Developing LDAP Applications

October 16, 2002

Chris Apple
Principal Architect
DSI Consulting, Inc.

capple@dsi-consulting.net

DSI Corporate Overview

- Provides Directory Services and Directory-Enabled IT Solutions to medium- and large-sized organizations
- Founded in 2001 by 3 Novell Alumni:
  - Sandra Harrell – CEO/President
  - Jerry Combs – Chief Architect
  - Mike Saunders – Business Manager
- Broad coverage of the US with locations in: Virginia/DC, Boston, Atlanta, Florida, Oklahoma, California, and Philadelphia
- Average DSI Associate has over 15 years of Directory-Related Experience
- Currently have 10 consultants on staff and growing
- Working relationships with several other Directory-Related consulting shops to augment staff as needed

dsi
CONSULTING, INC

Agenda

- What is a Directory? “Sounds simple to me…”
- What’s Important in Directories?
- Directories 101
- Application Development Guidelines

What is a Directory?

- A special-purpose, typed data set
- A means of translating one type of information into a different, but associated type of information
- Typically, a directory is read more often than written to
- It is not:
  - A general-purpose database
  - A solution to every data storage problem

Sounds Simple. Right?

- Well, not really…
- Why?
  - Early stage standards groups and industry consortia tend to create inflated expectations of what a technology will be capable of delivering
  - Inflated expectations tend to result in complicated specifications
  - Complicated specifications? We all know what that means…
  - X.500 and LDAP were not immune
- The Truth?
  - Directories, directory services, and directory-enabled applications can be combined with other technologies to form solutions that solve many of today's IT problems
  - Integrating directory technology into IT infrastructure is anything but trivial
  - You should consider what's important to you before attempting it

dsi CONSULTING, INC

Perspectives of Importance

- Users
- Developers and Administrators
- Decision Makers
- Technology “Good Things”

What's Important to Users?

- Finding each other
- Communicating
- Sharing information
- Collaborating on projects
- Running business applications
- Using the personal computing device/OS of their choice
- Reliability
- Ease of use

What's Important to Administrators and Developers?

- Plug-and-play installation utilities
- Flexible diagnostic and monitoring tools
- Data conversion tools and interfaces
- A powerful but simple SDK
- Training on how to use administrative and development tools
- Server OS portability
- Documentation, documentation, documentation
- Access to 24x7 developer support

dsi CONSULTING, INC

What's Important to Decision Makers?

- Low start-up and maintenance costs
- Increasing productivity
- Decreasing IT operations costs
- Open standards support
- ROI
- Future-proofing IT investments

So

dsi CONSULTING, INC

What's Important in Technology?

- A simple, well-defined directory service access protocol
- A simple API
- Support for security-related key storage
- OS portability
- Replication/synchronization methods
- A high-performance search engine
- An extensible directory schema
- Seamless integration with existing corporate Intranet

dsi
CONSULTING, INC

LDAP 101

LDAP is NOT X.500

LDAP Background

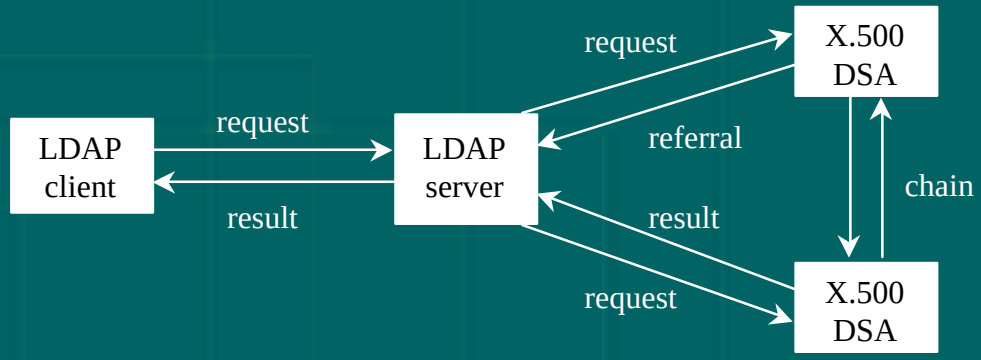
- LDAP = Lightweight Directory Access Protocol
- Current version is LDAPv3
- PDUs encoded for transport across TCP using a lightweight subset of the BER used for X.500 DAP (this means the byte stream is binary and not easily readable by humans)
- Queries are constructed using a string representation of a boolean combination of attribute-value assertions
- Matching algorithms are similar to those for X.500
- Standards are based on a hierarchical information model similar to that for X.500
- Standard object classes and attributes are also taken largely from X.500
- A simple LDAP API with C and Java bindings is available
- Implementations are easier to build, configure, administer, and maintain than X.500…but LDAP is not X.500
- Can operate over SSL or in the clear

Technical Differences Between X.500 and LDAP?

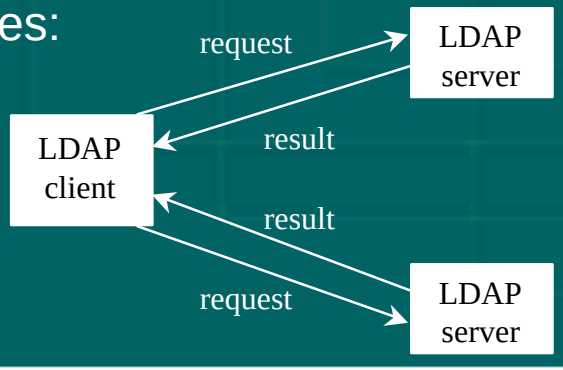
- LDAP is a simplified, lightweight subset of the X.500 DAP
- Specifically, LDAP features simplifications over X.500 DAP:
  - **Transport:** LDAP runs directly over TCP, bypassing some of the upper-layer overhead of an OSI stack
  - **Functionality:** LDAP simplifies the functionality provided by X.500 DAP, leaving out little-used features and redundant operations
  - **Data Representation:** LDAP represents most data elements using simple string formats, which require less processing than X.500's representation method (ASN.1)
  - **Encoding:** LDAP encodes data for transport over networks using a simplified version of the same encoding rules used by X.500 (BER)
- LDAP can be used as a gateway technology to interconnect LDAP clients with existing X.500 DSAs

What is Stand-alone LDAP?

- Historically, LDAP originated out of a desire to connect desktop applications with X.500 DSAs over TCP/IP.



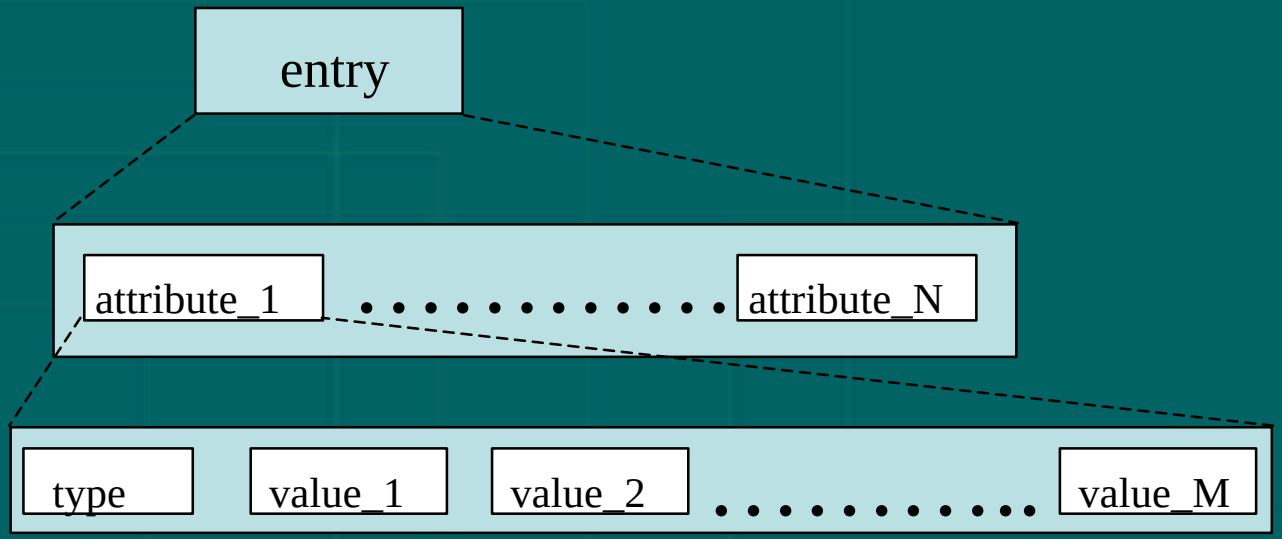
- Stand-alone LDAP originated out of a desire to divorce LDAP models from X.500 and to reduce the complexity of developing directory services:



The LDAP Models

- LDAP defines four models that describe its operation, the kinds of information that can be stored in LDAP directories, and what can be done with this information:
  - **Information Model:** defines what kind of information can be stored in an LDAP directory
  - **Naming Model:** defines how information in an LDAP directory can be organized and referenced
  - **Functional Model:** defines what can be done with the information in an LDAP directory as well as how it can be access and updated
  - **Security Model:** defines how the information in an LDAP directory can be protected from unauthorized access or modification

LDAP Information Model



- The LDAP information model is centered around the *entry*
- *entries* are often created to represent some real world object (exs: a person, an organization, an application, or a printer)
- *entries* are composed of *attributes* that contain information to be stored about the object in the directory
- each *attribute* has a *type* and one or more *values*

LDAP Information Model (cont.)

- the type of an attribute has an associated *syntax* that defines what kind of information can be stored in the attribute's values
  - ex: the cn (commonName) attribute has a syntax called caseIgnoreString that implies that case is ignored during comparisons and that values must be character strings
- attribute types can also have various constraints associated with them:
  - limiting the number of values stored in an attribute to one or many
  - limiting the size of value(s) stored in an attribute
- optional vs. mandatory status for attributes permitted in an entry are specified by *content rules* on a per-server basis or by a special attribute in every entry called objectClass
- values of the objectClass attribute for an entry identify its type and indicate which attributes are required and which are allowed
  - ex: the object class person requires the sn (surname) and cn (commonName) attributes and allows description, seeAlso, and other attributes

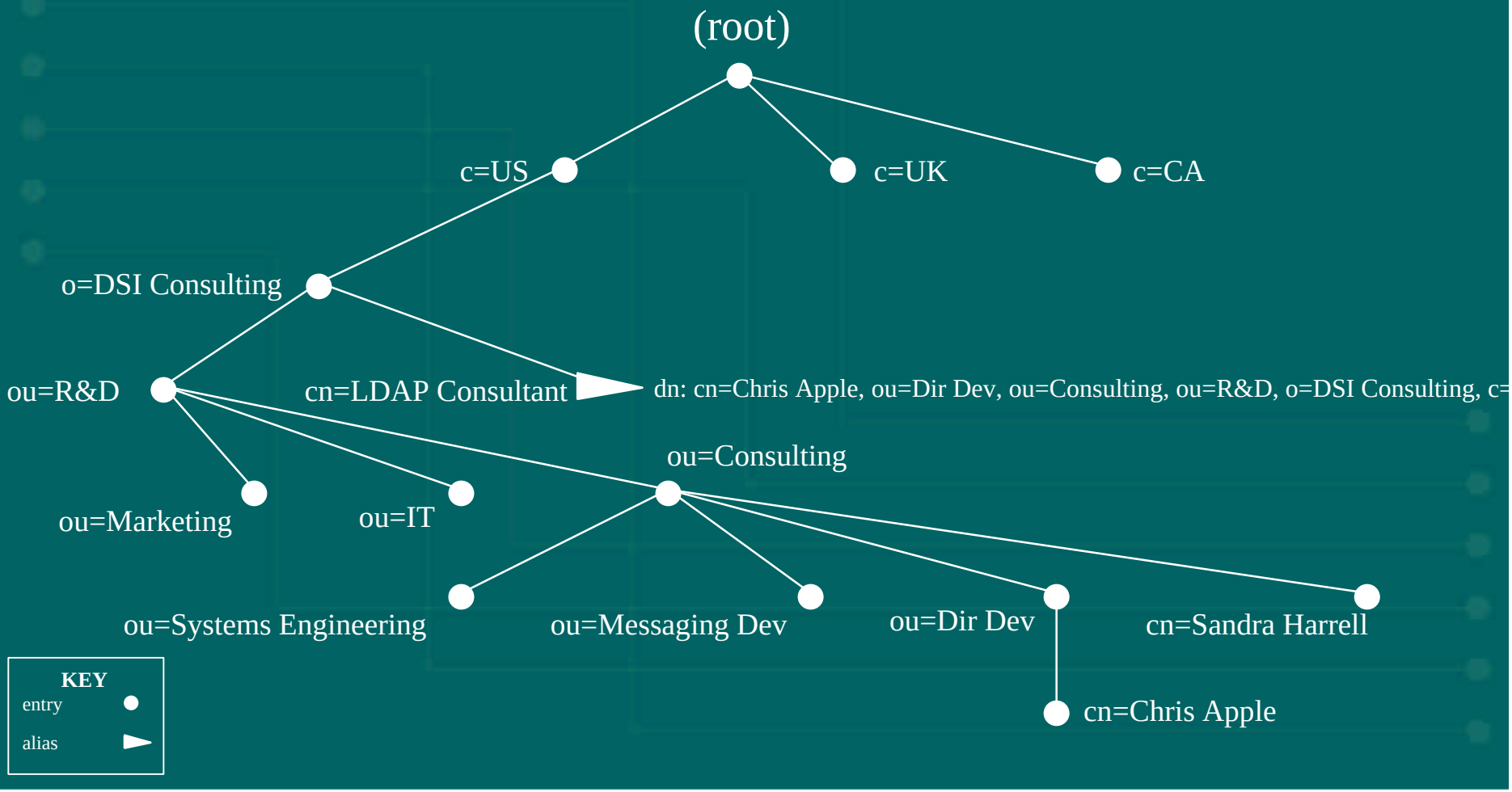
LDAP Information Model (cont.)

- this collection of terms, definitions, and relationships associated with components of an entry make up the LDAP equivalent of what traditional databases call a *schema*
- the schema in force for a particular entry may be changed by adding new object classes to the entry
- each entry contains an object class, called its *structural* object class, that determines what kind of entry it is; this object class cannot be changed
  - exs: person, organizationalPerson, and residentialPerson are structural object classes
- other object classes, called *auxiliary* object classes, may be added to or deleted from the entry, subject to any access control rules that may be in effect for that entry
  - ex: MHS-user is an auxiliary object class

LDAP Naming Model: Overview

- entries are usually arranged in a tree structure called a Directory Information Tree (DIT); often a DIT will be based on geographical and/or organizational distributions of entries
- entries are named according to their position in the DIT by a distinguished name (DN).
- each component of a DN is called a relative distinguished name (RDN) and is composed of one or more attributes from the entry to which the DN refers
- LDAP does not required that entries be organized in a hierarchical manner, but it does require that DNs be unique within the context of the DIT
- LDAP DNs are “little endian”
- *alias* entries, which point to other entries, are supported; thus the hierarchy can be circumvented; a loop detect mechanism exists in most LDAP implementations

LDAP Naming Model: Aliasing



LDAP Functional Model: Overview

- Functionally, LDAP defines nine operations in four areas:
  - Interrogation: Search, Compare
  - Update: Add, Delete, Modify, ModifyRDN
  - Authentication: Bind, Unbind
  - Cancellation: Abandon
- Neither client nor server is required to engage in a synchronous dialog
- Referrals are usually handled by the server
- Connection-Oriented transport streams are assumed (TCP/IP)
- LDAP Message PDUs are mapped directly onto a TCP byte stream
- Standard port is 389; Secure port is 686

LDAP Functional Model: Interrogation Operations

- Search Operation:
  - the search operation is used to select entries from a defined area of the DIT based on selection criteria referred to as a search filter
  - a requested set of attributes (with or without values) can be returned
  - if no attributes are explicitly requested as a part of a search operation, the server returns all attributes (with values) that exist for each matching entry
  - the scope of the search can be limited to one entry, an entry's children, or extended to span an entire subtree
  - alias entries can be followed automatically during a search
  - the client can specify size and time limits on the search
- Compare Operation:
  - the compare operation is used to test an attribute-value assertion without returning entries to the client
  - used in password-based authentication

dsi CONSULTING, INC

LDAP Functional Model: Update Operations

- Add Operation:
  - used to insert a new entry into the DIT
  - usually subject to access control rules
- Delete Operation:
  - used to remove an entry from the DIT
  - usually subject to access control rules
- Modify Operation:
  - used to change the attributes and values contained in an existing entry
  - usually subject to access control rules
- ModifyRDN Operation:
  - used to change the name of an entry
  - can only be used to change the RDN of the entry and thus is not useful for moving an entry under a new parent in the DIT
  - usually subject to access control rules

LDAP Functional Model: Authentication and Cancellation Operations

- Bind Operation:
  - allows a client to prove its identity to the directory service
  - anonymous, simple clear-text password, and kerberos-based authentication mechanisms are supported
  - the server does not prove its identity to the client
  - client supplies a DN and authentication credentials
- Unbind Operation:
  - allows a client to terminate a directory session
  - client provides no additional information, just sends request
- Abandon Operation:
  - allows a client to terminate a previously initiated operation
  - most useful in canceling a lengthy search operation before it times out

dsi
CONSULTING, INC

LDAP Security Model: Overview

- built around knowing the identity of the clients requesting access to the directory
- this information is provided by the Bind Operation
- once known, the identity of a client can be compared with access control information to determine if the client has rights to do what it is requesting
- LDAP does not specify the format or capabilities of access control information
- access control information is specified according to proprietary schemes intended to give administrators the ability to construct an access control (rules) list (ACL) for the directory services
- typically, clients can be granted or denied access rights based on DN, IP address, and/or domain name

Forming LDAP Queries: Components

- base DN (optional)
  - default is directory server root
  - must be a DN valid within the context of the DIT
  - a reference to the base object to which the search scope applies
- search scope (optional)
  - base: only search for the base object
  - one-level: search only the children of the base object
  - subtree: search entire subtree below and including base object (default)
- search filter
  - potentially an arbitrarily complex boolean combination of attribute-value assertions that act as a collective selection argument for entries retrieved from the directory
- list of attributes to return and attributes-only indicator (optional)
  - only attributes and values explicitly requested will be returned if specified
  - default is to return all attributes and values
- timeout (seconds) and size limits (max number listings) (optional)
  - smaller values override larger values when comparing server-side defaults and client-supplied values

LDAP Search Filters: Simple

Filter Type	Format	Example	Matches
Equality	(<attr>=<value>)	(sn=Apple)	surnames exactly (lexicographically) equal to Apple
Approximate	(<attr>~=<value>)	(sn~=Appel)	surnames approximately equal to Appel (typically soundex or metaphone)
Substring	(<attr>=[<leading>]*[<any>]*[<trailing>])	(sn=*ppl*) (sn=app*) (sn=*pple) (sn=ap*l*)	surnames containing the string “ppl” surnames starting with the string “app” surnames ending with the string “pple” surnames containing the substring tokens “ap” and “l” (order is significant)
Greater than or equal	(<attr>>=<value>)	(sn>=apple)	surname lexicographically greater than or equal to “apple”
Less than or equal	(<attr><=<value>)	(sn<=apple)	surname lexicographically less than or equal to “apple”
Presence	(<attr>=*)	(sn=*)	all objects with a surname

dsi CONSULTING, INC

LDAP Search Filters: Complex

Filter Type	Format	Example	Matches
AND	(&(<filter1>)(<filter2)…)	(&(sn=Apple)(objectclasss=person)) (&(sn=Apple)(mail=capple*))	people with a surname of Apple surname of Apple AND e-mail address starting with “capple”
OR	(|(<filter1>)(<filter2>)…)	(|(sn~=appel)(cn=*appel))	surname approximately equal to “appel” OR common name ending with “appel”
NOT	(!(<filter>))	(!(mail=*))	entries without an e-mail attribute

Complex LDAP search filters can be combined and nested to form arbitrarily complex search filters; this provides a very powerful entry selection facility for LDAP services, which is both good and bad:

- knowledgeable users can benefit from it
- inexperienced users can cause heavy load on LDAP servers
- malicious users could initiate denial-of-service attack

LDAP URLs

- Everything else has a URL, why not LDAP
  - useful for providing server-to-client and server-to-server referrals
  - not very useful from an end-user's perspective
- An LDAP URL is also used to implement Referrals to other LDAP servers

LDAP Application Development Guidelines

- Coping with the complexity of LDAP
- Naming
- Unique Identifiers
- DIT Structure
- Schema and Extensions
- Security
- Authentication
- Access Control
- Integrating with Other Systems
- Performance
- Administration and Maintenance
- Political

Coping With LDAP's Complexity

- See LDAP 101 – it's not a simple technology to master
- Determine if you have the skills in-house
- If not, you can either grow the skills, recruit people with the skills, or engage qualified consultants
- Recruiting people with these skills is hard
- Growing these skills can be expensive
  - Seek self-directed and instructor-led training
  - Prototype applications and experiment in a lab
  - Count on the first few projects being partial successes
- No worthwhile consultant comes cheap
- Any course of action can result in success eventually
- You should expect a few fumbles if you are just starting

dsi
CONSULTING, INC

Naming, The DIT, and Unique Identifiers

- Naming has always been a hard problem
- Namespace collisions are bad
- LDAP uses a hierarchical naming scheme related to DIT structure to avoid this
- If you have different organizations designing different LDAP services without collaboration, you may have collisions regardless
- Thus its wise to centralize directory design work to a single group for naming, DIT structure, and unique identifiers

Schema and Extensions

- Try to use existing schema elements
- Admittedly, the standard schema for LDAP isn't the most robust
- LDAP's schema extensibility methods:
  - creation of new attribute types
  - creation of new container object classes
  - creation of new auxiliary object classes
- There is no central resource for discovering schema elements
- Consider using schema publishing mechanisms
- Avoid hard-coding schema knowledge into applications developed
- A more common alternative to schema publishing is using a schema configuration file local to directory-enabled applications
- You should also consider centralizing schema design to avoid:
  - inconsistency/incompatibility for similar schema extensions
  - redundancy of schema extensions

dsi
CONSULTING, INC

Security

- Ask yourself all the typical questions:
  - Privacy?
  - Confidentiality?
  - Integrity?
- If the answer is yes to one or more, consider using LDAP over SSL to provide some protection.
- If its not really important in your context, plain old LDAP will generally perform at a higher level.

Authentication

- Analyze how strong your authentication should be for:
  - each particular application
  - each community of users
- Anonymous Binds have less operational overhead than Authenticated Binds.
- Don’t expose your system to unnecessary risk, but also keep in mind the performance implications (and associated costs) of Authenticated Binds.

Access Control

- Analyze access rights requirements for:
  - each application using the directory
  - each user community
- Design as simple a set of access rules as possible for your needs
- Keep in mind that performance degrades (or cost increases) as your ACL becomes more complicated and lengthy

dsi CONSULTING, INC

Integrating with Other Systems

- Collect information about:
  - the systems information repositories
  - the business processes that drive information into those repositories
  - privacy, legal, and regulatory requirements
- Determine information flows to/from the directory and these other systems
- Integration Options to consider:
  - File-based Imports/Exports
  - Meta Directories
  - Virtual Directories
  - Other Information Synchronization Products
  - Custom-Built Solutions

dsi
CONSULTING, INC

Performance

- Key Areas of Concern:
  - Reliability/Availability
  - Throughput and Latency
  - Time-to-Repair
  - Scalability
- Design Considerations:
  - DIT Structure
  - DIT Size
  - Entry Size
  - Number of Attributes Per Entry
  - Use of Large Binary Attributes
  - Server Indexing
  - Server-Side Sorting LDAP Extension
  - Paged Results LDAP Extension
  - Partition Size
  - Partition Topology
  - Redundancy of Equipment
  - Replication and Replication Topology
  - Core DBMS Scalability
  - Hardware Configuration
  - Operational Load Profile
  - Various Tunable Server Parameters
  - Core DBMS Tuning

dsi
CONSULTING, INC

Administration and Maintenance

- Periodic wise things:
  - export database to LDIF
  - replication change log cycling
  - DBMS admin as required
- Other wise things:
  - automated backups to tape of the above
  - automated backups of configuration files
  - active system performance monitoring
  - consider using SNMP MIB to manage server

Political

- Consider doing an ROI analysis before attempting to secure funding for the development project
- Develop a project plan that reaches out to the deployment stage
- Engage all stakeholders and obtain buy in based on value delivered to them and the organization as a whole
- By all means – be sure it is a solution to real problems your organization has.
- DO NOT SIMPLY SAY A DIRECTORY IS A GOOD THING

Questions?

Chris Apple
Principal Architect
DSI Consulting, Inc.
capple@dsi-consulting.net