1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”

To learn the current status of any Internet-Draft, please check the “id-abstracts.txt” listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

The distribution of this memo is unlimited. It is filed as <draft-aboba-radius-02.txt>, and expires August 1, 1998. Please send comments to the authors.

2. Abstract

This document defines a schema for the Remote Access Dialin User Service (RADIUS). This schema makes it possible to integrate a RADIUS server with an LDAP-based directory service, making it possible for an organization to maintain a single store of user information. This consolidation is desirable since it results in a reduction in the administrative workload, and eliminates the need to synchronize across multiple user information stores.

3. Introduction

Today enterprises are looking to simplify the process of user administration. With the advent of HR and personnel management systems, information on a user is typically entered at the time of hiring, and is retained until termination. If an LDAP-based directory is also deployed, this necessitates synchronization with the personnel database in order to maintain consistency.

Should the enterprise then deploy NAS devices or layer 2 tunneling solutions, there may be a need to add a RADIUS server or if extended
security is required, a backend security server. Each of these may require their own user information store.

Operating multiple stores of user information is not appealing, since this may require rekeying of information or synchronization between multiple stores, resulting in increased administrative costs. Maintaining multiple stores also raises concerns about inconsistency and replication delays. In order to avoid these problems, it is desirable to consolidate stores of user information. One way this can be achieved is to make it possible for RADIUS servers and security add-ons to store their user information in an LDAP-based directory.

This document is one of three related specifications which describe how a RADIUS server may be integrated with an LDAP-based directory service. Reference [12] describes a schema designed for tracking sessions in progress. Such information can be useful for a variety of purposes including security incident response; simultaneous usage control; or monitoring of connection quality, login time, packet size or bandwidth usage. Since this data change frequently, dynamic attributes must be employed as described in [9]. Reference [13] describes how user credentials submitted during PPP authentication may be validated by the RADIUS server.

This document defines an LDAP schema for the Remote Access Dialin User Service (RADIUS). The RADIUS protocol, described in [6]-[8], supports authentication, authorization and accounting for dialup users. To date, RADIUS servers have stored user data in a variety of ways, including databases and flat files. A goal of this schema is to make it possible to add support for LDAP-based directory services to existing RADIUS server implementations. In order to permit this schema to be used with a wide range of directory service implementations, we avoid reliance on features that have not been widely implemented, such as multiple inheritance.

3.1. Objects and attributes

The RADIUS schema defined in this document requires support for several new classes: radiusProfileClass, radiusPolicyClass, radiusDictionaryClass, and eapDictionaryClass. The radiusProfileClass is used to store RADIUS attributes relevant to groups of users. The radiusPolicyClass is used to describe conditions under which a given profile may be applied. The radiusDictionaryClass is used to store the RADIUS Dictionary. This provides extensibility and allows RADIUS profile objects to be self describing. The eapDictionaryClass is used to store a list mapping EAP Types to names.

The attributes in radiusProfileClass fall into two categories: attributes present in the Access-Reply, and attributes representing access constraints. An access constraint is a set of conditions that must be satisfied in order for access to be granted. These are expressed in the form of matching rules involving attributes present in the Access-Reply, as well as other attributes such as the time of day. For example, a matching rule involving the calledStationId and
time of day can be created in order to limit access to those calling a given phone number during specified hours.

Attributes present in the Access-Reply are stored in the directory so that the RADIUS server can retrieve them and include them in the Access-Reply. Access constraints are stored in the directory so that the RADIUS server can test the incoming Access-Request to determine whether to proceed with authentication, or immediately send an Access-Reject. Note that only static attributes present in Access-Reply need be stored in the directory; attributes which are computed on the fly can be recreated as needed.

The attributes in radiusPolicyClass represent conditions which must hold for the profile indicated in radiusProfilePointer to be applied. As with access constraints, these conditions may involve matching rules applied to attributes in the Access-Request, as well as conditions involving time of day, Nas-Port-Type, or group memberships.

For example, it may be desirable to give users different Session-Time or Port-Limit attributes depending on the time of day, or group memberships. This can be accomplished by creating policy expressions and profiles for each time of day/group membership combination. Similarly, it may be desirable to require that analog and ISDN callers do callback or call from a particular callingStationId, while this may not make sense for users connecting over a VPN. This can be accomplished by creating a policy expression that returns different profiles, depending on nasPortType.

3.2. Administrative model

The schema defined in this document includes user object attributes, as well as profile and policy objects.

User object attributes are used in situations where it may be desirable to override behavior supplied in a profile, or where it is desired that individual users be given an unique value for an attribute. For example, where static addresses are assigned, each user will typically have a different IP address; similarly, where callback is used, callbackNumber will typically differ between users.

In the early versions of this document, it was envisaged that all attributes would be contained within the user object. This is wasteful because it is likely that groups of users will tend to have the same parameter values. Thus a schema based solely on user-object results in unnecessary replication, and also makes it difficult to change attributes for all members of a group.

To reduce the replication problem, enable more effective caching, and ease the administrative burden, profiles were added to the schema. Profiles support definition of parameter sets which apply to a group of users in a particular situation. Since it is expected that profiles
will apply to large group of users, they can be effectively cached. Reference [14] describes how object caching can be supported within LDAP-based directory services.

In an earlier version of this document, profiles could be related to users via the radiusProfilePointer attribute included in the user object. While this method of mapping users to profiles is still supported in this revision, this approach does not scale well, since it requires administrators to modify the directory entries for each user, in order to add the required radiusProfilePointer. Network administrators typically manage the authorization process via group assignments, and therefore will typically desire to fit profiles within the existing administrative model. In particular, it is highly desirable to allow an administrator to change the profile values applying to a group without having to edit the user objects for each member of the group.

Several methods for binding a profile to a group suggest themselves. Within this schema, the mapping is achieved via a policy object which may include group membership among the conditions evaluated in assignment of a profile. It should be noted that policy objects are not the only way to bind profiles to groups, nor are they necessarily the most efficient. For example, it is also possible to handle profile/group binding via a table, or even by encoding policy restrictions on a user certificate. The later may prove popular in the long term, given that today many firms already encode privileges relating to time of day and organizational function on employee badges.

The profile/group binding method chosen in this document was selected primarily since it proved to be a degenerate case of the general conditional profile problem. In this schema, we support the conditional application of profiles, with the policy object expressing the conditions that must be satisfied for a profile to be executed. Thus, profile/group binding can be expressed as a condition (group membership) resulting in assignment of a profile (the profile for that group).

### 3.2.1. User object attributes

This schema proposes addition of attributes to the user object. As noted earlier, to enhance scalability, it is recommended that user object attributes only be used in cases where profile override is necessary, or assignment of per-user attributes is required. Override can in principle be required for any attribute that may be included in the Access-Reply, and so these attributes are among those that are added to the user object. Examples of attributes that may be assigned on a per-user basis include radiusFramedIPAddress, radiusCallbackNumber and radiusFramedRoute.

Since many RADIUS parameters are expected to be identical for a group of users, typically the user object will contain a small set of Radius attributes. No user object attributes may be present if profiles are being applied conditionally and no per-user values are required.
If it is desired that a profile be unconditionally executed, then this can be achieved either by creating a policy object with a radiusProfilePointer attribute but no npConstraint attribute, or by adding radiusPolicyPointer (a distinguished name pointing to a RADIUS Profile Object) as a user object attribute.

3.2.2. Profiles

Profile attributes fall into two major categories. One category of attributes are static attributes that may be returned in an Access-Reply. These attributes use a prefix of ‘radius’ and are included within the profile so that the RADIUS server may copy the values into the Access-Reply.

Another category of attributes are those which represent conditions that must be satisfied for an Access-Accept to be sent. These attributes use a prefix of ‘np’, which stands for Network Policy. These attributes include npIPPoolName, npSessionsAllowed, npEAPType, npConstraint, and npAuthenticationType. npSessionsAllowed is used to limit the number of simultaneous sessions; npAuthenticationType indicates the acceptable authentication types (PAP, CHAP, MS-CHAP, EAP); npEAPType indicates the EAP-Type to be used to authenticate the user if EAP is negotiated as an authentication type; npIPPoolName indicates the name of the IP address pool that should be used in assigning the user’s IP address. npConstraint is a string attribute used to express constraints based on time of day, or attributes present in the Access-Request, such as NAS-Port-Type or NAS-Identifier.

Within this document, we allow profiles to include pointers to other profiles, so that profiles may form a linked list. This allows a hierarchy of profiles to be provided. More specific attributes override more general ones.

3.2.3. Example

All BIGCO employees are required to use token card authentication, and thus in the company profile the radiusAuthenticationType attribute is set to only allow EAP, and the radiusEAPType attribute is set for BIGCO’s token card type. BIGCO also sets up a marketing profile providing a radiusSessionTimeout value of 30 minutes, a radiusPortLimit of one, and radiusFramedIpAddress set to indicate dynamic address allocation. However, Fred requires a static IP address, and thus his user object will contain a radiusFramedIpAddress attribute.

Since BIGCO profiles are unconditionally applied, a policy object with a condition of (group == marketing) is used to assign a profile to marketing personnel. Another policy object of lower priority is used with no npConstraint attribute in order to assign a default profile.
3.3. Policy support

The schema described in this document provides for the conditional application of a profile to a user via policy objects. Policy objects make it possible to have profile A apply to a user in one set of circumstances, and profile B apply in another set of circumstances. They also enable binding of profiles to groups.

Each policy object corresponds to an IF/THEN statement; multiple policy objects may be required to express complex policies. Attributes in the policy object include npConstraint, a string attribute which expresses the conditions under which a profile will be applied; npSequence, an integer attribute which describes the order in which the policy object will be evaluated; and radiusProfilePointer, a Distinguished Name pointing to the RADIUS profile that will be applied if the conditions hold. The matching rule stored in npConstraint is an expression which may reference other attribute values and include pattern matching and other operations, such as equality tests. Policy objects without an npConstraint attribute can be used to indicate unconditional execution of a profile.

Although a simple Policy Object is presented in this schema, more complex versions are possible. For example, a wider variety of operators and pattern matches might be supported within npConstraint.

3.3.1. Example

Let us assume that BIGCO wishes to offer dialin access to their domestic sales force, as well as VPN access to contractors and to individuals from the finance group travelling overseas. In order to consistently manage and account for the use of their NAS devices and Layer 2 tunnel servers (PPTP/L2F/L2TP), BIGCO has chosen to adopt the RADIUS protocol. However, given the large number of employees and contractors that need to be managed, BIGCO desires a RADIUS solution integrated with their existing LDAP-based directory service and group structure. This will allow the network administrator to edit the user’s RADIUS attributes with the same user-interface as they use to edit other user attributes, profiles, and policies, and will eliminate the need to maintain multiple stores of user information.

As part of this service offering, BIGCO may wish to implement a number of policies. For example, in order to make sure that high speed dialin access is available to the sales force when they need it, BIGCO may wish to restrict use of the ISDN ports to sales personnel only during the hours of 9-5, and permit the use of multilink. Since contractors are only to be given access to selected subnets, BIGCO may wish to apply a filter to their traffic. Since individuals in the finance group often access highly confidential information over the VPN, BIGCO may wish to require that these users authenticate via a smartcard, and use only 128-bit encryption so as to provide for extended security. For security reasons, BIGCO may wish to restrict contractors and finance users to a single login at a time.
In certain cases, BIGCO may also wish to implement policies that depend on the type of port that the user is connecting to. For example, if the user is connecting via dialup, then it may be appropriate to include tunnel attributes within the Access-Accept, so as to set up a tunnel for the user. However, if the user is already connected via a tunnel, this would not be necessary. Similarly, if BIGCO only has a limited number of ISDN ports available, it may be desirable to set a shorter Session-Timeout or Idle-Timeout on these ports, or to set Port-Limit to one so as to not allow multi-link. The schema defined in this document permits enforcement of these and many other policies.

3.4. Caching

Reference [14] describes a simple caching scheme for LDAP-based directories. A new operational attribute, ttl, is defined which specifies the maximum time an object may remain in the cache. Such a caching scheme is particularly beneficial for the schema presented in this document, since it is expected that profiles and policies will apply to large numbers of users. The first time the RADIUS server encounters a pointer to a given profile or policy, the profile or policy will be retrieved from the directory and cached. Subsequently, the profile or policy may be retrieved from the cache, speeding the retrieval process. As a result, it is to be expected that caching should result in a substantial performance gain. As noted in [14], the ttl attribute denotes the number of seconds that an entry may remain in the cache before becoming stale. A value of 0 implies that the object must not be cached.

3.5. Extensibility

Today vendors distinguish their RADIUS servers by a variety of means, including the range of supported attributes (standard and vendor-specific), and the breadth of policies that may be represented. As a result, while it is desirable to provide a common base set of classes and attributes which all RADIUS schemas will share, RADIUS server capabilities differ substantially from implementation to implementation, and a successful RADIUS schema definition must support this differentiation.

The schema described in this document provides support for most of the attributes defined in [6]-[8], as well as including support for the RADIUS Dictionary and vendor-specific attributes, as well as conditional application of profiles. Within this framework, vendor differentiation can be achieved via two methods: adding attributes to the base RADIUS profile and policy classes, or creating subclasses inheriting from the base classes. Adding attributes to the base class is recommended in cases where the new attributes to be added do not conflict with those described in this document or in [6]-[8].

Where conflicts do not arise, new attributes, including vendor-specific attributes, may be added to the RADIUS dictionary, which allows RADIUS Profile objects to be self-describing. The goal is to allow
attributes to be added without having to require an update to the RADIUS server code. Note however that a conventional RADIUS dictionary is only designed to describe attributes that are sent on the wire, while the RADIUS Dictionary object defined in this schema may also be used to define additional non-wire attributes (such as radiusAuthenticationType). This provides an additional element of flexibility, allowing new attributes to be defined and used within existing policy objects, without code changes.

Creating a sub-class is desirable in cases where conflicts are possible. Such conflicts can arise for example, when vendors have defined attributes which conflict with the standard RADIUS attribute space described in [6]-[8]. In this case, the radiusVendorId attribute should included and set to the SMI Vendor Code, indicating that the profile is specific to a given vendor, and contains potentially conflicting elements. Since a RADIUS server searching for a profile with objectclass=radiusProfileClass will encounter both base class profiles and subclasses, the radiusVendorId attribute is critical in allowing an implementation to differentiate the profiles it can understand from those that it cannot. Typically an implementation will only wish to work with profiles whose radiusVendorId is either not present, zero (IETF RADIUS) or set to their own SMI Vendor Code. As with addition of attributes to the base class, when attributes are added to a subclass, the RADIUS Dictionary class should modified to allow the subclass to be self-describing.

Since it is conceivable that RADIUS servers from two vendors may be deployed simultaneously, both desiring to store objects in the same LDAP-based directory service, and each implementing their own profile subclass, a method must be provided to allow a user to have more than one set of RADIUS profile and policy objects. This can be achieved by allowing the radiusProfilePointer to point to a container object rather than pointing to an object itself. The RADIUS server would then search the container for a RADIUS profile or policy with an appropriate radiusVendorId.

In order to prevent name conflicts, it is recommended that vendors adding their own attributes prepend a suffix to all attribute names. The IETF Schema Working Group has announced its intention to manage suffix allocation in order to avoid name conflicts. Rather than redefining existing attributes, vendor should create their own attributes using suffixes in order to avoid conflict.

To illustrate how extensibility features may be used, the additional attributes supported by a hypothetical BIGCO Profile Class are included.

4. User object additions

The RADIUS schema proposes addition of the following attributes to the user object:
MAY (radiusServiceType $ radiusFramedProtocol $ radiusFramedIPAddress $ radiusFramedIPNetmask $ radiusFramedRoute $ radiusFramedRouting $ radiusFilterId $ radiusFramedMTU $ radiusFramedCompression $ radiusLoginIPHost $ radiusLoginService $ radiusLoginTCPPort $ radiusCallbackNumber $ radiusCallbackId $ radiusFramedRoute $ radiusFramedIPXNetwork $ radiusClass $ radiusVSA $ radiusSessionTimeout $ radiusIdleTimeout $ radiusTerminationAction $ radiusCalledStationId $ radiusCallingStationId $ radiusLoginLATService $ radiusLoginLATNode $ radiusLoginLATGroup $ radiusFramedAppleTalkLink $ radiusFramedAppleTalkNetwork $ radiusFramedAppleTalkZone $ radiusPortLimit $ radiusLoginLATPort $ radiusTunnelType $ radiusTunnelMediumType $ radiusTunnelServerEndpoint $ radiusTunnelPrivateGroupId $ radiusTunnelAssignmentId $ radiusTunnelClientEndpoint $ radiusTunnelPreference $ radiusTunnelPassword $ radiusArapFeatures $ radiusArapZoneAccess $ radiusArapSecurity $ radiusPasswordRetry $ radiusPrompt $ npSessionsAllowed $ npAuthenticationType $ npEAPType $ npConstraint $ npIPPoolName $ radiusProfilePointer $ radiusVendorId)

5. Object definitions

The RADIUS schema includes definition of the following objects:

RADIUS Profile Class
RADIUS Policy Class
RADIUS Dictionary Class
EAP Dictionary Class

5.1. RADIUS Profile Class

(radiusProfileClass 1
  NAME 'radiusProfile'
  SUP profile
  PARENT (country $ organization $ organizationalUnit $
                    locality $ container)
     STRUCTURAL
     MUST (cn
     )
     MAY (radiusServiceType $ radiusFramedProtocol $ radiusFramedIPAddress $ radiusFramedIPNetmask $ radiusFramedRoute $ radiusFramedRouting $ radiusFilterId $ radiusFramedMTU $ radiusFramedCompression $ radiusLoginIPHost $ radiusLoginService $ radiusLoginTCPPort $ radiusCallbackNumber $ radiusCallbackId $ radiusFramedRoute $ radiusFramedIPXNetwork $ radiusClass $ radiusVSA $ radiusSessionTimeout $ radiusIdleTimeout $ radiusTerminationAction $ radiusCalledStationId $ radiusCallingStationId $ radiusLoginLATService $ radiusLoginLATNode $ radiusLoginLATGroup $ radiusFramedAppleTalkLink $ radiusFramedAppleTalkNetwork $ radiusFramedAppleTalkZone $ radiusPortLimit $ radiusLoginLATPort $ radiusTunnelType $ radiusTunnelMediumType $ radiusTunnelServerEndpoint $ radiusTunnelPrivateGroupId $ radiusTunnelAssignmentId $ radiusTunnelClientEndpoint $ radiusTunnelPreference $ radiusTunnelPassword $ radiusArapFeatures $ radiusArapZoneAccess $ radiusArapSecurity $ radiusPasswordRetry $ radiusPrompt $ npSessionsAllowed $ npAuthenticationType $ npEAPType $ npConstraint $ npIPPoolName $ radiusProfilePointer $ radiusVendorId)
radiusLoginLATNode $ radiusLoginLATGroup $ radiusFramedAppleTalkLink $ radiusFramedAppleTalkNetwork $ radiusFramedAppleTalkZone $ radiusPortLimit $ radiusLoginLATPort $ radiusTunnelType $ radiusTunnelMediumType $ radiusTunnelServerEndpoint $ radiusTunnelPrivateGroupId $ radiusTunnelAssignmentId $ radiusTunnelClientEndpoint $ radiusTunnelPreference $ radiusTunnelPassword $ radiusArapFeatures $ radiusArapZoneAccess $ radiusArapSecurity $ radiusPasswordRetry $ radiusPrompt $ npSessionsAllowed $ npAuthenticationType $ npEAPType $ npConstraint $ npIPPoolName $ radiusProfilePointer $ radiusVendorId $ radiusDictionaryPointer

5.2. RADIUS Policy Class

(radiusPolicyClass 1
 NAME 'radiusPolicy'
 SUP policy
 PARENT (country $ organization $ organizationalUnit $ locality $ container)
 STRUCTURAL
 MUST ( cn $ radiusProfilePointer
 )
 MAY ( npConstraint $ npSequence
 )
)

5.3. RADIUS Dictionary Class

(radiusDictionaryClass 1
 NAME 'radiusDictionaryClass'
 SUP top
 PARENT (country $ organization $ organizationalUnit $ locality $ container)
 STRUCTURAL
 MUST ( cn $ radiusDictionaryEntry
 )
)

5.4. EAP Dictionary Class

(eapDictionaryClass 1
 NAME 'eapDictionaryClass'
 SUP top
 PARENT (country $ organization $ organizationalUnit $ locality $ container)
)
5.5. BIGCO Profile Class

As described earlier, the base classes may be extended by attribute addition, subclassing, or both. An example of the subclassing approach is illustrated below. Here the bigcoProfileClass is created as a subclass of the radiusProfileClass and adds several attributes, each of which uses bigco as a suffix to avoid name collisions.

```plaintext
( bigcoProfileClass 1
  NAME 'bigcoProfile'
  SUP radiusProfileClass
  PARENT (country $ organization $ organizationalUnit $ locality $ container)
  STRUCTURAL
  MUST {
  }
  MAY ( bigcoBapRequired $ bigcoBapLinednLimit $ bigcoBapLinednTime $ bigcoDynDirServer
  }
)
```

6. Attribute definitions

6.1. New Attribute Types Used in the user object and RADIUS Profile Class

```plaintext
( radius radiusProfileClass 6
  NAME 'radiusServiceType'
  DESC 'The service to be provided to the user. Values include: Login(1), Framed(2), Callback Login(3), Callback Framed(4), Outbound(5), Administrative(6), NAS Prompt(7), Authenticate Only(8), Callback NAS Prompt(9)'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)
```

```plaintext
( radius radiusProfileClass 7
  NAME 'radiusFramedProtocol'
  DESC 'For Framed service, the protocol to be provided to the user. Values include PPP(1), SLIP(2), ARAP(3), Gandalf(4), Xylogics(5)'
)
EQUALITY integerMatch
SYNTAX 'INTEGER'
SINGLE-VALUE
)

(radius radiusProfileClass 8
  NAME 'radiusFramedIPAddress'
  DESC 'IP address to be assigned to the user
  in dotted decimal notation'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 9
  NAME 'radiusFramedIPNetmask'
  DESC 'Netmask to apply to the user
  in dotted decimal notation'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 10
  NAME 'radiusFramedRouting'
  DESC 'Routing method for the user.
  Values include None(1), Send(2),
  Listen(3), Send & Listen(4)'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 11
  NAME 'radiusFilterId'
  DESC 'String representing the filter list for the user'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
)

(radius radiusProfileClass 12
  NAME 'radiusFramedMTU'
  DESC 'Maximum Transmission Unit for the user'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 13
  NAME 'radiusFramedCompression'
  DESC 'Compression protocol to be used on
  the link. Values include: None(1),
  VJ compression(2),
  IPX header compression(3)'
)
EQUALITY integerMatch
SYNTAX 'INTEGER'
)

(radius radiusProfileClass 14
  NAME 'radiusLoginIPHost'
  DESC 'System with which to connect the user in dotted decimal notation'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 15
  NAME 'radiusLoginService'
  DESC 'Service to be used to connect the user to the login host. Values include Telnet(1), Rlogin(2), TCP Clear(3), PortMaster(4), and LAT(5)'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 16
  NAME 'radiusLoginTCPPort'
  DESC 'The TCP port with which the user is to be connected'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 19
  NAME 'radiusCallbackNumber'
  DESC 'Number to be called'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

(radius radiusProfileClass 20
  NAME 'radiusCallbackId'
  DESC 'Name of place to be called'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

(radius radiusProfileClass 22
  NAME 'radiusFramedRoute'
  DESC 'Routes to be plumbed for the user'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
)
( radius radiusProfileClass 23
  NAME 'radiusFramedIPXNetwork'
  DESC 'IPX Network number to be configured for the user'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 24
  NAME 'radiusClass'
  DESC 'Class attribute for the user'
  SYNTAX 'OCTETSTRING'
)

( radius radiusProfileClass 25
  NAME 'radiusVSA'
  DESC 'Vendor Specific Attribute for the user'
  SYNTAX 'OCTETSTRING'
)

( radius radiusProfileClass 27
  NAME 'radiusSessionTimeout'
  DESC 'Per-session time limit in seconds.
       After this expires, the action specified in Termination-Action is taken'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 28
  NAME 'radiusIdleTimeout'
  DESC 'The maximum number of consecutive seconds of idle connection allowed before session termination'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 29
  NAME 'radiusTerminationAction'
  DESC 'Action taken when specified service is completed. Values include Default(1) or RADIUS-Request(2)'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 34
  NAME 'radiusLoginLATService'
DESC 'Identity of the LAT service to use'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}'
SINGLE-VALUE

(radius radiusProfileClass 35
 NAME 'radiusLoginLATNode'
 DESC 'The node with which the user is to be automatically connected by LAT'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

(radius radiusProfileClass 36
 NAME 'radiusLoginLATGroup'
 DESC 'The LAT group codes which this user is authorized to use'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

(radius radiusProfileClass 37
 NAME 'radiusFramedAppleTalkLink'
 DESC 'The AppleTalk network number which should be used for the user'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(radius radiusProfileClass 38
 NAME 'radiusFramedAppleTalkNetwork'
 DESC 'The AppleTalk network number which the NAS should probe to allocate an AppleTalk node for the user'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'INTEGER'
)

(radius radiusProfileClass 39
 NAME 'radiusFramedAppleTalkZone'
 DESC 'The name of the Default AppleTalk Zone'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

(radius radiusProfileClass 62
 NAME 'radiusPortLimit'
 DESC 'Maximum number of ports to be provided'
 EQUALITY integerMatch
)
SYNTAX 'INTEGER'
SINGLE-VALUE
)

(radius radiusProfileClass 39
 NAME 'radiusLoginLATPort'
 DESC 'The Port with which the user is to connected by LAT'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

(radius radiusProfileClass 64
 NAME 'radiusTunnelType'
 DESC 'String representing the type of tunnel to be set up, of the form Tag: Value. Values include PPTP(1), L2F(2), L2TP(3), ATMP(4), VTP(5), AH(6), IP-IP(7).'
 SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass 65
 NAME 'radiusTunnelMediumType'
 DESC 'String representing the medium for the tunnel to run over, of the form Tag: Value. Values include IP(1), X.25(2), ATM(3), Frame Relay(4).'
 SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass 67
 NAME 'radiusTunnelServerEndpoint'
 DESC 'String representing the address of the tunnel server, of the form Tag: Value. The format of the value field depends on the tunnelMediumType attribute'
 SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass ?
 NAME 'radiusTunnelPrivateGroupId'
 DESC 'String representing the Private Group Id for the tunnel, of the form Tag: Value.'
 SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass ?
 NAME 'radiusTunnelPreference'
 DESC 'String representing the tunnel preference for the tunnel, of the form Tag: Value.'
 SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass ?

NAME 'radiusTunnelAssignmentId'
DESC 'String representing the Tunnel Assignment Id for the
tunnel, of the form Tag: Value.'
SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass 71
NAME 'radiusTunnelClientEndpoint'
DESC 'String representing the Tunnel Client Endpoint for the
tunnel, of the form Tag: Value.'
SYNTAX 'OCTETSTRING'
)

(radius radiusProfileClass 71
NAME 'radiusArapFeatures'
DESC 'This is a compound string containing info that
the NAS should send to the user in the ARAP
feature flags packet'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}'
SINGLE-VALUE
)

(radius radiusProfileClass 72
NAME 'radiusArapZoneAccess'
DESC 'This field controls access to ARAP zones.
Values include
   Only allow access to default zone(1),
   Use zone filter inclusively(2),
   Use zone filter exclusively(4)'
EQUALITY integerMatch
SYNTAX 'INTEGER'
SINGLE-VALUE
)

(radius radiusProfileClass 73
NAME 'radiusArapSecurity'
DESC 'This field contains an integer
   specifying the security module signature,
   which is a Macintosh OSTYPE'
EQUALITY integerMatch
SYNTAX 'INTEGER'
SINGLE-VALUE
)

(radius radiusProfileClass 75
NAME 'radiusPasswordRetry'
DESC 'This is an integer specifying the number
   of password retry attempts to permit the user'
EQUALITY integerMatch
SYNTAX 'INTEGER'
SINGLE-VALUE
)
( radius radiusProfileClass 76
   NAME 'radiusPrompt'
   DESC 'This attribute is used only in RADIUS Access-Challenge packets and indicates if the NAS should echo the user’s response as entered. Values include No Echo (0), or Echo(1).'
   EQUALITY integerMatch
   SYNTAX 'INTEGER'
   SINGLE-VALUE
)

( radius radiusProfileClass 257
   NAME 'npEAPType'
   DESC 'Allowable EAP types, in order of preference. If this attribute has a value, EAP must be included in the allowable authentication types.'
   EQUALITY caseIgnoreIA5Match
   SYNTAX 'IA5String{128}'
   SINGLE-VALUE
)

( radius radiusProfileClass 258
   NAME 'npConstraint'
   DESC 'A string expressing conditions which must hold in order for an Access-Accept to be sent. The string is of the format MATCH ( <attribute> = <pattern/value> OR <pattern/value> ) <AND/OR> TIMEOUTOFDAY. Brackets () can be used to group. When multiple msNPConstraints are present, all of them must be satisfied in order for a profile to be executed.'
   EQUALITY caseIgnoreIA5Match
   SYNTAX 'IA5String'
)

( radius radiusProfileClass 259
   NAME 'npIPPoolName'
   DESC 'The name of the IP Address Pool out of which the user’s IP address should be allocated.'
   EQUALITY caseIgnoreIA5Match
   SYNTAX 'IA5String'
)

( radius radiusProfileClass 260
   NAME 'npSessionsAllowed'
   DESC 'This attribute indicates the number of simultaneous sessions allowed for this user.'
   EQUALITY integerMatch
   SYNTAX 'INTEGER'
   SINGLE-VALUE
)

( radius radiusProfileClass 261
   NAME 'npAuthenticationType'
DESC 'Allowable authentication types (EAP, CHAP, PAP, MS-CHAP, etc.) in order of preference. If an attribute isn’t included, it isn’t allowed.'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}'
SINGLE-VALUE

(radius radiusProfileClass 262
 NAME 'radiusProfilePointer'
 DESC 'Distinguished Name of a RADIUS Profile Object.'
 EQUALITY distinguishedNameMatch
 SYNTAX 'DN'
 SINGLE-VALUE
)

(radius radiusProfileClass 263
 NAME 'radiusVendorId'
 DESC 'SMI Vendor Id. A non-zero value denotes a profile non-compliant with RFC 2138 and 2139.'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(radius radiusProfileClass 264
 NAME 'radiusDictionaryPointer'
 DESC 'A Distinguished Name pointing to the RADIUS dictionary for this profile. If not present the default dictionary is used.'
 EQUALITY distinguishedNameMatch
 SYNTAX 'DN'
 SINGLE-VALUE
)

6.2. New Attribute Types Used in the RADIUS Policy Class

(radius radiusPolicyClass 2
 NAME 'npSequence'
 DESC 'An integer indicating the order in which policy objects are to be evaluated.'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

6.3. New Attribute Types Used in the RADIUS Dictionary Class

(radius radiusDictionaryClass 1
 NAME 'dictionaryEntry'
DESC 'A dictionary entry in the RADIUS dictionary, of the form Attribute-Number:[Vendor-Type:]ldapDisplayName:Type. Vendor-Type may only be present with Attribute-Number=26 (Vendor Specific).’
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}’

6.4. New Attribute Types Used in the BIGCO Profile Class

( bigco bigcoProfileClass 263
  NAME 'bigcoBapRequired'
  DESC 'This attribute indicates whether Bandwidth Allocation Protocol (BAP) is required for this user. Values include BAP Not Required (0) and BAP Required (1).’
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( bigco bigcoProfileClass 264
  NAME 'bigcoBapLinednLimit'
  DESC 'Percent of capacity utilized at which to bring a line down for this user. ’
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( bigco bigcoProfileClass 265
  NAME 'bigcoBapLinednTime'
  DESC 'Time in seconds for the capacity utilization calculation.’
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( bigco bigcoProfileClass 266
  NAME 'bigcoDynDirServer'
  DESC 'Fully qualified domain name or IP address of the dynamic directory server for this user.’
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String{128}’
  SINGLE-VALUE
)

7. Security issues

Integration of a RADIUS server with an LDAP-based directory service can result in a variety of security threats, including:
Rogue LDAP-servers
Theft of passwords
Inappropriate use

These threats are discussed in turn.

7.1. Rogue LDAP servers

Were a rogue LDAP server to respond to queries from the RADIUS server and have its responses accepted, it is possible that users could gain inappropriate access to the network. In order to protect against this, the conversation between the RADIUS server and the LDAP-based directory service SHOULD be mutually authenticated via SSL/TLS or IPSEC.

7.2. Theft of passwords

RADIUS servers supporting PAP authentication or attributes such as Tunnel-Password SHOULD provide for confidentiality of packets sent to and from the LDAP server. This can be achieved using SSL/TLS or IPSEC ESP.

7.3. Inappropriate use

This schema is intended for use by a RADIUS server integrating with an LDAP-enabled directory. This schema SHOULD NOT be used by devices looking to access the directory directly.

LDAP-enabling of devices would introduce several security problems. As described in [13], LDAP-enabling a RADIUS server requires that the RADIUS server be given permissions to access a user’s RADIUS objects and attributes. If the dynamic attributes described in [12] are supported, then the RADIUS service must also be able to write those attributes to the DS. As a result, the administrator of the RADIUS server should exercise care to ensure that the RADIUS account password is not compromised. If at all possible, the RADIUS server should be physically secured.

In contrast, LDAP-enabling of devices requires that devices be given these access-rights. This can be achieved by making the devices members of a group, and giving the group access rights to this portion of the schema. However, such an expansion of access rights is undesirable, since while RADIUS servers can often be physically secured, widely deployed devices typically cannot be.

Furthermore, direct use of LDAP across a WAN typically requires that LDAP pass through a firewall. This is problematic since LDAP-based directories can be used to store a wide variety of data, much of it sensitive. Thus without implementing an LDAP proxy to limit access only to appropriate portions of the schema, it is difficult to enforce security. Since humans are notoriously lax in administration of access rights, an attacker obtaining a device password would typically also
obtain access not only to RADIUS attributes for every user, but to other information as well.

Beyond the security issues, LDAP-enabling of devices increases the size of the device binaries, and may in some cases introduce dependencies in the device boot sequence that can be problematic.

8. Acknowledgments

Thanks to Steven Judd, Ashwin Palekar, David Eitelbach, Narendra Gidwani and Donald Rule of Microsoft for useful discussions of this problem space.

9. References


10. Authors’ Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: 425-936-6605
EMail: bernarda@microsoft.com