

# X.500

## The X.500 Directory Service Environment

### **X.500 Directory Service**

- Distributed Service to access and maintain a tree structured logic data base called *Directory Information Base (DIB)*
- DIB is held by a number of *Directory Service Agents (DSAs)*
- Service users are represented by, and access the service via *Directory User Agents*.
- DSAs and DUAs are application entities.
- Service users are applications that use X.500 to improve the end user services. The User Interface is just one specific application program.

### **The Process Architecture of an X.500 Service**



#### **The Directory Name Space**

- Each *entry (node)* of the DIB is identified by a *Distinguished Name*: the path from the (nameless) Root to the entry
- Eg: [*Country=It; Org=System Wizards*]



## The Structure of an Entry

- Each entry belongs to an *Object Class*, consists in a collection of *attributes*, which in turn consist of a set of *values* of a specific type.



## The Name of an Entry

- The values of one or more attributes provide the *Relative Distinguished Name* of the entry
- The sequence of relative distinguished names provides the *Distinguished Name* of the entry.
- The actual name space of a DIB can be enriched by means of *Aliases*: entries of specific object class whose *AliasedObjectName* attribute value is a Distinguished Name
- The DSA performs *Name Resolution*

## The X.500 Service

- The service provision first requires the DUA to bind to a specific DSA. At bind time, DUA and DSA can verify each other identity (Authentication)
- Service elements are available to:

- Query the Directory
  - Read
  - Compare
  - List
  - Search
- Modify the Directory
  - Add Entry
  - Remove Entry
  - Modify Entry
  - Modify RDN

### **The X.500 Service (Cont)**

- Manage the Association
  - Abandon
- Service Controls are available to control:
  - Time Limit
  - Result Size Limit
  - Search Scope
  - Request Priority
  - Use of Alias
  - Distributed Operation Options

### **Directory Distribution**

- The DIB is partitioned into subtrees called *naming contexts*
- Each DSA takes the responsibility of *mastering*

one or more naming contexts. In addition, it may hold *shadow copies* of one or more naming contexts.

- The Directory Service takes the responsibility for locating the information requested by the DUA
- To find the required entry, it uses *Knowledge Information (KI)*: information about the distribution of the DIB

## Naming Contexts



### DUAs and DSAs Co-operation

- Chaining



### DUAs and DSAs Co-operation

- Referral
- Directory Access Protocol (DAP)
- Directory Service Protocol (DSP)



## The Protocol Stack

- Asynchronous ROSE Association Class 1 for

## DAP

- Asynchronous ROSE Association Class 3 for DSP
- ASN.1 encoding of operations, arguments, results and errors

## **X.500 DUA API**

- Provides access to all the X.500 services
- Services are invoked through a C programming interface
- Very simple interaction with the application: easy to use
- Synchronous operations
- High degree of transparency : the communication details are hidden from the application as much as possible
- Automatic handling of referrals
- Support for all the mandatory NIST attributes

## **X.500 DUA API (Cont)**

- Association Management
  - `dua_bind (dsa_address, bind_arg, bind_res)`
  - `dua_unbind (conn_id)`
  - `dua_abandon (conn_id)`
- Directory Modification
  - `dua_add (add_arg, common_arg)`
  - `dua_remove (remove_arg, common_arg)`

- dua\_modify (modify\_arg, common\_arg)
- dua\_modrdn (modrdn\_arg, common\_arg)
- Directory Query
  - dua\_read (read\_arg, common\_arg, read\_res, common\_res)
  - dua\_compare (compare\_arg, common\_arg, compare\_res, common\_res)
  - dua\_list (list\_arg, common\_arg, list\_res, common\_res)
  - dua\_search (search\_arg, common\_arg, search\_res, common\_res)

## **Bulk Loading and Unloading**

- Used for
  - Installation
  - Backup
  - Data base schema modification
- ASCII format
- Implemented on top of the DAP API
  - Easy to run on a LAN
  - Useful on a multivendor environment
  - Gives flexibility for the configuration of the DSA machine

## **Man Machine Interfaces**

- A Directory is sold:
  - On the benefits applications can obtain from it
  - On the interoperability it provides among heterogeneous systems
- MMIs provide an added value to an already sold Directory
- Administrators are very important users
- Administration interfaces are key factor to the success of a Directory: especially for X.500

## **Testing Tools**

- Invaluable tool for development
- Developed on top of an independent OSI stack
- DSA testing only
- X.500 service testing is quite complex
- Conformance is even more complex
- DIB schemata introduce subtle interoperability problems

## **Process Structure**

- Very different configurations can be of interest
- Minimal configuration
  - A single DSA process for DAP and DSP
  - Disk is bottleneck

- Service can be saturated by long lasting operations
- Medium performance
  - Several DSA processes
  - Service can be saturated by long lasting operations
  - If chaining is needed only one can perform it (weakness of the standard)
- High performance on all operations
  - Several DSAs but specialised on operation types
  - One process for managing chained operations
  - One dispatcher process that manages DAP and schedules operations on DSA processes

## **The Extensions**

- Strictly speaking only *Shadowing*
- Local matters:
  - *Access Control*
  - *Schema*
  - *Knowledge Representation*

## **Shadowing**

- A *shadow naming context* is
  - a copy of naming context *mastered* by another DSA

- a *read-only* copy
- hopefully *more available* than the master
- *not* guaranteed to be 100% of the time identical to the master
- The propagation of modification is taken care of by the Directory
- Knowledge information allows
  - to recognise shadows
  - to decide where to propagate updates
  - to recognise updates from the master
- Shadows provide good response time and high data availability
- Shadows are unmanageable unless specific tools are provided in the product

## **Access Control**

- Access Control List Model
- ACLs can be present at (and their scope is that of) each entry
- Lists may give access to
  - everybody
  - nobody
  - individual users
  - group of users depending on the Name Tree
  - self
  - self+subordinates
- Ereditarietà delle ACL

- There is an ACL
  - for each group of attributes
  - for each group of operations
- The groups of attributes and the group of operations can easily be configured
- (Unprotected simple) authentication
  - Easy to manage with the access control schema itself

## **Schema**

- Defines
  - naming rules
  - object classes and their attributes
  - attribute types semantics
- Could be described in the DIB
  - by a single specific entry
  - with specific attributes
  - for each of the components of the schema
- Applications can be parametric to schema
- Schema could be managed using DAP APIs

## **Knowledge Representation**

- Explicit DIB representation of
  - superior references
  - subordinate references
  - cross references
  - shadow references

- Management utilities are built on top of DAP
- Didn't prove a great advantage
- The Directory can describe itself to a great extent

## **Configuration Management**

- The DSA code reconfiguration (e.g. attribute types support) should not require source code recompilation
- Distribution of Directory is a complex configuration task
- Global consistency of a distributed system is quite difficult to achieve
- Explicit representation of Knowledge Information at least allows an easy check of the configuration

## **Directory Management**

- Directory makes networks and distributed applications management easy
- Management of the Directory is a complex task
- Management is impossible without proper support tools designed for Directory Managers

## **Performance and Functionality:**

### **What for?**

- Directories for Users
- Directories for Applications

## **Directories for Users**

- Very large databases
  - Rich data model
  - Problem of presenting the model to the user
  - => X.500 could be glue for a world-wide network
- of Information Retrieval Systems

## **Directories for Applications**

- Real-time response is not always needed
- Data model must be simple and uniform
- Data model needs customisation for specific application areas
- X.500 standard with suggested object classes and attributes is just a good start
- The data base technology to adopt needs to be co-ordinated with the commercial strategy of the vendor (and of the customer!)
- User browsing is an added value: users want to see the data their applications are making use of