

Recommendation H.350 Directory Services Architecture for Multimedia Conferencing

Summary

This Recommendation describes a directory services architecture for multimedia conferencing using LDAP. Standardized directory services can support association of persons with endpoints, searchable white pages, and clickable dialling. Directory services can also assist in the configuration of endpoints, and user authentication based on authoritative data sources. This document describes a standardized LDAP schema to represent endpoints on the network and associate those endpoints with users. It discusses design and implementation considerations for the inter-relation of video and voice-specific directories, enterprise directories, call servers and endpoints.

Keywords

LDAP, Directory Services, H.320, H.323, H.235, SIP

Contact:	Tyler Miller Johnson University of North Carolina at Chapel Hill USA	Tel: +1.919.843.7004 Fax: +1.919.843.7008 Email: Tyler_Johnson@unc.edu
-----------------	--	--

Attention: This is not a publication made available to the public, but **an internal ITU-T Document** intended only for use by the Member States of the ITU, by ITU-T Sector Members and Associates, and their respective staff and collaborators in their ITU related work. It shall not be made available to, and used by, any other persons or entities without the prior written consent of the ITU-T.

Table of Contents

1	Scope	4
1.1	Design Goals	5
1.2	Extending the Schema	5
1.2.1	Extension Through Subclass	6
1.2.2	Extension Through The Use Of Auxiliary Classes	6
1.2.3	Object Identifiers	7
2	References.....	7
2.1	Normative References	7
2.2	Non-Normative References	8
3	Definitions	8
4	Abbreviations.....	8
5	Conventions	8
6	commURIObject Definition	9
6.1	commURIObject.....	9
6.2	commURI	9
7	CommObject Definition	10
7.1	commObject	10
7.2	commUniqueId.....	10
7.3	commOwner	10
7.4	commPrivate.....	11
8	CommObject LDIF Files	12
8.1	LDIF for commURIObject.....	12
8.2	LDIF for commObject.....	13
A	Annex A Indexing Profile.....	15
I	Relationship of Enterprise Directories to commObject Directories.....	16
I.1	People Versus Resources.....	16
I.2	Security and Authentication	17
II	Call Flow Scenarios.....	18
II.1	White Pages User Lookup	18
II.2	Basic Directory Enabled Registration	19
II.3	Secure Single Sign-On	19
III	Electronic Attachments.....	21

1 Scope

This Recommendation describes a directory services architecture for multimedia conferencing using LDAP. Standardized directory services can support association of persons with endpoints, searchable white pages, and clickable dialling. Directory services can also assist in the configuration of endpoints, and user authentication based on authoritative data sources. This document describes a standardized LDAP schema to represent endpoints on the network and associate those endpoints with users. It discusses design and implementation considerations for the inter-relation of video and voice-specific directories, enterprise directories, call servers and endpoints.

The use of a common, authoritative data source for call server, endpoint, user, authentication and white pages information is an important aspect of large scale multimedia conferencing environments. Without a common data source, service providers must create separate processes to manage each of these functions. By standardizing the LDAP schema used to represent the underlying data, products from different system vendors can be deployed together to create an overall application environment. For example, a white pages search engine developed by one provider could serve directory information to IP telephones produced by a second provider, with signalling managed by a call server produced by yet a third provider. Each of these disparate systems can access the same underlying data source, reducing or eliminating the need to coordinate separate management of each system. A significant benefit to the user is that the management of this data can be incorporated into existing customer management tools, allowing for quick and flexible scaling up of applications. Indeed, many technology providers have already incorporate LDAP into their products, but have been forced to do so without benefit of a standardized schema. This Recommendation represents an effort to standardize those representations to improve interoperability and performance.

While URLs are already standardized for several conferencing protocols, their representation in a directory is not. This Recommendation supports a standardized way for URLs to be searched and located. This is a necessary step to support ‘clickable dialling.’

Management of endpoint configurations can be improved if the correct settings are stored by the service provider in a location that is accessible to both service provider and endpoint. LDAP provides a convenient storage location that can be accessed by both call server and endpoint; thus it is possible to use the directory to support endpoint configuration, which is important for simplified operation and supporting user mobility. Note that other technologies also support endpoint configuration, notably the use of SNMP for complete configuration and SRV records for obtaining registration server addresses. Therefore H.350 should be viewed not as an authoritative endpoint configuration architecture, but rather one tool that can assist with this task. Note that the use of H.350 has as a feature endpoint specific configuration, where it is desirable that each endpoint have a unique configuration.

This architecture uses a generic object class, called `commObject`, to represent attributes common to any video or voice protocol. Auxiliary classes represent specific protocols, such as h.323, h.235, or h.320, as described in the H.350.X series of Recommendations. Multiple H.350.X classes can be combined to represent endpoints that support more than one protocol. For example, endpoints that support H.323, H.235 and H.320 would include H.350, H.350.1, H.350.2, and H.350.3 in their LDAP representations. Further, each entry should contain `commObject` to serve as the entry’s structural object class.

There are two basic components in the architecture. The `commURI` object is a class whose only purpose is to link a person or resource to a `commObject`. By placing a `commURI` ‘pointer’ in an individual’s directory entry, that individual becomes associated with the particular targeted `commObject`. Similarly, `commObject` contains a pointer, called `commOwner`, which points to the

individual or resource that is associated with the commObject. In this way, people or resources can be associated with endpoints. The only change required in the enterprise directory is the addition of the simple object class commURI. CommObject data may be instantiated in the same or in entirely separate directories, thus allowing flexibility in implementation.

1.1 Design Goals

Large-scale deployments of IP video and voice services have demonstrated the need for complementary directory services middleware. Service administrators need call servers that are aware of enterprise directories to avoid duplication of account management processes. Users need 'white pages' to locate other users with whom they wish to communicate. All of these processes should pull their information from canonical data sources in order to reduce redundant administrative processes and ensure information accuracy. The following design criteria are established for this architecture. The architecture will:

1. Enable endpoint information to be associated with people. Alternately it enables endpoint information to be associated with resources such as conference rooms or classrooms.
2. Enable online searchable "white pages" where dialling information (e.g. endpoint addresses) can be found, along with other "traditional" directory information about a user, such as name, address, telephone, email, etc.
3. Enable all endpoint information to be stored in a canonical data source (the Directory), rather than local to the call server, so that endpoints can be managed through manipulations of an enterprise directory, rather than by direct entry into the call server.
4. Support the creation of very large-scale distributed directories. These include white pages "portals" that allow searching for users across multiple institutional directories. In this application, each enterprise directory registers itself with (or is unknowingly discovered by) a directory of directories that is capable of searching across multiple LDAP directories.
5. Be able to support multiple instances of endpoints per user or resource.
6. Represent endpoints that support more than one protocol, for example endpoints that are both H.320 and H.323.
7. Store enough information about endpoint configuration so that correct configuration settings can be documented to end users on a per-endpoint basis, as a support tool, or loaded automatically into the endpoint.
8. Be extendable as necessary to allow implementation specific attributes to be included.
9. Be non-invasive to the enterprise directory, so that support for multimedia conferencing can be added in a modular fashion without significant changes to the enterprise directory.

The scope of this Recommendation does not include extensions of functionality to protocols as defined within the protocols themselves. It is not the intent of the Recommendation to add features, but merely to represent existing protocol attributes. The exception to this case is when functionality is implied by the directory itself, such as the commPrivate attribute.

1.2 Extending the Schema

H.350 object classes may be extended as necessary for specific implementations. For example, a class may be extended to support billing reference codes. Extensions to the schema are not considered as part of the standard and do not signify compliance.

In some cases it may be necessary to extend the H.350 schemas in order to represent more information than is supported by the Recommendations. This may be important for developers that implement proprietary endpoint functionality that needs to be represented by attributes in the

directory. It may also be important for enterprise applications. For example 'modelNumber', and 'accountNumber' are examples of attributes that are not defined in the standard but may be useful if implemented. Adding attributes to this architecture must be done in a way that does not break compatibility with this Recommendation.

A full discussion of schema design and extension is beyond the scope of this document. See IETF RFC 2252 for details. Two basic approaches to schema extension that do not break compatibility with this Recommendation are extension through subclass and extension through the use of auxiliary classes.

1.2.1 Extension Through Subclass

It is possible to create subclass of an existing predefined object class in order to add new attributes to it. To create a subclass, a new object class must be defined that is a subclass of the existing one by indicating in the definition of the new class that the existing class is its superior. Once the subclass is created, new attributes can be defined within it.

The following example shows how the commObject class can be sub classed in order to add an attribute to represent a billing account and a billing manager.

```
objectclass ( BillingInfo-OID
NAME 'BillingInfo'
DESC 'Billing Reference Information'
SUP commObject STRUCTURAL
MAY ( BillingAccount $ BillingManager $ )
)
```

Note that BillingInfo-OID must be replaced by an actual OID. Also note that whenever a structural class is extended, its subclass must also be structural.

The following sample entry shows the newly created attributes. This example also uses H.350.1 for h323Identity.

```
dn: commUniqueId=2000,ou=h323identity, dc=company, dc=com
objectclass: top
objectclass: commObject
objectclass: h323Identity
objectclass: BillingInfo
commUniqueId: 2000
BillingAccount: 0023456
BillingManager: John Smith
```

Note that this example and approach demonstrate extension of the general commObject object class, and not any individual H.350.X classes. If it is desired to extend an H.350.X auxiliary class, then that should be accomplished through the definition of additional auxiliary classes that support the desired attributes, as described in section 1.2.2.

1.2.2 Extension Through The Use Of Auxiliary Classes

It is possible to add attributes to an LDAP entry by defining an auxiliary class containing the new attributes and applying those attributes to instantiated values in the directory. The auxiliary class will not be sub classed from any existing object class. Note that it should have the special class top as its superior. The following example creates the same billing account and billing manager attributes as the previous example, but does so by defining them in their own auxiliary class.

```
objectclass ( BillingInfo-OID
```

```
NAME 'BillingInfo'  
DESC 'Billing Reference Information'  
SUP top AUXILIARY  
MAY ( BillingAccount $ BillingManager $ )  
)
```

Note how the superior was changed from commObject to top and the object class changed from being a structural to auxiliary.

It is recommended that all attributes in the auxiliary class be optional rather than mandatory. In this way, the auxiliary object class itself can be associated with an entry regardless of whether any values for its attributes are present.

The following example shows a sample endpoint that utilizes the new auxiliary class and attributes. This example also uses H.350.1 for h323Identity.

```
dn: commUniqueId=2000,ou=h323identity, dc=company, dc=com  
objectclass: top  
objectclass: commObject  
objectclass: BillingInfo  
commUniqueId: 2000  
BillingAccount: 0023456  
BillingManager: John Smith
```

1.2.3 Object Identifiers

An attribute's Object Identifier (OID) is a unique numerical identifier usually written as a sequence of integers separated by dots. For example, the OID for the commUniqueId is 0.0.8.350.1.1.2.1.1. All attributes must have an OID. OIDs can be obtained from anyone who has one and is willing to delegate a portion of it as an arc, keeping a record of the arc to avoid duplication. Further, the Internet Assigned Numbers Authority (IANA) gives out OIDs to any organization that asks.

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation

2.1 Normative References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published

- IETF RFC 3377 (2002), *Lightweight Directory Access Protocol (v3): Technical Specification*.

2.2 Non-Normative References

- ITU-T Recommendation H.323 (2000), *Packet-based multimedia communications systems*.
- ITU-T Recommendation H.235 (2000), *Security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals*.
- ITU-T Recommendation H.225.0 (2000), *Call signalling protocols and media stream packetization for packet-based multimedia communications systems*.
- ITU-T Recommendation H.320 (1999), *Narrow-band visual telephone systems and terminal equipment*.
- Timothy A. Howes, PhD, Mark C. Smith, Gordon S. Good, New Riders Publishing (1999), ISBN: 1578700701, *Understanding And Deploying LDAP Directory Services*.
- Timothy A. Howes, PhD, Mark C. Smith, New Riders Publishing (1997), ISBN: 1578700000, *LDAP Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*.

3 Definitions

The following terms used throughout the document:

call server: a protocol-specific signalling engine that routes video or voice calls on the network. In H.323 this entity is a gatekeeper. In SIP, this entity is a SIP Proxy Server. Note that not all signalling protocols use a call server.

endpoint: a logical device that provides video and/or voice media encoding/decoding, and signalling functions. Examples include:

1. a group teleconferencing appliance that is located in a conference room
2. an IP telephone.
3. a software program that takes video and voice from a camera and microphone and encodes it and applies signalling using a host computer.

enterprise directory: A canonical collection of information about users in an organization. Typically this information is collected from a variety of organizational units to create a whole. For example, Human Resources may provide name and address, Telecommunications may provide the telephone number, Information Technology may provide the email address, etc. For the purposes of this architecture, it is assumed that an enterprise directory is accessible via LDAP.

White Pages: An application that allows end users to look up the address of another user. This may be web-based or use some other user interface.

4 Abbreviations

CN: Common Name

DN: Distinguished Name

LDAP: Lightweight Directory Access Protocol as defined in RFC 1777.

RDN: Relative Distinguished Name

5 Conventions

In this Recommendation, the following conventions are used:

"Shall" indicates a mandatory requirement.

"Should" indicates a suggested but optional course of action.

"May" indicates an optional course of action rather than a recommendation that something take place.

References to clauses, sub clauses, annexes and appendices refer to those items within this Recommendation unless another specification is explicitly listed.

6 commURIObject Definition

Auxiliary object class that contains the commURI attribute. This attribute is added to a person or resource object to associate one or more commObject instances with that object. Its values are LDAP URIs that point to the associated commObjects, for example, to a user's H.323 conferencing station and SIP IP phone. Note that multiple instances of commURI need not point to the same commObject directory. In fact, each commURI instance could point to an endpoint managed by a different service provider.

6.1 commURIObject

```
OID: 0.0.8.350.1.1.1.2.1
objectclasses: (0.0.8.350.1.1.1.2.1
NAME 'commURIObject'
DESC 'object that contains the URI attribute type'
SUP top AUXILIARY
MAY ( commURI )
)
```

6.2 commURI

```
OID: 0.0.8.350.1.1.1.1.1
attributetypes: ( 0.0.8.350.1.1.1.1.1
NAME 'commURI'
DESC 'Labeled URI format to point to the distinguished name of the commUniqueId'
EQUALITY caseExactMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Application utility class

Standard

Number of values

multi

Definition

Labelled URI containing an LDAP URL identifying the directory containing the referenced commObject instance. The search filter specified by this LDAP URL shall specify an equality search of the commUniqueId attribute of the commObject class.

Permissible values (if controlled)

Notes

Used to find the endpoint of the user in question. The label field may be used to represent the function of the endpoint, such as 'home IP phone' or 'desktop video' for user interface display purposes.

Note that the label portion of the field may contain spaces as in the example below showing 'desktop video'.

Semantics

Example applications for which this attribute would be useful

Example (LDIF fragment)

```
commURI: ldap://directory.acme.com/dc=acme,dc=com??sub?(commUniqueId=bob)
desktop video
```

7 CommObject Definition

Abstraction of video or voice over IP device. The commObject class permits an endpoint (H.323 endpoint or SIP user agent or other protocol endpoint) and all their aliases to be represented by a single entry in a directory. Note that every directory entry should contain commObject as the entry's structural object class. That entry may also contain H.350.X auxiliary classes.

7.1 commObject

```
OID: 0.0.8.350.1.1.2.2.1
objectclasses: (0.0.8.350.1.1.2.2.1
NAME 'commObject'
DESC 'object that contains the Communication attributes'
SUP top STRUCTURAL
MUST commUniqueId
MAY ( commOwner $ commPrivate )
)
```

7.2 commUniqueId

```
OID: 0.0.8.350.1.1.2.1.1
attributetypes: (0.0.8.350.1.1.2.1.1
NAME 'commUniqueId'
DESC 'To hold the endpoints unique Id'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

Application utility class

standard

Number of values

multi

Definition

The endpoint's unique ID.

Permissible values (if controlled)

Notes

This is the RDN of this object. In practice, there will always be one and only one commUniqueId for every endpoint. This attribute uniquely identifies an endpoint in the commObject directory. It must be unique within that directory, but need not be unique globally. This attribute has no relationship to the enterprise directory.

Semantics

Example applications for which this attribute would be useful

Example (LDIF fragment)

```
commUniqueId: bob
```

7.3 commOwner

```
OID: 0.0.8.350.1.1.2.1.2
attributetypes: 0.0.8.350.1.1.2.1.2
NAME 'commOwner'
```

```
DESC 'Labeled URI to point back to the original owner'  
EQUALITY caseExactMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Application utility class

Standard

Number of values

multi

Definition

Labelled URI format to point back to the person or resource object associated with this entry.

Permissible values (if controlled)

Notes

Used as a reverse entry finder of the owner(s). This attribute may point to groups. Note that this URI can point to a cn, but in applications where it is desired to bind authentication information across both the commObject and enterprise directories, it may be desirable that commOwner points to a dn rather than a cn, thus uniquely identifying the owner of the commObject.

Semantics

Example applications for which this attribute would be useful

Example (LDIF fragment)

```
commOwner: ldap://directory.acme.com/dc=acme,dc=com??sub?(cn=bob%20smith)  
commOwner: uid=bob,ou=people,dc=acme,dc=com
```

7.4 commPrivate

```
OID: 0.0.8.350.1.1.2.1.3  
attributetypes: (0.0.8.350.1.1.2.1.3  
NAME 'commPrivate'  
DESC 'To decide whether the entry is visible to world or not'  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

Application utility class

Standard

Number of values

multi

Definition

To be used by the user and indicate privacy options for an endpoint, i.e. unlisted number.

Permissible values (if controlled)

Notes

This attribute is defined as Boolean. Future version of this Recommendation may develop a controlled vocabulary for this attribute to accommodate multiple types of privacy.

Semantics

Example applications for which this attribute would be useful

Example (LDIF fragment)

```
commPrivate: true
```

8 CommObject LDIF Files

This section contains a schema configuration file for commURIObject and commObject that can be used to configure an LDAP server to support these classes.

8.1 LDIF for commURIObject

```
# Communication Object Schema
#
# Schema for Representing Communication Objects in an LDAP Directory
#
# Abstract
#
# This document defines the schema for representing Communication
# objects in an LDAP directory [LDAPv3]. It defines schema elements
# to represent a communication object URI [commURIObject].
#
#
#           .1 = Communication related work
#           .1.1 = commURIObject
#           .1.1.1 = attributes
#           .1.1.2 = objectclass
#           .1.1.3 = syntax
#
# Attribute Type Definitions
#
#   The following attribute types are defined in this document:
#
#       commURI
dn: cn=schema
changetype: modify
#
# if you need to change the definition of an attribute,
#       then first delete and re-add in one step
#
# if this is the first time you are adding the commObject
# objectclass using this LDIF file, then you should comment
# out the delete attributetypes modification since this will
# fail. Alternatively, if your ldapmodify has a switch to continue
# on errors, then just use that switch -- if you're careful
#
delete: attributetypes
attributetypes: (0.0.8.350.1.1.1.1.1.1 NAME 'commURI' )
-
#
# re-add the attributes -- in case there is a change of definition
#
#
add: attributetypes
attributetypes: (0.0.8.350.1.1.1.1.1
    NAME 'commURI'
    DESC 'Labeled URI format to point to the distinguished name of the
commUniqueId'
    EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
-
# Object Class Definitions
#
#   The following object classes are defined in this document:
#
#       commURIObject
```

```
#
# commURIObject
#
#   This auxiliary object class represents a URI attribute type
#
#
delete: objectclasses
objectclasses: (0.0.8.350.1.1.1.2.1 NAME 'commURIObject' )
-
add: objectclasses
objectclasses: (0.0.8.350.1.1.1.2.1
  NAME 'commURIObject'
  DESC 'object that contains the URI attribute type'
  SUP top AUXILIARY
  MAY ( commURI )
  )
-
#
# end of LDIF
#
```

8.2 LDIF for commObject

```
# Communication Object Schema
#
# Schema for Representing Communication Objects in an LDAP Directory
#
# Abstract
#
# This document defines the schema for representing Communication
# objects in an LDAP directory [LDAPv3]. It defines schema elements
# to represent a communication object [commObject].
#
#
#           .1 = Communication related work
#           .1.2 = commObject
#           .1.2.1 = attributes
#           .1.2.2 = objectclass
#           .1.2.3 = syntax
#
#
# Attribute Type Definitions
#
#   The following attribute types are defined in this document:
#
#       commUniqueId
#       commOwner
#       commPrivate
dn: cn=schema
changetype: modify
#
# if you need to change the definition of an attribute,
#       then first delete and re-add in one step
#
# if this is the first time you are adding the commObject
# objectclass using this LDIF file, then you should comment
# out the delete attributetypes modification since this will
# fail. Alternatively, if your ldapmodify has a switch to continue
# on errors, then just use that switch -- if you're careful
#
delete: attributetypes
attributetypes: (0.0.8.350.1.1.2.1.1 NAME 'commUniqueId' )
```

```
attributetypes: (0.0.8.350.1.1.2.1.2 NAME 'commOwner' )
attributetypes: (0.0.8.350.1.1.2.1.3 NAME 'commPrivate' )
-
#
# re-add the attributes -- in case there is a change of definition
#
#
add: attributetypes
attributetypes: (0.0.8.350.1.1.2.1.1
  NAME 'commUniqueId'
  DESC 'To hold the endpoints unique Id'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
attributetypes: (0.0.8.350.1.1.2.1.2
  NAME 'commOwner'
  DESC 'Labeled URI to point back to the original owner'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
attributetypes: (0.0.8.350.1.1.2.1.3
  NAME 'commPrivate'
  DESC 'To decide whether the entry is visible to world or not'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
-
# Object Class Definitions
#
#   The following object classes are defined in this document:
#
#       commObject
#
# commObject
#
#
delete: objectclasses
objectclasses: (0.0.8.350.1.1.2.2.1 NAME 'commObject' )
-
add: objectclasses
objectclasses: (0.0.8.350.1.1.2.2.1
  NAME 'commObject'
  DESC 'object that contains the Communication attributes'
  SUP top STRUCTURAL
  MUST commUniqueId
  MAY ( commOwner $ commPrivate )
  )
-
#
# end of LDIF
#
```

Annex A Indexing Profile

A Annex A Indexing Profile

Indexing of attributes is an implementation-specific activity and depends upon the desired application. Non-indexed attributes can result in search times sufficiently long to render some applications unusable. Notably, user and alias lookup should be fast. The Annex A Indexing Profile describes an indexing configuration for commObject directories that will be optimized for use in directory of directories applications. Use of this profile is optional.

commURI: no recommendation.

commUniqueId: equality

commOwner: presence

commPrivate: presence

Appendix I Implementation Considerations

I Relationship of Enterprise Directories to commObject Directories

CommObject information is located separately from person or resource information. Its location may be a sub-tree of the larger enterprise directory or on a separate logical server. The person directory will continue to host traditional person or resource information such as name, telephone, address, etc. Additionally it will have a commURI link to the commUniqueId attribute in commObject. Rather than extending the enterprise directory's person object class, this linking provides the following advantages:

1. Changes to the enterprise directory are not to be undertaken lightly and are often not under the administrative control of the video/voice over IP service provider.
2. Elements associated with video and voice over IP communications are very dynamic. The technology itself is changing quickly in relation to the enterprise directory. For example, changes to a specific protocol would require changes to the enterprise directory if its representation were inherited from an enterprise directory person object class and embedded within the enterprise directory. Separation allows changes to the commObject LDAP infrastructure without modifying the enterprise directory.
3. A call server may need to access commObject data very differently than other applications access the enterprise directory. A separate server can be tuned for performance and access policy to accommodate these implementation requirements, if so desired. For example, a call server may need to query the commObject server many times per second in order to handle real-time call processing, or it may read and cache many commObject attributes at once.

Any user or resource with multimedia conferencing capabilities should have an instance of commObject created and linked to an existing entry in the enterprise directory with a commURI. Call servers may operate in one of two ways. The simplest method is for the call server to periodically read instances of commObject into its internal endpoint table. The preferred and more scalable method is for the call server to query the commObject server each time it needs information, such as upon endpoint registration or call setup.

I.1 People Versus Resources

Some multimedia conferencing implementations are heavily endpoint-oriented, whereas others are user-oriented. For example, it is common to encounter a group video teleconferencing endpoint in a conference room whose identity never changes. This endpoint may be referred to as 'Conference Room 201'. This endpoint is not associated with any particular person, but is associated with the resource Conference Room 201 and is shared by whoever needs to use the conference room.

On the other hand, some endpoints are user context-specific, deriving their identities from the current users. For example, when logging onto a computer as jdoe, a computer-based endpoint may configure itself with the address of jdoe as stored in that user's profile and register with a call server accordingly. Other users logging onto the same computer may have different identities associated with them, hence their registration messages will contain different identity information.

This dichotomy of users versus resources makes it difficult to associate endpoints with users or resources. Indeed, while person object classes are readily available, resource object classes are less so. Linking commObject to a person via a commURI generalizes this relationship. If a commOwner attribute is pointing to a person object class, then that commObject is associated with that person. If a commOwner attribute is pointing to a resource object class, then that commObject is associated with that resource. Conversely, either people or resources can have commURI pointers that

associate endpoints with them. Enterprise directories that only support people and not resources may choose to simply treat resources as people.

I.2 Security and Authentication

Most authentication realms are oriented toward people. Thus, shared resources such as conference room teleconferencing systems are often less secure, because they have no meaningful authentication identity associated with them. It is beyond the scope of this Recommendation to discuss authentication concerns, but implementers are strongly encouraged to thoroughly explore the security aspects of various architectural choices.

Access control lists and other security mechanisms associated with the directory are beyond the scope of this Recommendation. Implementers are encouraged to carefully consider privacy and security of the data in the directory.

Appendix II Call Flows

II Call Flow Scenarios

The following call flows represent possible application scenarios demonstrating how H.350 can be used. These call flows are illustrative only and are non-normative. Specifically, H.350 defines how data elements are represented in LDAP directories, not how those data elements are used.

II.1 White Pages User Lookup

Figure A.1 shows how an LDAP-enabled endpoint can search a directory to find a user, get that user's commObject information, and dial the user's endpoint. In this scenario the endpoint could be pre-configured to search a particular enterprise directory, or it could be preconfigured to search a directory portal that itself searches many directories. This scenario demonstrates how an appliance can handle the searching, presenting choices and results through a user interface. However, this functionality can also be implemented through a web page, where the search criteria are entered into a web form and results are returned and displayed via web page, thus enabling clickable dialling. Note that while this example illustrates an endpoint doing direct LDAP lookup, it is also possible that a call server could perform lookups on behalf of the endpoint and pass the information to the endpoint through an alternate communication path, thus centralizing some aspects of white page access.

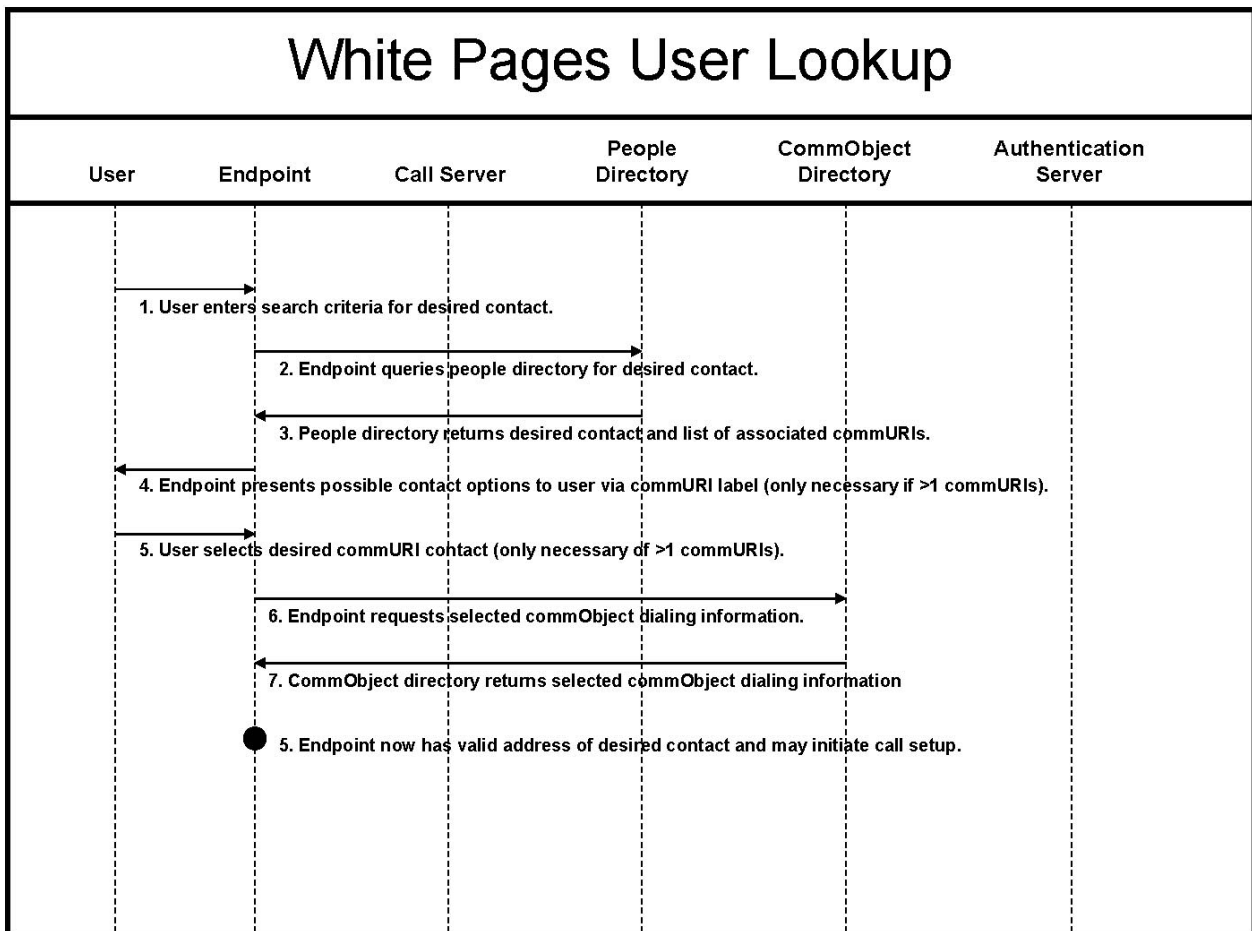


Figure A.1 White pages user lookup

II.2 Basic Directory Enabled Registration

Figure A.2 shows how a call server can access a commObject directory to retrieve endpoint information, thus eliminating the need for the call server to have a proprietary internal endpoint database. The call server is always accessing authoritative and up to date information.

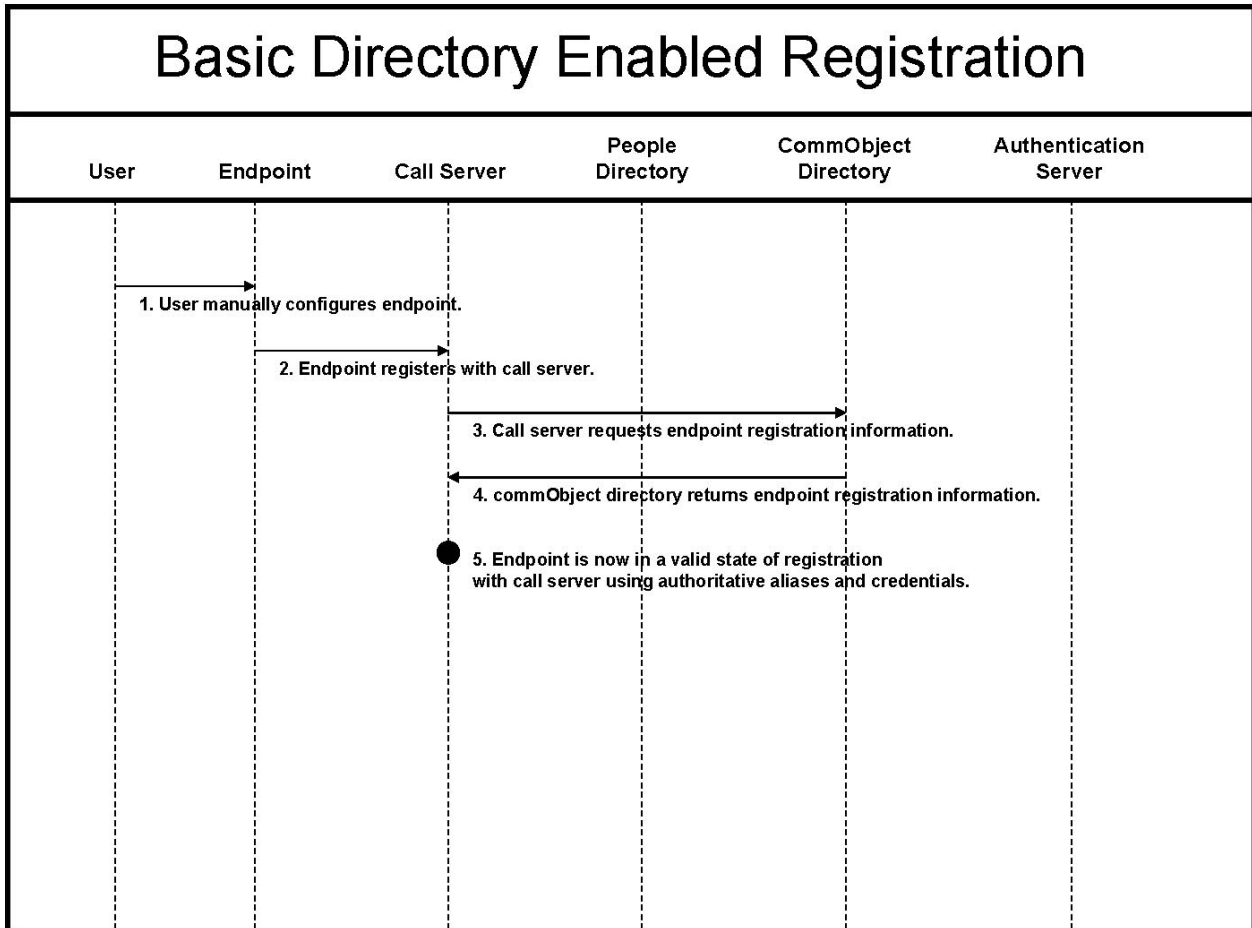


Figure A.2 Basic Directory Enabled Registration

II.3 Secure Single Sign-On

Figure A.3 illustrates how an environment can be created in which a user uses an existing enterprise authentication identity to access any of several endpoint identities. This scenario has several key features that are desirable for large scale deployments. The endpoint is automatically configured using information from the directory. This eliminates user error in the configuration process and simplifies deployment. Second, the user is only required to remember their single sign on credentials, which is typically their enterprise user ID and password. Endpoints can (and often should) have different credentials, but the user does not need to know them, because they are loaded directly into the endpoint from the directory. Because the endpoint credentials are loaded automatically, it is possible that these credentials could be frequently refreshed. For example, a management tool could generate random credentials for each endpoint and store them in the commObject directory each night. This creates a highly secure environment in which credentials can be very strong, and even if compromised are aged-out and recreated in a short period of time. This scenario supports the use of ID/password or certificate based endpoint credentials. Certificates have traditionally been found to be difficult to deploy because they are difficult for users to manage. This scenario solves the certificate management problem, and opens the possibility that certificates

can be managed by and on behalf of a central certificate management system, rather than on behalf of users, thus shielding users from the complexity of PKI while gaining its security advantages.

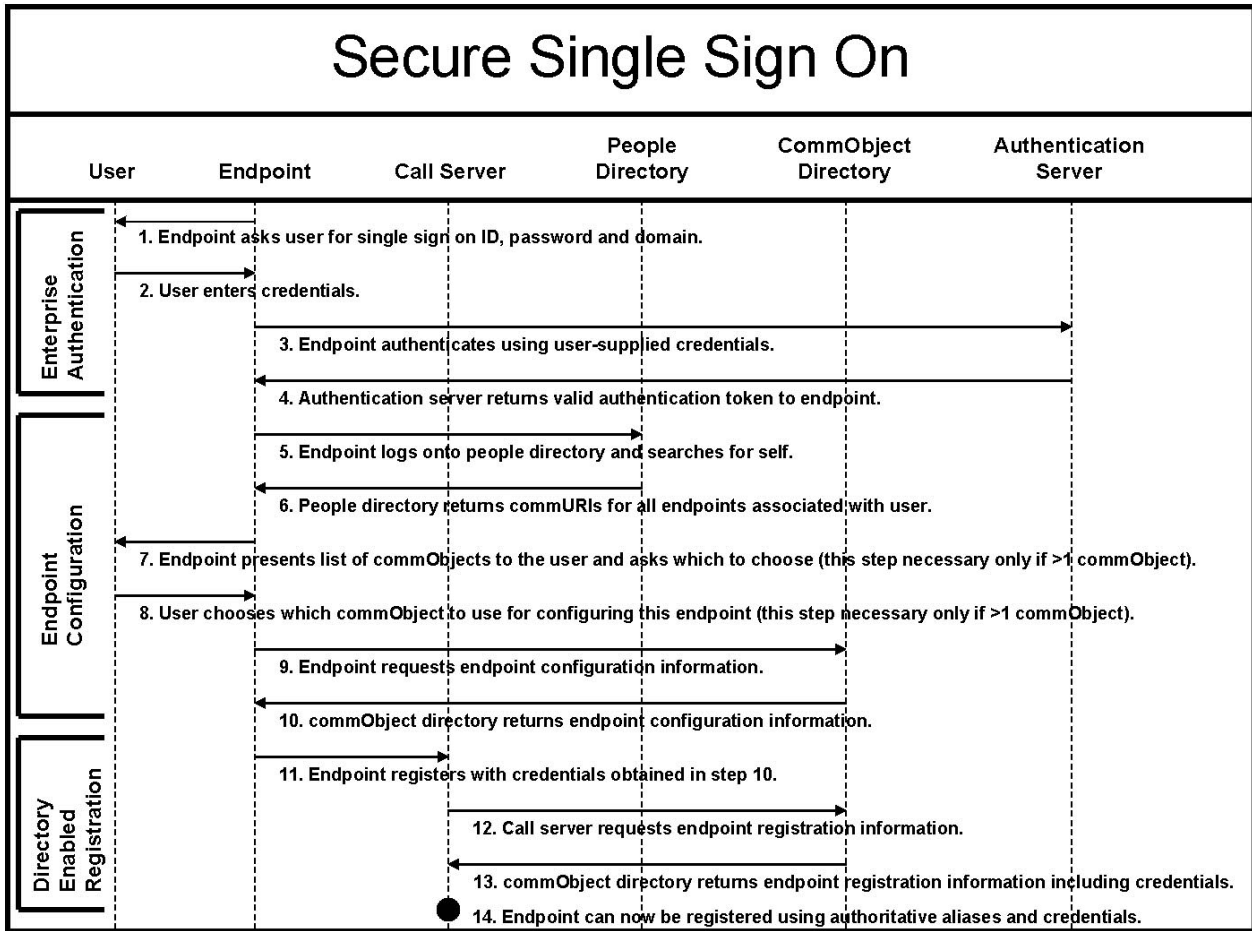


Figure A.3 Secure Single Sign-On

In this scenario, the enterprise authentication steps represent a user authenticating to an existing authentication server already deployed for general purpose (e.g. email, web, file sharing) single sign on authentication system. Once authenticated, the user can bind to the LDAP server directly and retrieve all configuration information for the selected endpoint, which includes configuration data and authentication credentials for the endpoint. Finally, using these credentials, the endpoint can authenticate to the call server using whichever authentication scheme is in place (for example, H.235 Annex D or E). Transport layer security should be used to ensure privacy of these transactions.

Appendix III Electronic Attachments

III Electronic Attachments

The attached file `commURI.ldif.txt` contains a text only version of the LDIF file described in section 8.1.



`commURI.ldif.txt`

The attached file `commObject.ldif.txt` contains a text only version of the LDIF file described in section 8.2.



`commObject.ldif.t
xt`
