ATNP WG3/15 WP 29

AERONAUTICAL TELECOMMUNICATIONS NETWORK PANEL

WORKING GROUP 3 (APPLICATIONS AND UPPER LAYERS)
Honolulu, USA
January 18 - 22, 1998

The Use of X.500 Protocols in ATM Data Link Technology: ATN Directory Approach

Prepared by: G. Saccone

SUMMARY

This paper outlines the approach to the content of SV7, the X.500 Directory.

1. Introduction

1.1 Need for a Directory Service

1.1.1 As operating concepts develop for the ATN, the notion of an application information server has been introduced. This server concept is mentioned in the CM guidance material, and is currently under investigation by various states and organizations. The server concept consists of a ground system being able to relay other facilities' application information to a requesting aircraft or ground system. This can lead to operational advantages, including more efficient use of bandwidth and fewer required connections.

1.1.2 These concepts revolve around access to an application information (i.e., application names and addresses) repository, either local or remote. There is not currently an ATN application that has this capability. Package 1 CM acts as the mechanism to actually exchange the necessary application information between an aircraft and ground system or two ground systems, but does not provide the capability to access an application information database. Ground-ground applications do not have an equivalent CM function; addressing is based on a priori knowledge of addresses. Since there is a distinct need for an application information database the concept of a directory service is introduced.

1.1.3 This repository is also necessary due in part to the complexity of the ATN naming and addressing scheme. Presently, FANS-1/A has a fairly simple naming and addressing scheme. The data link applications on board an aircraft are addressed by the aircraft flight ID. Likewise, a ground system's data link address is directly related to its four character facility designation. Individual applications are distinguished for both the aircraft and ground system by the use of an Imbedded Message Identifier (IMI) within an ACARS message header (such as "AD1" for ARINC Characteristic 745-1). Unlike FANS-1/A, the ATN is based on OSI protocols. End systems within an aircraft are identifed by naming corresponding to Subvolume IV and addressing corresponding to Subvolume V. This makes naming and addressing much less intuitive for the ATN as compared to FANS-1/A. Therefore, a priori knowledge is not easily known or calculated, so it would make sense to store this information in a data base.

1.1.4 Distributed Directory Systems

1.1.5 Having a centralized data base for information is a convenient way to access information. However, for large organizations this can become a daunting task. Different locations' information may change frequently, and keeping the information up to date can be difficult and incur a large overhead. A distributed directory system allows maintainence of information to be done by the local organization responsible for that information, while still making it available to the entire system. X.500 assumes that the information in the directory is distributed, and has provisions to handle this. For the ATN, this means that individual countries and organizations can manage their own data bases, while still giving the ATN community as a whole relevant information (according

to the access policies). In addition, contact information (such as a person in that country or organization, along with telephone number, for example) can also be conveyed.

1.2 X.500 Introduction

1.2.1 Directory Information Base

1.2.1.1 The CCITT X.500 series of documents (and their equivalent ISO standards, ISO 9594) are a published set of specifications representing a widely distributed (but logically centralized) information base of objects, such as users, systems, phone numbers, etc. The information held by the Directory is collectively termed the Directory Information Base (DIB). The DIB is made up of entries, each one of which describes a single object in the real world (for example, “person”). Each object has a group of features which describe that object called attributes. Each attribute in turn has a type and one or more values (for example, the entry for the object “person” may have a “telephone-number” attribute with one or more telephone number values). In addition, there may be one or more context values per attribute value. These context values are used to specify information which determines the applicability of that attribute (for example, a context would be used to tell how a time or date value should be interpreted). Figure 1a depicts these relationships.

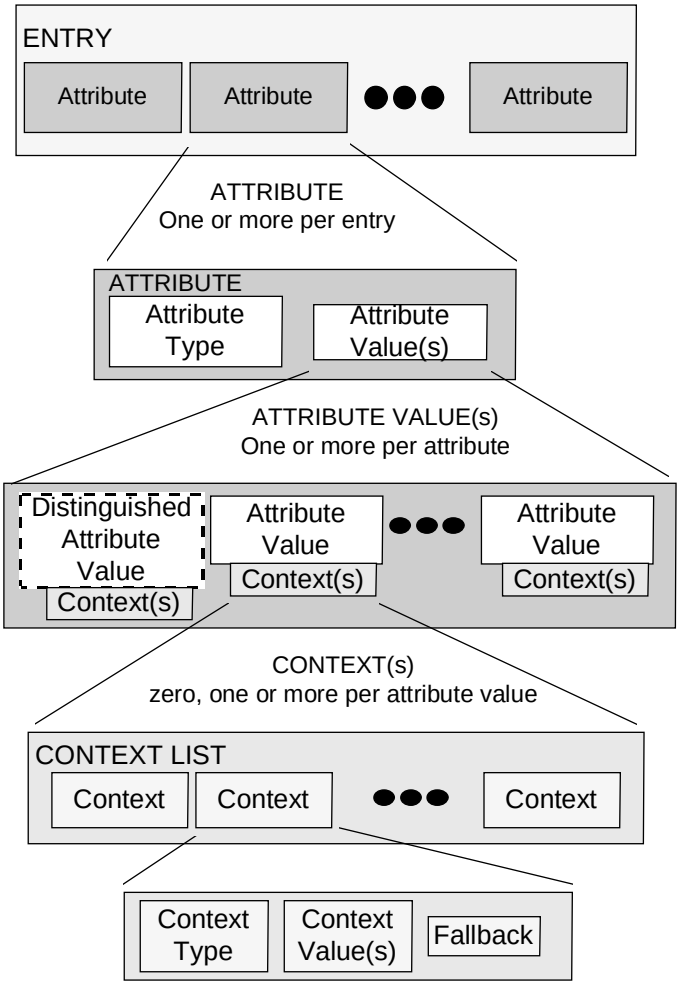


Figure 1a. DIB Entry Composition

1.2.1.2 The DIB is organized into a Directory Information Tree (DIT), which is based upon the hierarchies among the objects in the DIB (for instance, “person” works for an “organization” which is located within a “country”). A sample DIT is shown in Figure 1b.

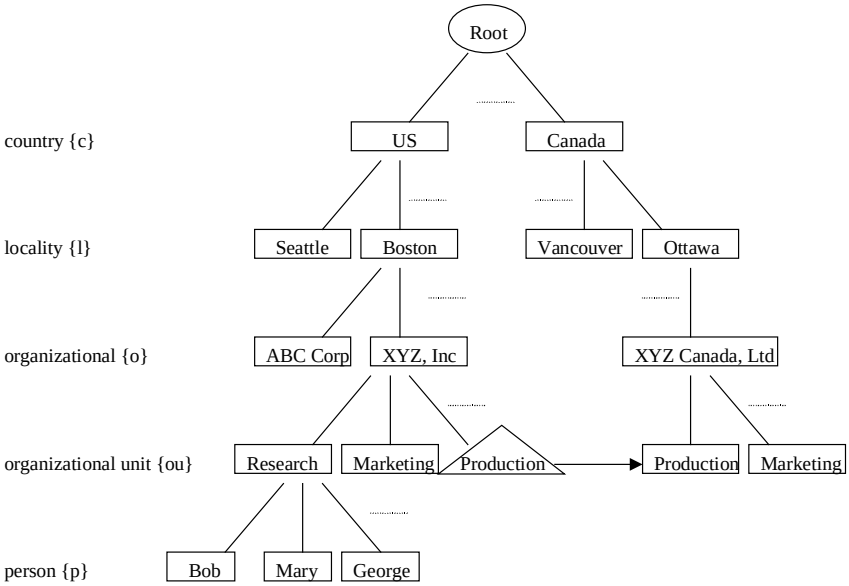


Figure 1b. Sample DIT

1.2.1.3 Each object has one immediate superior object, which is located immediately above the object in the DIT. An object may have subordinates, which are located immediately beneath it in the DIT. Additionally, an object may have an alias in the DIT, which is a pointer to an object entry under a different superior (see Figure 1).

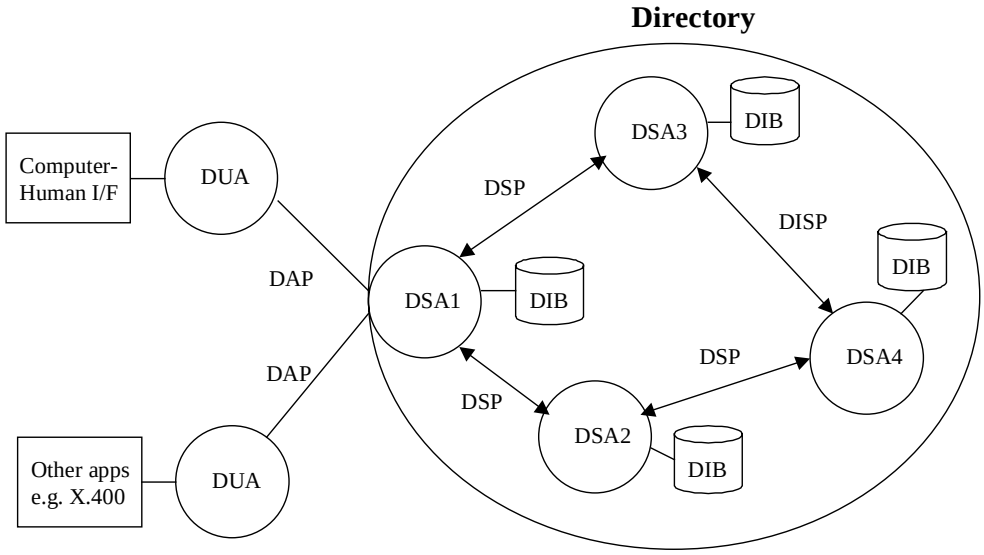
1.2.1.4 The DIT also provides the mechanism for the naming of objects in the DIB. Each entry in the Directory has a Relative Distinguished Name (RDN) that identifies an entry (in Figure 1, {country = "US"}, {organization = "XYZ, Inc"} and {locality = "Ottawa"} are all examples of RDNs). The sequence of RDNs all the way to a leaf, or end, node is a distinguished name (DN). The DN is a globally unique identifier for a directory object. In Figure 1, the DN for the research department of XYZ, Inc is {country = "US", locality = "Boston", organization = "XYZ, Inc", organizational unit = "Research"}. Note that the alias entry for {organizational unit = "Production"} is under the Canadian company, so there are two valid, but unique, DNs.

1.2.1.5 The Directory specifies a set of rules called the Directory schema which dictates the types and attributes valid for DIB entries. In addition, a Directory system schema dictates how operational information (e.g. create/modify timestamps, administrative roles, etc) is stored in the Directory.

1.2.2 X.500 Protocols and Components

1.2.2.1 There are four kinds of protocol associated with the Directory (1993 version): the Directory Access Protocol (DAP), Directory Systems Protocol (DSP), Directory Information Shadowing Protocol (DISP) and the Directory Operational Binding Protocol (DOP). These protocols provides the means for the various Directory agents—the Directory User Agent (DUA) and Directory Service Agent (DSA) to perform operations

on the DIB. Figures 2 and 3 show two different views of the Directory model, and are further explained below.



Note: DOP is not shown, but could take place between DSA pairs DSA1-DSA2, DSA1-DSA3, DSA2-DSA4

Figure 2. Directory Model

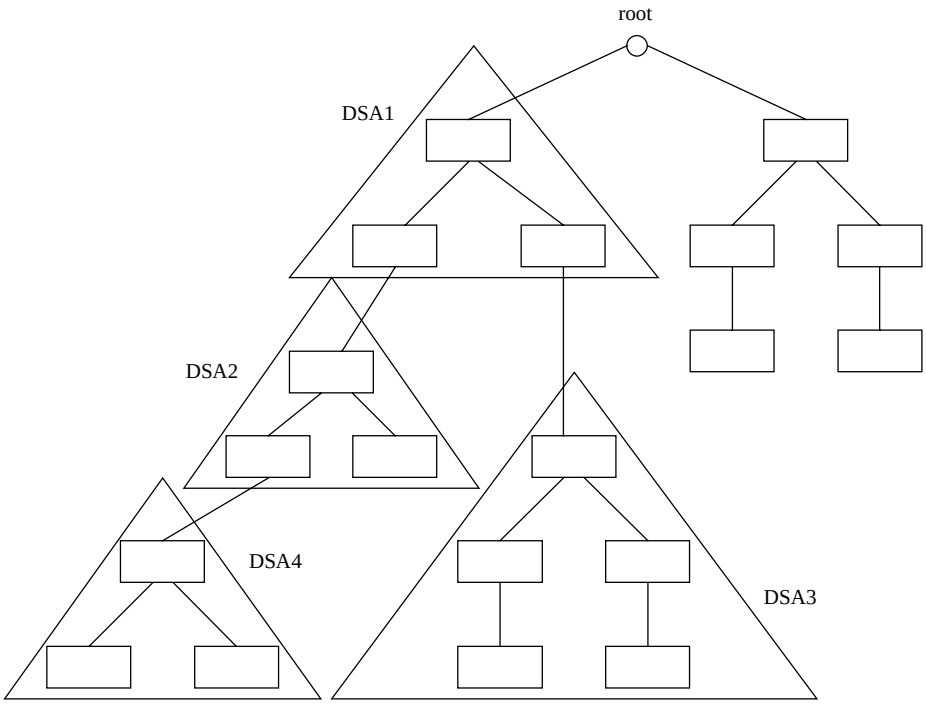


Figure 3. Directory Model Showing DSAs

1.2.2.2 Users access the Directory via the DUA. Note that the user can either be a human or an application. The DAP provides the means for the DUA to communication with the DSA. The DSA manages the information in the Directory. A directory is considered to

be distributed if there is more than one DSA. DSAs communicate with each other via the DSP (for distributed directory operations, e.g. search and read as directed by a DUA command) and DISP (for DIB replication transfers). The DOP defines the operational relationship (i.e. administrative agreements) between pairs of cooperating DSAs. Note that the DAP and DSP also make use of the Remote Operations Service Element (ROSE) and Association Control Service Element (ACSE) ASEs. Note that some actions, like the DSA interface the local DIB, is beyond the scope of the Directory.

1.2.2.3 The DUA has a number of operations it can perform with the DSA. These include read (read, compare, abandon), search (list, search) and modify (add entry, remove entry, modify entry, modify RDN) operations. The DSA, upon receipt of a query from the DUA, can perform basically basically the same operations, but are predicated with "chain" which implies that the DSA will pass on the request to as many DSAs as necessary in order to complete the request. There is also a "referral" response, which a DSA can return to either the requesting DSA or DUA. The referral response is used when the DSA has knowledge of the proper DSA to contact (i.e. name and address). The DSA or DUA that receives the referral response can then contact that DSA directly in order to carry out the operation.

1.2.2.4 The DOP has two types of bindings that can be established: the Hierarchical Operational Binding (HOB) and Shadow Operational Binding (SOB). The HOB governs the relationship bewteen a pair of DSAs, such as DSA1 and DSA2 in Figure 3. Some properties of the DIT governed by DSA1 will apply to the DIT governed by DSA2. The HOB provides a mechanism for the subordinate DSA (e.g. DSA2) to receive any administrative information from its superior DSA (e.g. DSA1). The SOB is responsible for setting up the binding necessary between two DSAs so that replication can take place. This typically involves determining the portion of the DIT to be copied as well as the supplier and consumer DSAs. Once this binding is known, then the DISP can be used. The DISP consists of three operations, Coordinate Shadow Update, Request Shadow Update, and Update Shadow. These operations all involve the replication of DIT information.

1.3 X.500 Applied to the ATN

1.3.1 In order to apply X.500 to the ATN, there are a number of design decisions that must first be made. These include what type of information the Directory will hold, how the information will be entered into the Directory, how the users will access the Directory, and who administers the information (and how it is administered) in the Directory? If these issues are not implemented consistently across the ATN Directory, then there will be serious problems in the operation, and the result will be a Directory that at best will only be useful to local users. These issues are described further in the following paragraphs.

1.3.2 ATN Information in the Directory

1.3.2.1 The ATN Directory will need to hold all information that have utility to the users of the ATN. In addition, there may be information that is only useful on a local basis, but needs to be included. This information is determined by the the DIT along with the corresponding object classes and attribute definitions for each node of the DIT. This information is described in Section 3 of this document.

1.3.3 ATN Directory Information Updates

1.3.3.1 Another aspect of the ATN Directory that needs standardization is how information is entered into the Directory itself. This will most likely be administered on a local level, but certain aspects of the information should be made available to all users, such as who entered or modified the data, the date and time of the modification, etc. This is discussed in Sections 4 and 5 of this document.

1.3.4 ATN Directory Access

1.3.4.1 The ATN Directory Access encompasses two main issues: First, how will the information from the Directory be retrieved by users. Information can be obtained both by other applications (e.g., an extension to CM to retrieve requested information in response to a logon request) or by users (e.g., a human requesting AIDC information for a specific area). This is discussed in Section 6 of this document. The second consideration is what kind of access controls (administrative and security) will need to be put in place to ensure that access to the Directory itself is only granted to properly authorized users. This is discussed in Sections 2 and 5 of this document.

1.3.5 ATN Directory Administration

1.3.5.1 The Directory is capable of supporting a wide range of administration models. To a large extent administration will be locally administered. However, there will need to minimum consistent strategy in order to ensure that changing information has been accounted for. The administrator of the Directory will also need to ensure that appropriate replication information is propagated to the appropriate users, taking into account performance and backup issues (i.e. handle the DISP parameters). This is discussed in Sections 2, 4 and 5 of this document.

1.3.5.2 Conceptual ATN Directory

1.3.5.3 The ATN Directory will look conceptually like Figure 4 below. Note that the DIBs in Figure 4 are ATN DIB fragments, and collectively make up the ATN DIB.

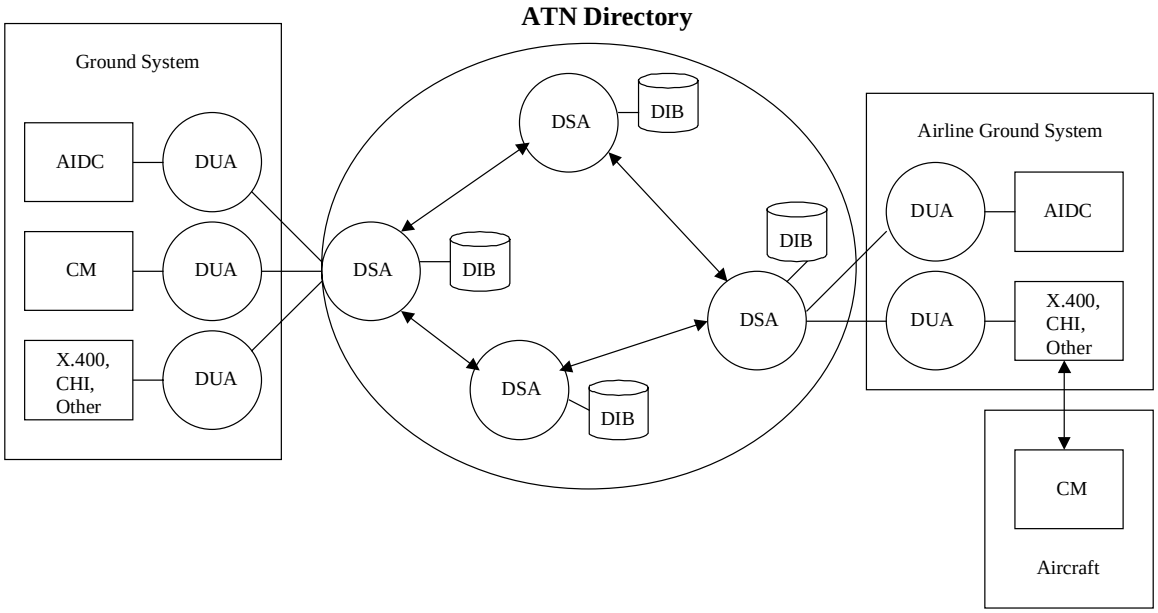


Figure 4. Conceptual View of ATN Directory

2. ATN X.500 Directory Information Tree

2.1 Information for Directory

2.1.1 In order to create a Directory for the ATN, the first step is to determine what kind of information needs to be part of the DIB, and how that relates to the DIT. As an initial cut, the information in Table 1 needs to be conveyed (note: this is in tabular form only; the object and attribute definitions will be fleshed out in a later section). This information should represent a minimal set of information in order to keep the ATN Directory relatively small and uncomplicated (i.e. a CAA's entire phone directory should not be included). Table 1 also proposes the Directory element which will hold this information. This will also be explained in later sections.

Table 1. Information Contained in the ATN Directory

Type of Information	Object Class	Description
Country	country	Name of country where ATN end systems will be employed
Organization	organization	Name of the CAA, Airline or Service Provider that will be responsible for the ATN end system. Note that aliases may be used so that an organization serving more than one country (e.g. Eurocontrol or SITA) may represented under each of the countries it serves.
Organizational Unit	organizational unit	This will allow for different departments of an organization to have its own ATN end systems. For example, certification or flight standards under a CAA.

Type of Information	Object Class	Description
Country	country	Name of country where ATN end systems will be employed
Locality	organization, organizational unit	Location of organizational unit (state, province, city, etc)
Location of End System – Facility Designation/24 bit ID (Application Process Title)	ATN End System	This gives the four to eight character facility designation, or alternately in the case of an aircraft end system, the 24 bit identifier.
Location of End System - Physical	ATN End System, ATN Application Entity	This will tell the physical location of an ATN end system. An example would be "CYVR, Annex"
Contact Person(s) and Titles	Individual domain subentries*	Personnel who are responsible for operation and maintenance of ATN end systems, including alternates
Contact Information	ATN End System	Gives the phone numbers, fax numbers, email address, X.400 addresses, AFTN addresses, etc of the contact persons.
Supported Application Information	ATN Application Entity	This includes the AE Qualifiers, application addresses, application version numbers, any build information (e.g. for CPDLC supported message subsets, CM server supported, etc), and subsetting details (chapter 2.x.8 of the SARPs).
Last Updated Information, Access control specifics	organizational unit, ATN End System, ATN Application Entity	Includes date, time and person/organization who last updated directory information. Also includes what kind of access and operation restrictions apply to data.
End System Service Area	organizational unit, ATN End System	Facility designations or ICAO regions that the identified end system serves
tbd – pariticularly g-g/X.400 needs		
tbd – X.509 stuff		

*Note – this means it is up to local implementation. This decision needs to be reviewed.

2.1.2 Once that all needed information is identified, that information must be placed as entries in the DIT. In order to do this, the entries must be broken down into object form,

with the appropriate object classes, attributes, and attribute syntaxes assigned. Currently, it is viewed that most of the ATN information necessary for inclusion in the Directory can fit into existing object classes and attributes as specified in X.521 and X.520, respectively. If need be, all of the required information can fit into the existing object classes and attributes. However, this would entail storing attribute values like version numbers as the same part of a text string as build information (for example). Therefore, in order to aid the search and storage characteristics (i.e. alleviate the need for extra sofware that has to parse large text strings) of the Directory, there will need to be some specialized object classes and attribute types specified. (Note: this is the author's opinion and may be revisited).

2.1.3 The object classes are country, organization, organizational-unit, application-process and application-entity. The application-process and application-entity object classes serve as the basis for two new object classes: aTNEndSystem and aTNApplicationEntity, respectively. These new classes will inherit the application-process and application-entity object classes characteristics. (Note: there are other ways to accomplish this without creating new object classes, such as allowing objects to belong to multiple classes. This decision may be revisited) The details of the attributes and their corresponding ATN data mapping are given in Section 3 of this document.

2.2 Directory Information Tree

2.2.1 Based on the ATN information and object classes identified, the DIT for the ATN Directory is depicted in Figure 5. Note that the application process and application entity object classes are used in the DIT, since they are the structural object classes for the newly created ones.

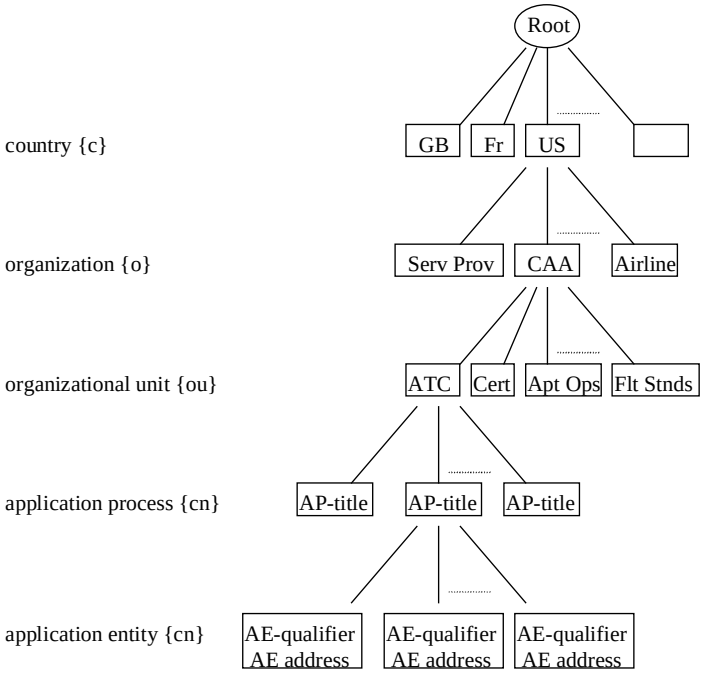


Figure 5. Proposed ATN Directory Information Tree

2.2.2 A question of where organizations such as service providers should be placed in the DIT was raised. Since service providers may not be limited to a particular country, it might be appropriate to place the service provider at a country level rather than an organization level (i.e. immediately under the root). This was not done for the ATN DIT for two reasons: one, it goes against the established naming and structuring guidelines for X.500, and two there are ways to accomplish the same virtual structure while still adhering to the X.500 guidelines.

2.2.3 The naming convention for X.500 does allow certain organizations to be at the level immediately under the root, but those organizations are international organizations, which are defined as an organization whose scope and remit covers many nations and has a quasi-governmental function. These organizations are known as "supra-national" organizations, an example of which would be the United Nations. Service providers fall under the "multi-national" organization category, which is defined as an organization which operates in more than one country, but is not considered governmental. However, the DIT shown in Figure 5 does not place restrictions on the use of aliases. These could be used under each country that the service providers administer to point to the respository for the information. This is depicted in Figure 6. Note that this example is meant to illustrate the flexibility allowed by the use of aliases; or course any user of the Directory is free to use or not use aliases as they see fit.

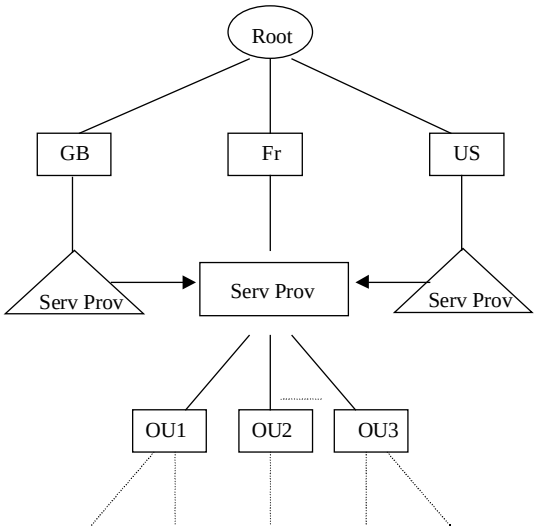


Figure 6. Sample Alias Scheme

2.3 Administrative Domains

2.3.1 The Directory standard allows for different parts of the DIT to be administered and managed by different authorities. This is the basis for a distributed DIB, and further allows individual countries and organizations to dictate some of the specifics of the administration of the DIT that applies to their own domains (this information is passed by the HOB). An Autonomous Adminstrative Area (AAA) defines a complete break between different administrators of the directory. For the ATN Directory, it is proposed that each organization under the country entry serve as an Autonomous Adminstrative Point (AAP), or the demarcation of an AAA. In addition to the AAA, another type of

administrative area is known as an Inner Administrative Area (IAA). An IAA is subordinate to an AAA, and represents a adminstrative area which is still ultimately under control of the AAA but may have some flexibility in defining its own administration fules. The start of an IAA is depicted by an Inner Administrative Point (IAP). The IAAs of a country are local to that country. For instance, an organization may choose to allow each of its organizational units to excerise some control of the adminstration, so each organizational unit entry would also be an IAP. This example is depicted in Figure 7. Note that although the DIT under the country XYZ is ATN compliant, the ATC inner adminstrative authority (under the CAA autonomous adminstrative authority) has also elected to add additional entries to the DIT, which are specific to ATC's needs. However, any entries added must conform to the hierarchical rule set forth in X.521 (i.e., a country can't be subordinate to a person).

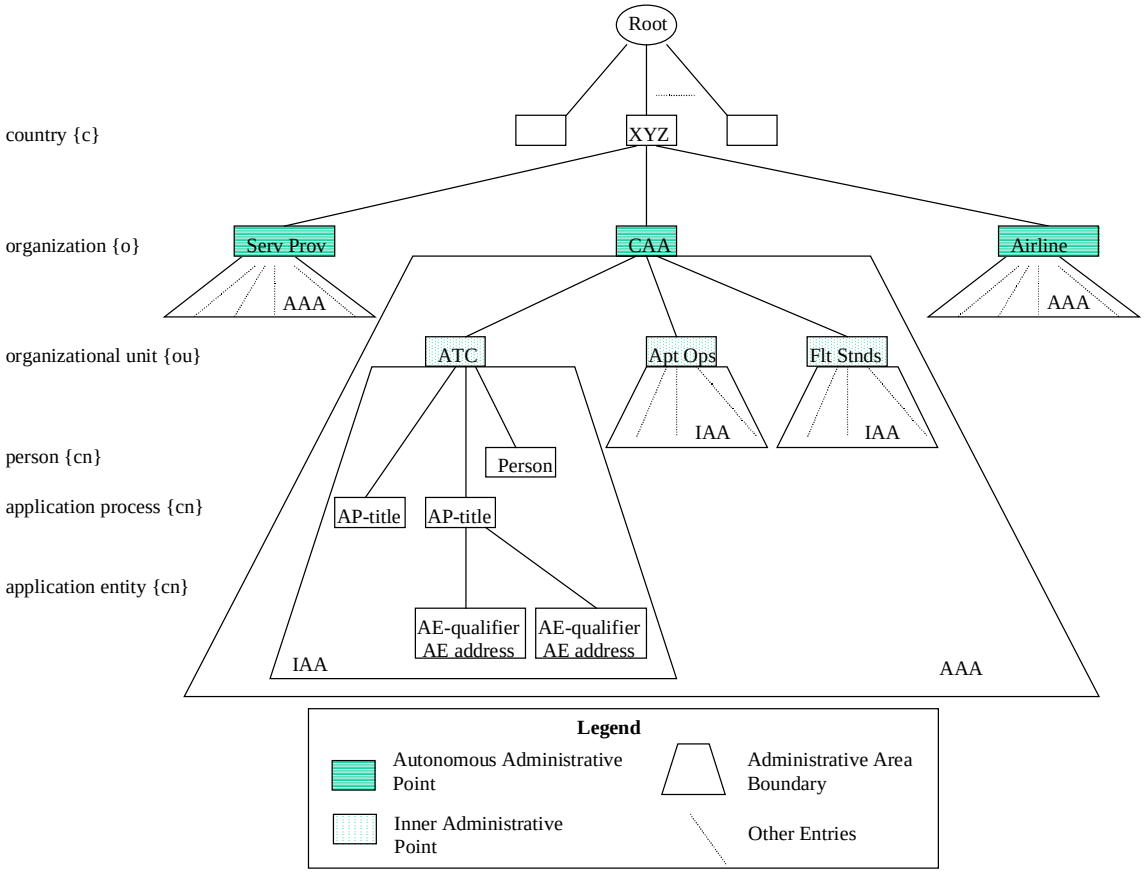


Figure 7. ATN Adminstrative Domains

2.3.2 An adminstrative authority can act in three different roles: subschema adminstration, access control adminstration and collective attribute adminstration. These roles must be coordinated throught the autonomous adminstrative authority. The roles are explained further in the following paragraphs. Their actual use by the ATN (i.e. representation and storage in the DIT) is defined in Section 4, ATN Directory System Schema.

2.3.3 The subschema administration dictates what can be added to the subschema; i.e. the rules for data entry for the portion of the DIT that is under the AAA control (there is

no concept of an IAA subschema, since this would imply that different data represented under the same administrative authority could be incompatible). Since the subschema has to be compliant with the overall Directory schema, an AAA is free to add additional entries as long as they are registered at the organizational level. The "person" entry in Figure 7 is such an example. Any information in a subschema which is critical to the operation of the ATN Directory should be promulgated to the rest of the ATN Directory users (in fact, if there such information it should probably become part of the ATN schema instead of a subschema).

2.3.4 The access control adminstration is concerned with administering a security policy that is enforced for a particular part of the DIT. The access control adminstration does not have be uniform across a whole organization. Taking the example in Figure 7, ATC may have very different security requirements than flight standards or airport operations. Therefore there could be three distince access control adminstrative areas. Note that access control administration does not have to follow a particular adminstrative domain; that is, the access control policy may be the same for ATC and airport operations (both under the same access control adminstration) and different flight standards (under its own access control adminstration).

2.3.5 Collective attribute dictates which collective attributes, or attributes that are assigned to each entry, will be present. An example would be a facsimilie phone number that is common to an entire adminstrative area. This phone number would then become a part of every entry. A collective attribute defined by an AAA will be present in all IAAs, but a collective attribute defined in an IAA will not be present in an AAA.

3. ATN X.500 Schema and Subschemas

3.1 Directory Schema Definition

3.1.1 The Directory Schema is a set of definitions and constraints concerning the structure of the DIT. This includes the possible ways entries are named, the information that can be held in an entry, the attributes used to represent that information, their organization into hierarchies to facilitate search and retrieval, and the ways in which values of attributes may be matched in attribute value and matching rule assertions. Thus, the schema is the rule book of the Directory, and ensures that a country name is not placed in an organizational entry or that an organization is not listed in a DIT as a subordinate to a person.

3.1.2 Formally, the Directory Schema comprises a set of:

a) *Name Form* definitions that define primitive naming relations for structural object classes,

b) *DIT Structure Rule* definitions that define the names that entries may have and the ways in which the the entries may be related to one another in the DIT,

c) *DIT Content Rule* definitions that extend the specification of allowable attributes for entries beyond those indicated by the structural object classes of the entries,

d) *Object Class* definitions that define the basic set of mandatory and optional attributes that shall be present, and may be present, respectively, in an entry of a given class, and which indicate the kind of object class that is being defined,

e) *Attribute Type* definitions that identify the object identifier by which an attribute is known, its syntax (specified in ASN.1), associated matching rules, whether it is an operational attribute and if so its type, whether it is a collective attribute, whether it is permitted to have multiple values and whether or not it is derived from another attribute type,

f) *Matching Rule* definitions that define matching rules, and

g) *DIT Context Use* definitions that govern the context types that may be associated with attribute values of any particular attribute type.

3.1.3 Figure 8 illustrates the relationships between schema and subschema definitions on the one side, and the DIT, directory entries, attributes, and attribute values on the other.

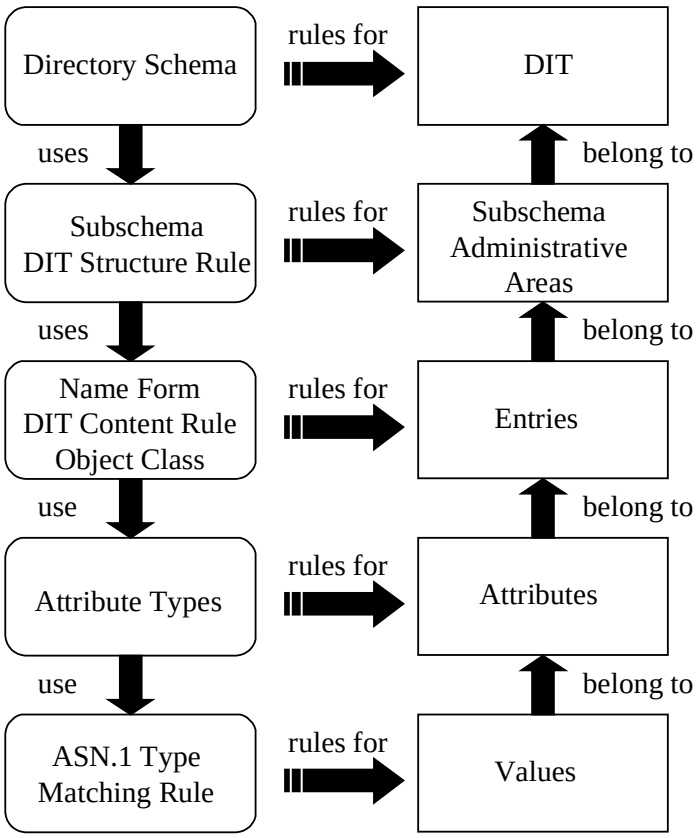


Figure 8. Directory Schema Relationships

3.1.4 The ATN Directory will mainly use X.521 and X.520 standard object types and attributes. Therefore, a list of the object classes and interpretation of their attributes and

attribute syntaxes is described next. For Subvolume 7, it is proposed not to re-copy all of the applicable object classes, attributes, and ASN.1, but merely to refer to the appropriate standards; it is reproduced here for the convenience of the reader. Any new object classes, attributes and attribute syntaxes will be defined in Subvolume 7.

3.2 Object Classes Used

3.2.1 The object classes proposed for use with the ATN Directory are listed below, by standard category.

3.2.2 X.500 Object Classes

3.2.2.1 The X.500 object classes and attributes are contained in X.521 and X.520, respectively. Their ASN.1 is included below, along with explanatory information on their usage for the ATN Directory. A Protocol Implementation Conformance Statement (PICS) will be completed for the ATN Directory. In the mean time, the notes below the object classes and attributes will explain intentions for ATN conformance (e.g., for Organization, the MAY CONTAIN { OrganizationalAttributeSet } is mandatory for the ATN; the attribute OrganizationalAttributeSet will have further restrictions on its elements, and so forth). The object classes are presented in hierarchical order as in the DIT.

3.2.2.2 Country

A *Country* object class is used to define country entries in the DIT.

```
country          OBJECT-CLASS          ::=       {
      SUBCLASS OF                              { top }
      MUST CONTAIN                { countryName }
      MAY CONTAIN                 { description | searchGuide }
      ID                                       id-oc-country }
```

3.2.2.3 Organization

The *Organization* object class is used to define organization entries in the DIT.

```
organization     OBJECT-CLASS          ::=       {
      SUBCLASS OF                              { top }
      MUST CONTAIN                { organizationName }
      MAY CONTAIN                 { OrganizationalAttributeSet }
      ID                                       id-oc-organization }
```

NOTE – The **OrganizationalAttributeSet** attribute is mandatory for the ATN.

3.2.2.4 Organizational Unit

The *Organizational Unit* object class is used to define entries representing subdivisions of organizations.

```
organizationalUnit              OBJECT-CLASS ::=    {
    SUBCLASS OF                 { top }
    MUST CONTAIN            { organizationalUnitName }
    MAY CONTAIN             { OrganizationalAttributeSet }
    ID                          id-oc-organiationalUnit }
```

3.2.2.5 Application Process

The *Application Process* object class is used to define entries representing application processes. *Application Process* is required to be superior to *Application Entity* by the hierarchy rules of X.521, Appendix B. For the ATN, *Application Process* is used to convey the AP-title and any relevant physical location information pertaining to an ATN end system.

```
applicationProcess              OBJECT-CLASS        ::=     {
    SUBCLASS OF                 { top }
    MUST CONTAIN            { commonName }
    MAY CONTAIN             { description |
                                localityName |
                                organizationalUnitName |
                                seeAlso }
    ID                          id-oc-applicationProcess }
```

NOTE – The **commonName** attribute will contain the AP-Title for the ATN application.

NOTE – The **localityName** and **organizationalUnitName** attributes will identify where the ATN end system is located.

3.2.2.6 Application Entity

The *Application Entity* object class is used to define entries representing application entities. For the ATN, this includes the application entity details of the individual ATN applications.

```
applicationEntity           OBJECT-CLASS        ::=     {
    SUBCLASS OF                 { top }
    MUST CONTAIN            { commonName | presentationAddress }
    MAY CONTAIN             { description |
                                localityName |
                                organizationName |
                                organizationalUnitName |
                                seeAlso |
                                supportedApplicationContext }
    ID                          id-oc-applicationEntity }
```

NOTE – If an application-entity is represented as a Directory object that is distinct from an application-process, the **commonName** attribute is used to carry the value of the Application Entity Qualifier.

NOTE – The **presentationAddress** attribute is used to carry the value of the ATN application's (as identified in the Application Entity Qualifier) address.

NOTE – The **localityName** and **organizationalUnitName** attributes will identify where the ATN end system is located.

NOTE – The **supportedApplicationContext** attribute will contain application context as specifed in Subvolume 4.

NOTE – The **description** attribute may describe the type of AE, subset number choice from the set specified in SARPs, and any other special indications such as “CM server functionality” or “CPDLC Build 1 message set supported”.

Tbd – X.509 classes

3.2.3 X.400 Object Classes

3.2.3.1 TBD

3.2.4 ATN-Specific Object Classes

3.2.4.1 ATN-Specific object classes are classes that have been created to meet unique ATN requirements.

3.2.4.2 ATN End System

The *aTNEndSystem* object class is used to define entries representing ATN systems. This object class inherits properties from the Application Process object class, and includes additional attribute sets in order to provide contact information (telecommunication and postal) for the facilities where the ATN end systems are actually located.

```
aTNEndSystem            OBJECT-CLASS        ::=     {
    SUBCLASS OF             { applicationProcess }
    MAY CONTAIN             { description |
                              localeAttributeSet |
                              telecommunicationAttributeSet }
    ID                      id-oc-aTNEndSystem }
```

3.2.4.3 ATN Application Entity

The *aTNApplicationEntity* object class is used to define entries representing ATN application entities, i.e. application details. This object class inherits properties from the Application Entity object class, and includes additional attribute sets in order to provide application version numbers as well as any helpful descriptive information (e.g., local software version number, which processor this AE resides on, etc).

```
aTNApplicationEntity            OBJECT-CLASS        ::=     {
    SUBCLASS OF                 { applicationEntity }

    MUST CONTAIN            { versionNumber }
    MAY CONTAIN             { description }
    ID                      id-oc-aTNApplicationEntity }
```

3.2.4.4 More TBD

3.3 Attribute Types Used

3.3.1 The attribute types proposed for use with the ATN Directory are listed below, by standard category. The attribute types are listed in alphabetical order. Attribute sets (groups of attribute types) are also included.

3.3.2 X.500 Standard Attributes and Attribute Sets

3.3.2.1 Business Category

The *Business Category* attribute type specifies information concerning the occupation of some common objects, e.g. people. For example, this attribute provides the facility to interrogate the Directory about people sharing the same occupation.

```
businessCategory ATTRIBUTE    ::=    {
    WITH SYNTAX                      DirectoryString {ub-business-category}
    EQUALITY MATCHING RULE           caseIgnoreMatch
    SUBSTRINGS MATCHING RULE         caseIgnoreSubstringsMatch
    ID                               id-at-businessCategory }
```

3.3.2.2 Common Name

The *Common Name* attribute type specifies an identifier of an object. A Common Name is not a directory name; it is a (possibly ambiguous) name by which the object is commonly known in some limited scope (such as an organization) and conforms to the naming conventions of the country or culture with which it is associated.

An attribute value for common name is a string chosen either by the person or organization it describes or the organization responsible for the object it describes for devices and application entities. For example, a typical name of a person in an English-speaking country comprises a personal title (e.g. Mr., Ms, Rd, Professor, Sir, Lord), a first name, middle name(s), last name, generation qualifier (if any, e.g. Jr.) and decorations and awards (if any, e.g. QC).

Examples:

CN = “Mr. Robin Lachlan McLeod BSc(Hons) CEng MIEE”;

CN = “Divisional Coordination Committee”;

CN = “High Speed Modem”.

Any variants should be associated with the named object as separate and alternative attribute values.

Other common variants should also be admitted, e.g. use of a middle name as a preferred first name; use of “Bill” in place of “William”, etc.

```
commonName ATTRIBUTE    ::=    {
```

```
    SUBTYPE OF                  name
    WITH SYNTAX                 DirectoryString {ub-common-name}
    ID                          id-at-commonName }
```

3.3.2.3 Country Name

The *Country Name* attribute type specifies a country. When used as a component of a directory name, it identifies the country in which the named object is physically located or with which it is associated in some other important way.

An attribute value for country name is a string chosen from ISO 3166.

```
countryName ATTRIBUTE ::= {
        SUBTYPE OF          name
        WITH SYNTAX         CountryName
        SINGLE VALUE        TRUE
        ID                  id-at-countryName }
```

3.3.2.4 Description

The *Description* attribute type specifies text which describes the associated object.
For example, the object “Standards Interest” might have the associated description “distribution list for exchange of information about intra-company standards development”.

An attribute value for Description is a string.

```
description ATTRIBUTE           ::=     {
    WITH SYNTAX                         DirectoryString {ub-description}
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
    ID                                  id-at-description }
```

3.3.2.5 Destination Indicator

The *Destination Indicator* attribute type specifies (according to CCITT Recommendation F.1 and CCITT Recom-mendation F.31) the country and city associated with the object (the addressee) needed to provide the Public Telegram Service.

An attribute value for Destination Indicator is a string.

```
destinationIndicator ATTRIBUTE ::= {
        WITH SYNTAX                 DestinationIndicator
        EQUALITY MATCHING RULE      caseIgnoreMatch
        SUBSTRINGS MATCHING RULE    caseIgnoreSubstringsMatch
        ID                          id-at-destinationIndicator }
```

3.3.2.6 Facsimile Telephone Number

The *Facsimile Telephone Number* attribute type specifies a telephone number for a facsimile terminal (and optionally its parameters) associated with an object.

An attribute value for the facsimile telephone number is a string that complies with the internationally agreed format for showing international telephone numbers, CCITT Recommendation E.123 (e.g. “+81 3 347 7418”) and an optional bit string (formatted according to CCITT Recommendation T.30).

```
facsimileTelephoneNumber ATTRIBUTE ::= {
        WITH SYNTAX         FacsimileTelephoneNumber
        ID                  id-at-facsimileTelephoneNumber }
```

3.3.2.7 International ISDN Number

The *International ISDN Number* attribute type specifies an International ISDN Number associated with an object.

An attribute value for International ISDN Number is a string which complies with the internationally agreed format for ISDN addresses given in CCITT Recommendation E.164.

```
internationalISDNNumber ATTRIBUTE ::= {
        WITH SYNTAX                     InternationalISDNNumber
        EQUALITY MATCHING RULE          numericStringMatch
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch
        ID                              id-at-internationalISDNNumber }
```

3.3.2.8 Locale Attribute Set

This set of attributes is used to define those which are commonly used for search purposes to indicate the locale of an object.

```
LocaleAttributeSet          ATTRIBUTE  ::= {
    localityName |
    stateOrProvinceName |
    streetAddress }
```

3.3.2.9 Locality Name

The *Locality Name* attribute type specifies a locality. When used as a component of a directory name, it identifies a geographical area or locality in which the named object is physically located or with which it is associated in some other important way.
An attribute value for Locality Name is a string, e.g. L = “Edinburgh”.

```
localityName ATTRIBUTE          ::=     {
    SUBTYPE OF                          name
    WITH SYNTAX                         DirectoryString {ub-locality-name}
    ID                                  id-at-localityName }
```

3.3.2.10 OrganizationName

The *OrganizationName* attribute type specifies an organization. When used as a component of a directory name it identifies an organization with which the named object is affiliated.

An attribute value for OrganizationName is a string chosen by the organization (e.g. O = “Scottish Telecom-munications plc”). Any variants should be associated with the named Organization as separate and alternative attribute values.

```
organizationName ATTRIBUTE    ::=    {
    SUBTYPE OF                       name
    WITH SYNTAX                      DirectoryString {ub-organization-name}
    ID                               id-at-organizationName }
```

3.3.2.11 Organizational Attribute Set

This set of attributes is used to define the attributes that an organization or organizational unit may typically possess.

```
OrganizationalAttributeSet                ATTRIBUTE  ::= {
   description |
   LocaleAttributeSet |
   PostalAttributeSet |
   TelecommunicationAttributeSet |
   businessCategory |
   seeAlso |
   searchGuide |
   userPassword }
```

NOTE – The **userPassword** attribute is defined by X.509.

3.3.2.12 Organizational Unit Name

The *Organizational Unit Name* attribute type specifies an organizational unit. When used as a component of a directory name it identifies an organizational unit with which the named object is affiliated.

The designated organizational unit is understood to be part of an organization designated by an OrganizationName attribute. It follows that if an Organizational Unit Name attribute is used in a directory name, it must be associated with an OrganizationName attribute.

An attribute value for Organizational Unit Name is a string chosen by the organization of which it is part (e.g. OU = “Technology Division”). Note that the commonly used abbreviation “TD” would be a separate and alternative attribute value.

Example:

O = “Scottel”, OU = “TD”

```
organizationalUnitName ATTRIBUTE      ::=      {
    SUBTYPE OF                              name
    WITH SYNTAX                             DirectoryString {ub-organizational-unit-name}
    ID                                      id-at-organizationalUnitName }
```

3.3.2.13 Physical Delivery Office Name

The *Physical Delivery Office Name* attribute type specifies the name of the city, village, etc. where a physical delivery office is situated.

An attribute value for Physical Delivery Office Name is a string.

```
physicalDeliveryOfficeName ATTRIBUTE    ::=     {
    WITH SYNTAX                         DirectoryString {ub-physical-office-name}
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
    ID                                  id-at-physicalDeliveryOfficeName }
```

3.3.2.14 Postal Address

The *Postal Address* attribute type specifies the address information required for the physical delivery of postal messages by the postal authority to the named object.

An attribute value for Postal Address will be typically composed of selected attributes from the MHS Unformatted Postal O/R Address version 1 according to CCITT Recommendation F.401 and limited to 6 lines of 30 characters each, including a Postal Country Name. Normally the information contained in such an address could include an addressee's name, street address, city, state or province, postal code and possibly a Post Office Box number depending on the specific requirements of the named object.

```
postalAddress ATTRIBUTE         ::=     {
    WITH SYNTAX                         PostalAddress
    EQUALITY MATCHING RULE              caseIgnoreListMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreListSubstringsMatch
    ID                                  id-at-postalAddress }
```

3.3.2.15 Postal Attribute Set

This set of attributes is used to define those which are directly associated with postal delivery.

```
PostalAttributeSet ATTRIBUTE  ::= {
    physicalDeliveryOfficeName |
    postalAddress |
    postalCode |
    postOfficeBox |
    streetAddress }
```

3.3.2.16 Postal Code

The *Postal Code* attribute type specifies the postal code of the named object. If this attribute value is present it will be part of the object's postal address.

An attribute value for Postal Code is a string.

```
postalCode ATTRIBUTE            ::=     {
    WITH SYNTAX                         DirectoryString {ub-postal-code}
    EQUALITY MATCHING RULE              caseIgnoreMatch
    SUBSTRINGS MATCHING RULE            caseIgnoreSubstringsMatch
    ID                                  id-at-postalCode }
```

3.3.2.17 Post Office Box

The *Post Office Box* attribute type specifies the Post Office Box by which the object will receive physical postal delivery. If present, the attribute value is part of the object's postal address.

```
postOfficeBox ATTRIBUTE        ::=    {
    WITH SYNTAX                       DirectoryString {ub-post-office-box}
    EQUALITY MATCHING RULE            caseIgnoreMatch
    SUBSTRINGS MATCHING RULE          caseIgnoreSubstringsMatch
    ID                                id-at-postOfficeBox }
```

3.3.2.18 Preferred Delivery Method

The *Preferred Delivery Method* attribute type specifies the object's priority order regarding the method to be used for communicating with it.

```
preferredDeliveryMethod ATTRIBUTE      ::= {
    WITH SYNTAX                          SEQUENCE OF INTEGER {
        ny-delivery-method                       (0),
        mhs-delivery                             (1),
        physical-delivery                        (2),
        telex-delivery                           (3),
        teletex-delivery                         (4),
        g3-facsimile-delivery                    (5),
        g4-facsimile-delivery                    (6),
        ia5-terminal-delivery                    (7),
        videotex-delivery                        (8),
        telephone-delivery                       (9) }
    SINGLE VALUE                         TRUE
    ID                                   id-at-preferredDeliveryMethod }
```

3.3.2.19 Presentation Address

The *Presentation Address* attribute type specifies a presentation address associated with an object representing an OSI application entity.

An attribute value for Presentation Address is a presentation address as defined in ISO 7498.

```
presentationAddress ATTRIBUTE::=       {
    WITH SYNTAX                        PresentationAddress
    EQUALITY MATCHING RULE             presentationAddressMatch
    SINGLE VALUE                       TRUE
    ID                                 id-at-presentationAddress }
```

3.3.2.20 Registered Address

The *Registered Address* attribute type specifies a mnemonic for an address associated with an object at a particular city location. The mnemonic is registered in the country in which the city is located and is used in the provision of the Public Telegram Service (according to CCITT Recommendation F.1).

```
registeredAddress ATTRIBUTE    ::=     {
```

```
    SUBTYPE OF                  postalAddress
    WITH SYNTAX                 PostalAddress
    ID                          id-at-registeredAddress }
```

3.3.2.21 Search Guide

The *Search Guide* attribute type specifies information of suggested search criteria which may be included in some entries expected to be a convenient base-object for the search operation, e.g. country or organization.

Search criteria consist of an optional identifier for the type of object sought and combinations of attribute types and logical operators to be used in the construction of a filter. It is possible to specify for each search criteria item the matching level, e.g. approximate match.

The Search Guide attribute may recur to reflect the various types of requests, e.g. search for a Residential Person or an Organizational Person, which may be fulfilled from the given base-object where the Search Guide is read.

```
searchGuide ATTRIBUTE                   ::=     {
    WITH SYNTAX                                 Guide
    ID                                          id-at-searchGuide }

Guide                           ::=     SET  {
    objectClass                 [0]     OBJECT-CLASS.&id OPTIONAL,
    criteria                    [1]     Criteria  }

Criteria                        ::=     CHOICE {
    type                        [0]     CriteriaItem,
    and                         [1]     SET OF Criteria,
    or                          [2]     SET OF Criteria,
    not                         [3]     Criteria }

CriteriaItem                    ::=     CHOICE {
    equality                    [0]     AttributeType,
    substrings                  [1]     AttributeType,
    greaterOrEqual              [2]     AttributeType,
    lessOrEqual                 [3]     AttributeType,
    approximateMatch            [4]     AttributeType }
```

Example:
The following is a potential value of the Search Guide attribute that could be stored in entries of object class Locality to indicate how entries of object class Residential Person might be found:

```
residential-person-guide Guide ::= {
    objectClass residentialPerson.&id,
    criteria and : {
        type : substrings : commonName.&id,
        type : substrings : streetAddress.&id }}
```

The construction of a filter from this value of Guide is straightforward.
Step (1) produces the intermediate Filter value

```
intermediate-filter Filter              ::=
    and : {
        item : substrings  {
            type  commonName.&id,
            strings { any : teletexString : ìDuboisî }},
```

```
        item : substrings  {
            type  streetAddress.&id,
            strings { any : teletexString  ìHugoî }}}
```

Step (2) produces a filter for matching Residential Person entries in the subtree:

```
residential-person-filter Filter ::=
    and : {
        item : equality : {
            type  objectClass.&id,
            assertion  residentialPerson.&id },
        intermediateFilter }
```

3.3.2.22 See Also

The *See Also* attribute type specifies names of other Directory objects which may be other aspects (in some sense) of the same real world object.

An attribute value for See Also is a distinguished name.

```
seeAlso ATTRIBUTE              ::=     {
    SUBTYPE OF                         distinguishedName
    ID                                 id-at-seeAlso }
```

3.3.2.23 State or Province Name

The *State or Province Name* attribute type specifies a state or province. When used as a component of a directory name, it identifies a geographical subdivision in which the named object is physically located or with which it is associated in some other important way.

An attribute value for State or Province Name is a string, e.g. S = “Ohio”.

```
stateOrProvinceName ATTRIBUTE          ::=     {
    SUBTYPE OF                                 name
    WITH SYNTAX                                DirectoryString {ub-state-name}
    ID                                         id-at-stateOrProvinceName }
```

3.3.2.24 Street Address

The *Street Address* attribute type specifies a site for the local distribution and physical delivery in a postal address, i.e. the street name, place, avenue, and the house number. When used as a component of a directory name, it identifies the street address at which the named object is located or with which it is associated in some other important way.

An attribute value for Street Address is a string, e.g. “Arnulfstraße 60”.

```
streetAddress ATTRIBUTE          ::=     {
    WITH SYNTAX                          DirectoryString {ub-street-address}
    EQUALITY MATCHING RULE               caseIgnoreMatch
    SUBSTRINGS MATCHING RULE             caseIgnoreSubstringsMatch
    ID                                   id-at-streetAddress }
```

3.3.2.25 Supported Application Context

The *Supported Application Context* attribute type specifies the object identifier(s) of application context(s) that the object (an OSI application entity) supports.

```
supportedApplicationContext ATTRIBUTE  ::=        {
    WITH SYNTAX                        OBJECT IDENTIFIER
    EQUALITY MATCHING RULE             objectIdentifierMatch
    ID                                 id-at-supportedApplicationContext }
```

3.3.2.26 Telecommunication Attribute Set

This set of attributes is used to define those which are commonly used for business communications.

```
TelecommunicationAttributeSet     ATTRIBUTE  ::= {
        facsimileTelephoneNumber |
        internationalISDNNumber |
        telephoneNumber |
        teletexTerminalIdentifier |
        telexNumber |
        preferredDeliveryMethod |
        destinationIndicator |
        registeredAddress |
        x121Address }
```

3.3.2.27 Telephone Number

The *Telephone Number* attribute type specifies a telephone number associated with an object.

An attribute value for Telephone Number is a string that complies with the internationally agreed format for showing international telephone numbers, CCITT Recommendation E.123 (e.g. "+ 44 582 10101").

```
telephoneNumber ATTRIBUTE ::= {
        WITH SYNTAX                    TelephoneNumber
        EQUALITY MATCHING RULE         telephoneNumberMatch
        SUBSTRINGS MATCHING RULE       telephoneNumberSubstringsMatch
        ID                             id-at-telephoneNumber }
```

3.3.2.28 Teletex Terminal Identifier

The *Teletex Terminal Identifier* attribute type specifies the Teletex terminal identifier (and, optionally, parameters) for a teletex terminal associated with an object.

An attribute value for Teletex Terminal Identifier is a string which complies with CCITT Recommendation F.200 and an optional set whose components are according to CCITT Recommendation T.62.

```
teletexTerminalIdentifier ATTRIBUTE        ::=        {
    WITH SYNTAX                            TeletexTerminalIdentifier
    ID                                     id-at-teletexTerminalIdentifier }
```

3.3.2.29 Telex Number

The *Telex Number* attribute type specifies the telex number, country code, and answerback code of a telex terminal associated with an object.

```
telexNumber ATTRIBUTE        ::=    {
    WITH SYNTAX                     TelexNumber
    ID                              id-at-telexNumber }
```

3.3.2.30 X.121 Address

The *X.121 Address* attribute type specifies an address as defined by CCITT Recommendation X.121 associated with an object.

```
x121Address ATTRIBUTE ::= {
        WITH SYNTAX                     X121Address
        EQUALITY MATCHING RULE          numericStringMatch
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch
        ID                              id-at-x121Address }
```

3.3.3 X.400 Attributes and Attribute Sets

3.3.4 TBD

3.3.5 ATN-Specific Attributes and Attribute Sets

3.3.5.1 The attribute types that are ATN specific are given below.

3.3.5.2 Version Number

The *verionNumber* attribute type specifies an ATN application's version number.

```
versionNumber ATTRIBUTE ::= {
        WITH SYNTAX                     VersionNumber
        EQUALITY MATCHING RULE          numericStringMatch
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch
        ID                              id-at-versionNumber }
```

3.3.5.3 More TBD (for example, AFTN address? Email address?)

3.4 Attribute Syntaxes Used

3.4.1 The attribute syntaxes proposed for use with the ATN Directory are listed below, by standard category. The attribute syntaxes are grouped by function as applicable, and listed in alphabetical order.

3.4.1.1 X.500 Attribute Syntaxes

```
--general

countryName ATTRIBUTE ::=     {
        SUBTYPE OF              name
        WITH SYNTAX                     PrintableString (SIZE (2)) -- IS 3166 codes only
        SINGLE VALUE                    TRUE
        ID                              id-at-countryName }

destinationIndicator ATTRIBUTE ::=     {
        WITH SYNTAX                     PrintableString (SIZE (1..ub-destination-indicator))
                                        -- alphabetical characters only
        EQUALITY MATCHING RULE          caseIgnoreMatch
        SUBSTRINGS MATCHING RULE        caseIgnoreSubstringsMatch
        ID                              id-at-destinationIndicator }

DirectoryString { INTEGER : maxSize } ::= CHOICE {
        teletexString           TeletexString (SIZE (1..maxSize)),
        printableString         PrintableString (SIZE (1..maxSize)),
        universalString         UniversalString (SIZE (1..maxSize)) }

distinguishedName ATTRIBUTE  ::=     {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        ID                              id-at-distinguishedName }

FacsimileTelephoneNumber        ::=     SEQUENCE {
        telephoneNumber         PrintableString (SIZE(1.. ub-telephone-number)),
        parameters              G3FacsimileNonBasicParameters  OPTIONAL}

internationalISDNNumber ATTRIBUTE  ::=     {
        WITH SYNTAX                     NumericString (SIZE (1..ub-international-isdn-number))
        EQUALITY MATCHING RULE          numericStringMatch
        SUBSTRINGS MATCHING RULE        numericStringSubstringsMatch
        ID                              id-at-internationalISDNNumber }

name ATTRIBUTE          ::=     {
        WITH SYNTAX                     DirectoryString { ub-name }
        EQUALITY MATCHING RULE          caseIgnoreMatch
        SUBSTRINGS MATCHING RULE        caseIgnoreSubstringsMatch
        ID                              id-at-name }

PostalAddress           ::=     SEQUENCE SIZE(1..ub-postal-line) OF  DirectoryString {ub-postal-
string}

PresentationAddress ::= SEQUENCE {
        pSelector       [0]     OCTET STRING OPTIONAL,
        sSelector       [1]     OCTET STRING OPTIONAL,
        tSelector       [2]     OCTET STRING OPTIONAL,
        nAddresses      [3]     SET SIZE (1..MAX)  OF  OCTET STRING}

telephoneNumber ATTRIBUTE   ::=     {
        WITH SYNTAX                     PrintableString (SIZE (1..ub-telephone-number))
```

```
        EQUALITY MATCHING RULE      telephoneNumberMatch
        SUBSTRINGS MATCHING RULE    telephoneNumberSubstringsMatch
        ID                          id-at-telephoneNumber }

TelexNumber ::= SEQUENCE {
        telexNumber         PrintableString (SIZE(1..ub-telex-number)),
        countryCode         PrintableString (SIZE(1..ub-country-code)),
        answerback                  PrintableString (SIZE(1..ub-answerback))}

TeletexTerminalIdentifier ::= SEQUENCE {
        teletexTerminal             PrintableString (SIZE(1..ub-teletex-terminal-id)),
        parameters                  TeletexNonBasicParameters OPTIONAL}

VersionNumber::=        INTEGER (1..255)

x121Address ATTRIBUTE ::=       {
        WITH SYNTAX                 NumericString (SIZE (1..ub-x121-address))
        EQUALITY MATCHING RULE      numericStringMatch
        SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
        ID                          id-at-x121Address }


--search

searchGuide ATTRIBUTE  ::=      {
        WITH SYNTAX                 Guide
        ID                          id-at-searchGuide }

Guide       ::=     SET  {
        objectClass         [0]     OBJECT-CLASS.&id OPTIONAL,
        criteria            [1]     Criteria  }

Criteria    ::=     CHOICE {
        type                [0]     CriteriaItem,
        and                 [1]     SET OF Criteria,
        or                  [2]     SET OF Criteria,
        not                 [3]     Criteria}

CriteriaItem ::= CHOICE {
        equality            [0]     AttributeType,
        substrings          [1]     AttributeType,
        greaterOrEqual      [2]     AttributeType,
        lessOrEqual         [3]     AttributeType,
        approximateMatch    [4]     AttributeType}

enhancedSearchGuide ATTRIBUTE       ::=             {
        WITH SYNTAX                 EnhancedGuide
        ID                          id-at-enhancedSearchGuide }

EnhancedGuide               ::=      SEQUENCE {
        objectClass         [0]     OBJECT-CLASS.&id,
        criteria            [1]     Criteria,
        subset              [2]     INTEGER
              { baseObject (0), oneLevel (1), wholeSubtree (2) }  DEFAULT oneLevel }
```

3.5 Object Identifiers

3.5.1 Object identifiers can be found in X.520 and X.521, so are not reproduced below for informational purposes.

-- Object identifier assignments --

-- Object classes

```
id-oc-aTNEndSystem                  OBJECT IDENTIFIER       ::=         {id-oc21}
```

```
id-oc-aTNApplicationEntity          OBJECT IDENTIFIER    ::=    {id-oc22}

-- Attributes --
id-at-versionNumber                 OBJECT IDENTIFIER    ::=    {id-at61}
```

3.6 Matching Rules

3.6.1 Matching rules are essential to the operation of a Directory. Matching rules are used by DSAs to select a set of entries from the DIB based on assertions concerning attribute values held by these entries. Each matching rule needs to state the attribute syntax that the rule applies to, the syntax of the user presented value, how the comparison is to performed and the range of results possible (e.g. “True”, “False”, “Greater than”, etc). The Directory recognizes five basic types of matching: present, equality, substrings, ordering and approximate match. These types can be used on a variety of attributes.

3.6.2 The present syntax may be used for any attribute of any type. The present match tests for the presence of any value of a particular type. Specific equality, substrings and ordering matching rules may be associated with an attribute type when it is defined. That is, under what conditions are two values considered equal (is “12” equal to “1100”?), or how is order determined (is “3” greater than “10”?). These specific rules are used when evaluating assertions of the equality, ordering and substrings rules made using the syntax built-in to the Directory Abstract Service. If specific rules are not provided, then assertions made concerning these attributes are undefined. The approximateMatch syntax supports an approximate matching rule whose definition is a local matter to a DSA (i.e. does “Smith” equal “Smythe”?).

3.6.3 The ATN Directory will use the standard X.500 matching rules. The exact extent of the matching rules used is TBD, but probably will be customized for the ATN to perform searches on ATN-specific components of the Directory. For example, there may need to be a special definition for “decscription search” for the Application Entity object class, which will be specific to the keywords defined for ATN usage (e.g. perform a search on “CM Server” in order to determine all of the CM implementations which support server functionality). These specialized searching rules will be developed as the Directory concept matures.

3.6.4 Standard X.520 matching rules are given below for information purposes.

```
-- Matching rules --

caseIgnoreMatch MATCHING-RULE      ::=     {
        SYNTAX                      DirectoryString {ub-match}
        ID                          id-mr-caseIgnoreMatch }

caseIgnoreOrderingMatch MATCHING-RULE   ::=     {
        SYNTAX                      DirectoryString {ub-match}
        ID                          id-mr-caseIgnoreOrderingMatch }

caseIgnoreSubstringsMatch MATCHING-RULE         ::=     {
        SYNTAX                      SubstringAssertion
        ID                          id-mr-caseIgnoreSubstringsMatch }
```

```
SubstringAssertion ::= SEQUENCE OF CHOICE {
        initial         [0]     DirectoryString {ub-match},
        any                     [1]     DirectoryString {ub-match},
        final                   [2]     DirectoryString {ub-match} }
        -- at most one initial and one final component

caseExactMatch MATCHING-RULE       ::=     {
        SYNTAX                          DirectoryString {ub-match}
        ID                              id-mr-caseExactMatch }

caseExactOrderingMatch MATCHING-RULE    ::=     {
        SYNTAX                          DirectoryString {ub-match}
        ID                              id-mr-caseExactOrderingMatch }

caseExactSubstringsMatch MATCHING-RULE ::=      {
        SYNTAX                          SubstringAssertion -- only the PrintableString choice
        ID                              id-mr-caseExactSubstringsMatch }

numericStringMatch MATCHING-RULE  ::=     {
        SYNTAX                          NumericString
        ID                              id-mr-numericStringMatch }

numericStringOrderingMatch MATCHING-RULE ::=    {
        SYNTAX                          NumericString
        ID                              id-mr-numericStringOrderingMatch }

numericStringSubstringsMatch MATCHING-RULE     ::=      {
        SYNTAX                          SubstringAssertion
        ID                              id-mr-numericStringSubstringsMatch }

caseIgnoreListMatch MATCHING-RULE ::=      {
        SYNTAX                          SEQUENCE OF DirectoryString {ub-match}
        ID                              id-mr-caseIgnoreListMatch }

caseIgnoreListSubstringsMatch MATCHING-RULE    ::=      {
        SYNTAX                          SubstringAssertion
        ID                              id-mr-caseIgnoreListSubstringsMatch }

booleanMatch MATCHING-RULE  ::=     {
        SYNTAX                          BOOLEAN
        ID                              id-mr-booleanMatch }

integerMatch MATCHING-RULE    ::=     {
        SYNTAX                          INTEGER
        ID                              id-mr-integerMatch }

integerOrderingMatch MATCHING-RULE ::=    {
        SYNTAX                          INTEGER
        ID                              id-mr-integerOrderingMatch }

bitStringMatch MATCHING-RULE ::=     {
        SYNTAX                          BIT STRING
        ID                              id-mr-bitStringMatch }

octetStringMatch MATCHING-RULE       ::=     {
        SYNTAX                          OCTET STRING
        ID                              id-mr-octetStringMatch }

octetStringOrderingMatch MATCHING-RULE   ::=     {
        SYNTAX                          OCTET STRING
        ID                              id-mr-octetStringOrderingMatch }

octetStringSubstringsMatch MATCHING-RULE         ::=     {
        SYNTAX                          OctetSubstringAssertion
        ID                              id-mr-octetStringSubstringsMatch }

OctetSubstringAssertion ::= SEQUENCE OF CHOICE {
```

```
        initial         [0]     OCTET STRING,
        any             [1]     OCTET STRING,
        final           [2]     OCTET STRING }
        -- at most one initial and one final component

telephoneNumberMatch MATCHING-RULE      ::=     {
        SYNTAX                          PrintableString
        ID                              id-mr-telephoneNumberMatch }

telephoneNumberSubstringsMatch MATCHING-RULE        ::=     {
        SYNTAX                          SubstringAssertion
        ID                              id-mr-telephoneNumberSubstringsMatch }

presentationAddressMatch MATCHING-RULE  ::=     {
        SYNTAX                          PresentationAddress
        ID                              id-mr-presentationAddressMatch }

uniqueMemberMatch MATCHING-RULE ::=     {
        SYNTAX                          NameAndOptionalUID
        ID                              id-mr-uniqueMemberMatch }

protocolInformationMatch MATCHING-RULE  ::=     {
        SYNTAX                          OCTET STRING
        ID                              id-mr-protocolInformationMatch }

uTCTimeMatch MATCHING-RULE ::=  {
        SYNTAX                          UTCTime
        ID                              id-mr-uTCTimeMatch }

uTCTimeOrderingMatch MATCHING-RULE      ::=     {
        SYNTAX                          UTCTime
        ID                              id-mr-uTCTimeOrderingMatch }

generalizedTimeMatch MATCHING-RULE ::=  {
        SYNTAX                          GeneralizedTime
                                        -- as per clauses 34.3 b) or c) of CCITT Rec. X.208 | ISO/IEC 8824
        ID                              id-mr-generalizedTimeMatch }

generalizedTimeOrderingMatch MATCHING-RULE      ::=     {
        SYNTAX                          GeneralizedTime
                                        -- as per clauses 34.3 b) or c) of CCITT Rec. X.208 | ISO/IEC 8824
        ID                              id-mr-generalizedTimeOrderingMatch }

integerFirstComponentMatch MATCHING-RULE        ::=     {
        SYNTAX                          INTEGER
        ID                              id-mr-integerFirstComponentMatch }

objectIdentifierFirstComponentMatch MATCHING-RULE ::= {
        SYNTAX                          OBJECT IDENTIFIER
        ID                              id-mr-objectIdentifierFirstComponentMatch }

directoryStringFirstComponentMatch MATCHING-RULE ::= {
        SYNTAX                          DirectoryString { ub-directory-string-first-component-match }
        ID                              id-mr-directoryStringFirstComponentMatch }

wordMatch MATCHING-RULE         ::=     {
        SYNTAX                          DirectoryString {ub-match}
        ID                              id-mr-wordMatch }

keywordMatch MATCHING-RULE ::=  {
        SYNTAX                          DirectoryString {ub-match}
        ID                              id-mr-keywordMatch }
```

4. ATN Directory System Schema

4.1 The Directory system schema is a set of rules that control how operational information (information concerned with the actual functioning of the Directory, not ATC (or any other domain, for that matter) operational information) is stored in the Directory. From an administrative perspective, user information held in the DIB is supplemented by administrative and operational information represented by subentries and operational attributes. Subentries contain a description of the part of the DIT to which their collective attributes apply (note: most attributes have a collective part, but this document did not reproduce those parts in the ASN.1 since collective attributes will be a local issue). Operation attributes represent information used to control the operation of the Directory (e.g. access control information) or used by the Directory to represent some aspect of its operation (e.g. time stamp information). Since subentries and operation attributes are parts of the system schema, they are not visible in the user information model, although they may be made visible to adminstrators. Additionally, some operational attributes (such as timestamps) may be made visible to users. These concepts are depicted in Figure 9.

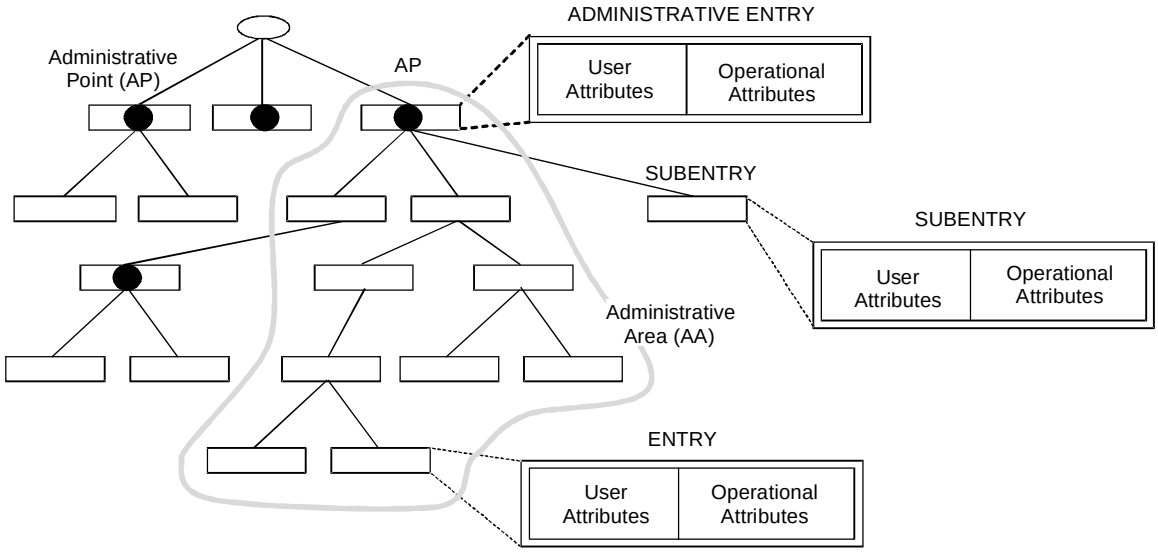


Figure 9. Administrative and Operational Entries

4.1.1 The Directory does not dictate which operational attributes may be held in which entries, and only dictates some of which operational attributes must be held in subentries. Therefore, the ATN Directory must specify the minimum set of operational attributes that must occur in entries and subentries. Additionally, there are subschema attributes, which apply to an AAA or an IAA. These are beyond the scope of Subvolume 7, since any additions to the ATN DIT will be a local implementation matter. However, it may be useful to register ATN DIT extensions with a body such as ICAO. The operational attributes that must be included with ATN entries are detailed in Section 4.2. It is TBD if there are additional, ATN-specific operational attributes that need to be defined (perhaps something like a log of the last 10 user that modified an entry). The use of adminstrative and operational entries in conjunction with the ATN Directory AAA and IAA (specifically access control adminstration, subschema adminstration and collective attribute adminstration) are discussed in sections 4.3- 4.5.

4.2 Directory Operational Attributes

4.2.1 The following Directory operational attributes are required to be present in the Directory system schema. Additional attributes, such as *Administrative Role*, may also be used if deemed necessary by the local user.

4.2.1.1 Create Timestamp

The *Create Timestamp* attribute is used to record when an entry was first created.

```
createTimestamp          ATTRIBUTE      ::=     {
        WITH SYNTAX                             GeneralizedTime
                        -- as per clause 34.3 b) and c) of CCITT Rec. X.20 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-createTimestamp }
```

4.2.1.2 Modify Timestamp

The *Modify Timestamp* attribute is used to record the last time an entry was modified.

```
modifyTimestamp          ATTRIBUTE      ::=     {
        WITH SYNTAX                             GeneralizedTime
                        -- as per clause 34.3 b) and c) of CCITT Rec. X.208 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifyTimestamp }
```

4.2.1.3 Creator's Name

The *Creator's Name* attribute is used to identify, via the distinguished name, the user that created an entry.

```
creatorsName            ATTRIBUTE     ::=     {
        WITH SYNTAX                           DistinguishedName
        EQUALITY MATCHING RULE                distinguishedNameMatch
        SINGLE VALUE                          TRUE
        NO USER MODIFICATION                  TRUE
        USAGE                                 directoryOperation
        ID                                    id-oa-creatorsName }
```

4.2.1.4 Modifier's Name

The *Modifier's Name* attribute is used to identify, via the distinguished name, the user that last modified an entry.

```
modifiersName           ATTRIBUTE     ::=     {
        WITH SYNTAX                           DistinguishedName
        EQUALITY MATCHING RULE                distinguishedNameMatch
        SINGLE VALUE                          TRUE
        NO USER MODIFICATION                  TRUE
        USAGE                                 irectoryOperation
        ID                                    id-oa-modifiersName }
```

4.2.1.5 Access Control Scheme

The *Access Control Scheme* attribute is held in an adminsitrative entry, and is used to identify the access control scheme in force.

```
accessControlScheme     ATTRIBUTE     ::=     {
        WITH SYNTAX                           OBJECT IDENTIFIER
        EQUALITY MATCHING RULE                objectIdentifierMatch
        SINGLE VALUE                          TRUE
        USAGE                                 directoryOperation
        ID                                    id-aca-accessControlScheme }
```

NOTE – the *Access Control Scheme* applies to ATN operational information only. The actual access control scheme to be put in place is TBD, but access control itself is described further in the following section.

4.2.1.6 Entry ACI

The *Entry ACI* attribute is used to control access to the entries in which they are held.

```
entryACI                ATTRIBUTE     ::=     {
        WITH SYNTAX                           ACIItem
        EQUALITY MATCHING RULE                directoryStringFirstComponentMatch
        USAGE                                 directoryOperation
        ID                                    id-aca-entryACI }
```

4.3 Access Control Aministration

4.3.1 Access control adminsitration for the ATN Directory will be reflected in the AAAs defined in Section 2.3. The goal of access control administration is to ensure that all properly authenticated ATN users have access to information necessary for the

operation and use of the ATN (e.g. application information and addresses, who that information belongs to, etc). This means that each of the AAPs and IAPs given in Section 2.3 will form an Access Control Specific Point (ACSP). The ACSP forms the root of the area over which an access policy will be enforced. This means that ACSPs for the ATN should be at the country, organization, and organizational unit level. This allows a country to be able to control access to non-ATN standard parts of the DIT. Of course, there is no requirement to have three different levels of access control; in fact a state could choose to implement just one at the country level, and have the entire part of the DIT under that country level adhere to the same access control scheme.

4.3.2 Access control is accomplished through matching lists of users' names with the information that they are allowed to access and the type of access they are allowed (e.g. modify, read only, etc). As mentioned previously, access control can be tailored to provide only access to the parts of a country's, organization's, or organizational unit's DIT that pertains to the operational use of the ATN. There are some issues that need to be addressed, namely who are these trusted users that should be allowed free reign over ATN information, and what kind of access should they be granted. An airline may also want to limit the spread of information about applications on board its aircraft, in order to conserve unwanted connection costs. These issues must be answered, but need to be coordinated with the security subgroup so that the two schemes do not contradict on another. Also, there may need to be additional information passed as part of the Directory that pertains to security and access control, and this needs to be firmly known before the details of access control can be documented.

4.4 Collective Attribute Administration

4.4.1 TBD, probably a local matter since this will merely say that a country, organization or organizational unit is free to expand upon the base ATN DIT, and then they become responsible for the definition of all aspects of that expansion. An exception may be any access control-related subentry stuff.

4.5 Subschema Administration

4.5.1 See 4.4.1.

5. DSA Description

5.1 DSA Distribution Model

5.1.1 As mentioned previously and depicted in Figure 2, the information in a Directory is managed by one or more DSAs. The DSAs manage access to the data that it controls (the DIB), as well as performs various operations on the data either on command of a DUA or another DSA.

5.1.2 Each entry within the DIB is administered by one, and only one, DSA's Administrator who is said to have administrative authority for that entry. Maintenance

and management of an entry takes place in a DSA administered by the administrative authority for the entry. This DSA is the master DSA for the entry.

5.1.3 Each master DSA within the Directory holds a fragment of the DIB. The DIB fragment held by a master DSA is described in terms of the DIT and comprises one or more naming contexts. A naming context is a subtree of the DIT, all entries of which have a common administrative authority and are held in the same master DSA. A naming context starts at a vertex of the DIT (other than the root) and extends downwards to leaf and/or non-leaf vertices. Such vertices constitute the border of the naming context. The superior of the starting vertex of a naming context is not held in that master DSA. Subordinates of the non-leaf vertices belonging to the border denote the start of further naming contexts. Note that a naming context in itself is not an administrative area having an administrative point or an explicit subtree specification; but it may coincide with an administrative area.

5.1.4

5.2 DSA Information Model

5.3 DSA Operational Bindings

5.4 Replication Strategy

5.4.1

5.5 Security/Access Control

5.5.1 TBD in conjunction with security subgroup.

5.6 Key Distribution

5.6.1 TBD in conjunction with security subgroup.

6. DUA Description

6.1 CM Interface

6.1.1

6.2 AIDC Interface

6.2.1

6.3 Recommendations for Generic User Interface

6.3.1

APPENDIX A – Sample Entries