container. The attributes of a **classSchema** object specify the characteristics of the class, such as the following:

- *Class identifiers.* Classes have several identifiers, of which the most interesting from a programming perspective are the **ldapDisplayName**, which is used by LDAP clients to identify the class in search filters, and the **schemaIDGUID**, which is used in security descriptors to control access to the class.
- *Possible attributes.* An object class definition includes lists of the mandatory and optional attributes that can be set on an instance of the class.
- *Possible parents.* Every object instance (except the root of the directory hierarchy) has exactly one parent. An object class definition includes lists of possible parents, that is, of the object classes that can contain an instance of the class.
- *Superclasses and auxiliary classes.* Every object class (except **top**) is derived from another class. A class inherits possible attributes and possible parents from the classes above it in the class hierarchy. A class can also have any number of auxiliary classes from which it inherits lists of possible attributes. See [Class Inheritance in the Active Directory Schema](#).

The following list shows the **IDAPDisplayName** and description of the key attributes of a **classSchema** object. For a complete list of the mandatory and optional attributes of a **classSchema** object, see **classSchema**.

### **cn (Common-Name)**

Every object in Active Directory has a naming attribute from which its Relative Distinguished Name (RDN) is formed. The naming attribute for **classSchema** objects is **cn** (Common-Name). The value assigned to **cn** is the value that the object class will have as its RDN. For example, the **cn** of the **organizationalUnit** object class is Organizational-Unit, which would appear in a distinguished name as CN=Organizational-Unit. The **cn** must be unique in the schema container.

### **IDAPDisplayName**

The name used by LDAP clients, such as the ADSI LDAP provider, to refer to the class, for example to specify the class in a search filter. A class's **IDAPDisplayName** must be unique in the schema container, which means it must be unique across all **classSchema** and **attributeSchema** objects. For information on composing a **cn** and an **IDAPDisplayName** for a new class, see [Naming Attributes and Classes](#).

**schemaIDGUID**

A GUID stored as an octet string. This GUID uniquely identifies the class. This GUID can be used in access control entries to control access to objects of this class. See [Setting Permissions on Child Object Operations](#).

On creation of the **classSchema** object, Active Directory generates this value if it is not specified. If you are creating a new class, it is recommended that you generate your own GUID for each class so that all installations of your extension will use the same **schemaIDGUID** to refer to the class.

**adminDisplayName**

A display name of the class for use in administrative tools. If **adminDisplayName** is not specified when a class is created, the system uses the Common-Name value as the display name.

This display name is used only if a mapping does not exist in the **classDisplayName** property of the display specifier for the class.

**governsID**

The object identifier (OID) of the class. This value must be unique among the **governsIDs** of all **classSchema** objects and the **attributeIDs** of all **attributeSchema** objects. See [Object Identifiers \(OIDs\)](#).

**rDnAttId**

Identifies the naming attribute, which is the attribute that provides the Relative Distinguished Name (RDN) for this class — if different than the default (**cn**). Use of a naming attribute other than **cn** is discouraged. Naming attributes should be drawn from the well-known set (**OU, CN, O, L and DC**) that is understood by all LDAP version 3 clients. See [Object Names and Identities](#) and [Syntaxes for Active Directory Attributes](#).

A naming attribute must have the Directory String syntax. See [Syntaxes for Active Directory Attributes](#).

**mustContain, systemMustContain**

A pair of multi-valued properties that specify the attributes that **MUST** be present on instances of this class. These are mandatory attributes that must be present during creation and cannot be cleared after creation. After creation of the class, these properties cannot be changed.

The full set of mandatory attributes for a class is the union of the **systemMustContain** and **mustContain** values on this class and all inherited classes.

**mayContain, systemMayContain**

A pair of multi-valued properties that specify the attributes that **MAY** be present on instances of this class. These are optional attributes that are not mandatory and, therefore, may or may not be present on an instance of this class. You can add or remove **mayContain** values from an existing category 1 or category 2 **classSchema**

object. Before removing a **mayContain** value from a **classSchema** object, you should search for instances of the object class and clear any values for the attribute that you are removing. After creation of the class, the **systemMayContain** property cannot be changed

The full set of optional attributes for a class is the union of the **systemMayContain** and **mayContain** values on this class and all inherited classes.

### **possSuperiors, systemPossSuperiors**

A pair of multi-valued properties that specify the structural classes that can be legal parents of instances of this class. The full set of possible superiors is the union of the **systemPossSuperiors** and **possSuperiors** values on this class and any inherited structural or abstract classes. Note that **systemPossSuperiors** and **possSuperiors** values are not inherited from auxiliary classes.

You can add or remove **possSuperiors** values from an existing category 1 or category 2 **classSchema** object. After creation of the class, the **systemPossSuperiors** property cannot be changed.

### ***objectClassCategory***

An integer value that specifies the category of the class, which can be one of the following:

- Structural, meaning it can be instantiated in the directory.
- Abstract, meaning the class provides a basic definition of a class that can be used to form structural classes.
- Auxiliary. meaning a class that can be used to extend the definition of a class that inherits from it but cannot be used to form a class by itself.

### **subClassOf**

An object identifier (OID) for the immediate superclass of this class, that is, the class from which this class is derived.

For structural classes, **subClassOf** can be a structural or abstract class.

For abstract classes, **subClassOf** can only be an abstract class.

For auxiliary classes, **subClassOf** can be an abstract or auxiliary class.

If you are defining a new class, you must ensure that the **subClassOf** class exists or will exist when the new class is written to the directory. If class does not exist, the **classSchema** object will fail to be added to the directory.

### *auxiliaryClass, systemAuxiliaryClass*

A pair of multi-valued properties that specify the auxiliary classes that this class inherits from. The full set of auxiliary classes is the union of the **systemAuxiliaryClass** and **auxiliaryClass** values on this class and all inherited classes.

For an existing **classSchema** object, values can be added to the **auxiliaryClass** property but not removed. After creation of the class, the **systemAuxiliaryClass** property cannot be changed.

### **defaultObjectCategory**

The distinguished name of this object class or one of its superclasses. When an instance of this object class is created, the system sets the **objectCategory** property of the new instance to the value specified in the **defaultObjectCategory** property of its object class. The **objectCategory** property is an indexed property used to make object class searches fast and efficient.

If **defaultObjectCategory** is not specified when a class is created, the system sets it to the DN of the **classSchema** object for this class. If this object will be frequently queried by the value of a superclass rather than the object's own class, you can set **defaultObjectCategory** to the DN of the superclass. For example, if you are subclassing a predefined (category 1) class, best practice is to set **defaultObjectCategory** to the same value as the superclass. This allows the standard UI to "find" your subclass.

### **defaultHidingValue**

A Boolean value that specifies the default setting of the **showInAdvancedViewOnly** property of new instances of this class. Many directory objects are not interesting to end users. To keep these objects from cluttering the UI, every object has a Boolean attribute called **showInAdvancedViewOnly**.

If **defaultHidingValue** is set to TRUE, new object instances are hidden in the Administrative snap-ins and the Windows shell. It also means that a menu item for the object class will not appear in the **New** context menu of the Administrative snap-ins — even if the appropriate creation wizard properties are set on the object class's **displaySpecifier** object.

If **defaultHidingValue** is set to FALSE, new instances of the object are displayed in the Administrative snap-ins and the Windows shell. Set this property to FALSE if you want to be able to see instances of the class in the administrative snap-ins and the shell *and* enable a creation wizard and its menu item in the **New** menu of the administrative snap-ins.

If the **defaultHidingValue** value is not set, the default is TRUE.

**systemFlags**

An integer value that contains flags that define additional properties of the class. The 0x10 bit identifies a category 1 class (a class that is part of the base schema that ships with the system). You cannot set this bit, which means that the bit is not set in category 2 classes (which are extensions to the schema).

**systemOnly**

A boolean value that specifies whether only Active Directory can modify the class. System-only classes can only be created or deleted by the Directory System Agent (DSA). System-only classes are those that the system depends on for normal operations.

**defaultSecurityDescriptor**

Specifies the default security descriptor for new objects of this class

**isDefunct**

A boolean value that indicates whether the class is defunct. See [Disabling Existing Classes and Attributes](#).

**description**

A text description of the class for use by administrative applications.

**objectClass**

Identifies the object class of which this object is an instance, which is the **classSchema** object class for all class definitions and the **attributeSchema** object class for all attribute definitions.