LDAP SERVER - DIRECTORY SERVER

Της Σπουδάστριας
ΑΛΕΞΑΝΔΡΑ ΤΡΙΣΣΟΠΟΥΛΟΥ

03/11/2005

ΕΙΣΗΓΗΤΗΣ
ΓΕΩΡΓΙΟΣ Ε. ΡΙΖΟΣ

Πτυχιακή Εργασία μέρος των απαιτήσεων
Του τμήματος Τηλεπληροφορικής και Διοίκησης

ΠΡΟΛΟΓΟΣ

Οι υπηρεσίες καταλόγου αποτελούν ένα σημαντικό μέρος των υπολογιστικών περιβαλλόντων των επιχειρήσεων. Οι υπηρεσίες αυτές μας δίνουν την δυνατότητα να συλλέγουμε πληροφορίες που αφορούν χρήστες, εφαρμογές, αρχεία, πηγές εκτυπωτών, ελέγχους πρόσβασης και άλλους πόρους. Οι πληροφορίες αυτές αποθηκεύονται σε έναν κατάλογο ο οποίος προσπελάζεται από τους χρήστες και τις εφαρμογές ενός δικτύου. Όλες οι εφαρμογές καταλόγου μπορούν να χρησιμοποιηθούν από τις επιχειρήσεις. Αυτό ελαχιστοποιεί τις νησίδες πληροφορίας που δημιουργούνται όταν οι εφαρμογές χρησιμοποιούν τις δικές τους εξειδικευμένες αποθήκες για να διαχειριστούν τις πληροφορίες των πόρων. Επιπλέον οι υπηρεσίες αυτές βελτιώνουν την διοίκηση επειδή η διαχείριση των πληροφοριών είναι συγκεντρωτική. Οποιεσδήποτε προσθήκες ή αλλαγές που γίνονται στις πληροφορίες καταλόγου είναι άμεσα διαθέσιμες σε όλους τους χρήστες και σε όλες τις εφαρμογές που χρησιμοποιούν τον κατάλογο. Παραδείγματος χάριν, αντί της αλλαγής του καταλόγου ελέγχου πρόσβασης ενός πόρου σε κάθε σύστημα που το προσπελάζει, μπορεί να γίνει αλλαγή μόνο μιας πληροφορίας και αυτή να χρησιμοποιείται από κάθε εφαρμογή που ελέγχει την πρόσβαση σε αυτόν τον πόρο.

Το LDAP (Lightweight Directory Access Protocol) είναι μια ταχέως αναπτυσσόμενη τεχνολογία για την πρόσβαση σε κοινές πληροφορίες καταλόγου και έχει υιοθετηθεί και εφαρμοσθεί από τα περισσότερα προϊόντα δικτύου. Ως ένα ανοιχτό πρότυπο παρέχει μια αρχιτεκτονική με δυνατότητα επέκτασης για την διαχείριση και συγκέντρωση αποθηκευμένων πληροφοριών οι οποίες πρέπει να είναι διαθέσιμες για τα σημερινά κατανεμημένα συστήματα και τις υπηρεσίες. Ύστερα από μια γρήγορη έναρξη, μπορεί να θεωρηθεί ότι το LDAP έχει γίνει η de facto μέθοδος πρόσβασης σε πληροφορίες καταλόγου, αρκετά παρόμοια με το DNS (Domain Name System) το οποίο χρησιμοποιείται για την αναζήτηση διευθύνσεων IP σχεδόν σε οποιοδήποτε σύστημα ενός Intranet ή στο Internet. Το LDAP πρόσφατα υποστηρίζεται από τα περισσότερα λειτουργικά δικτυακά συστήματα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: ΕΝΝΟΙΑ LDAP

Άτομα και επιχειρήσεις βασίζονται όλο και περισσότερο στα δικτυωμένα συστήματα υπολογιστών για να υποστηρίξουν κατανεμημένες εφαρμογές. Αυτές οι εφαρμογές ίσως να αλληλεπιδρούν με υπολογιστές που βρίσκονται στο ίδιο τοπικό δίκτυο (LAN), το οποίο είναι μέρος ενός εταιρικού δικτύου (Intranet) ή οπουδήποτε στο παγκόσμιο διαδίκτυο (Internet). Για την βελτίωση της λειτουργίας, της ευκολίας της χρήσης και για να είναι επιτρεπτή η οικονομικώς αποδοτική διαχείριση των κατανεμημένων εφαρμογών πληροφοριών που αφορούν υπηρεσίες, πόρους, χρήστες και άλλα αντικείμενα που είναι προσβάσιμα μέσω των εφαρμογών, χρειάζεται να είναι οργανωμένες με ένα σαφή και συνεπή τρόπο. Ένα μεγάλο μέρος αυτών των πληροφοριών μπορεί να διαμοιραστεί μεταξύ πολλών εφαρμογών, αλλά συγχρόνως πρέπει να προστατεύεται για να αποτρέπεται η παράνομη τροποποίηση ή η κοινοποίηση των ιδιωτικών πληροφοριών.

Οι πληροφορίες που χαρακτηρίζουν τους διάφορους χρήστες, τις εφαρμογές, τα αρχεία, τους εκτυπωτές και άλλους πόρους που προσπελαύνονται από ένα δίκτυο, συχνά συγκεντρώνονται σε μια ειδική βάση δεδομένων που μερικές φορές αποκαλείται «κατάλογος». Καθώς έχει αυξηθεί ο αριθμός των διαφορετικών δικτύων και εφαρμογών, έχει επίσης αυξηθεί ο αριθμός των εξειδικευμένων καταλόγων πληροφοριών με αποτέλεσμα να δημιουργούνται νησίδες πληροφοριών οι οποίες δεν μπορούν να είναι κοινόχρηστες και είναι πολύ δύσκολο να διατηρούνται. Εάν όλες αυτές οι πληροφορίες μπορούσαν να διατηρούνται και να είναι προσπελάσιμες κατά τρόπο συνεχή και ελεγχόμενο θα παρεχόταν ένα σημείο συγκέντρωσης για την ενσωμάτωση ενός κατανεμημένου περιβάλλοντος σε ένα συνεπές και χωρίς όρια σύστημα.

Το Ελαφρύ Πρωτόκολλο Πρόσβασης Καταλόγου (LDAP) είναι ένα ανοιχτό βιομηχανικό πρότυπο το οποίο έχει εξελιχθεί για να ικανοποιήσει αυτές τις ανάγκες. Το LDAP καθορίζει μια πρότυπη μέθοδο για την πρόσβαση και ενημέρωση των πληροφοριών σε ένα κατάλογο. Το LDAP κερδίζει ευρεία αποδοχή ως μέθοδος πρόσβασης καταλόγου του Internet και επομένως γίνεται στρατηγικό μέσο στα εταιρικά Intranets. Υποστηρίζεται από έναν αυξανόμενο αριθμό προμηθευτών λογισμικού και είναι ενσωματωμένο σε ένα μεγάλο αριθμό εφαρμογών που συνεχώς αυξάνεται.

1.1 Έννοια Καταλόγου και Υπηρεσίας Καταλόγου

Ένας κατάλογος (directory) είναι μια λίστα πληροφοριών για αντικείμενα τα οποία είναι τακτοποιημένα με κάποια σειρά και δίνει λεπτομέρειες για κάθε αντικείμενο χωριστά. Κοινά παραδείγματα αποτελούν ο τηλεφωνικός κατάλογος μιας πόλης και ο κατάλογος καρτών μιας βιβλιοθήκης. Για έναν τηλεφωνικό κατάλογο, τα αντικείμενα τα οποία ταξινομούνται είναι άτομα. Τα ονόματά τους τακτοποιούνται αλφαβητικά και οι λεπτομέρειες που δίνονται για κάθε άτομο είναι η διεύθυνσή του και ο τηλεφωνικός αριθμός του. Τα βιβλία σε έναν κατάλογο καρτών μιας βιβλιοθήκης διατάσσονται σύμφωνα με τον συγγραφέα και τον τίτλο του βιβλίου και

δίνονται πληροφορίες όπως ο αριθμός ISBN του βιβλίου ή οποιαδήποτε άλλη πληροφορία που αφορά τις εκδόσεις.

Στην ορολογία των υπολογιστών, ένας κατάλογος είναι μια εξειδικευμένη βάση δεδομένων, αποκαλούμενη επίσης «αποθήκη δεδομένων», στην οποία αποθηκεύονται τυποποιημένες και διατεταγμένες πληροφορίες για τα αντικείμενα. Ένας ειδικός κατάλογος μπορεί να απαριθμεί πληροφορίες για τους εκτυπωτές (αντικείμενα) οι οποίες περιέχουν τυποποιημένες πληροφορίες όπως η θέση των εκτυπωτών (ένα μορφοποιημένο string χαρακτήρων), η ταχύτητα των σελίδων ανά λεπτό (αριθμητικό), οι ροές εκτύπωσης που υποστηρίζονται (π.χ PostScript ή ASCII) και ούτω καθεξής. Οι κατάλογοι επιτρέπουν στους χρήστες ή στις εφαρμογές να βρουν πηγές που έχουν τα χαρακτηριστικά που χρειάζονται για ένα συγκεκριμένο σκοπό. Για παράδειγμα, ένας κατάλογος χρηστών μπορεί να χρησιμοποιηθεί για την ανεύρεση της ηλεκτρονικής διεύθυνσης ενός ατόμου ή ενός αριθμού FAX. Ή ένας κατάλογος που αποτελείται από εξυπηρετητές εφαρμογής μπορεί να αναζητήσει στα περιεχομενά του έναν εξυπηρετητή (server) ο οποίος παρέχει πρόσβαση σε πληροφορίες τιμολόγησης πελατών.

Μια υπηρεσία καταλόγου (directory service) αποθηκεύει πληροφορίες στον κατάλογο και ανακτά πληροφορίες από αυτόν εξ ονόματος ενός ή περισσοτέρων εξουσιοδοτημένων χρηστών. Οι υπηρεσίες καταλόγου υλοποιούνται με σκοπό την παροχή ορισμένων τύπων πληροφορίας για τις ανάγκες εφαρμογών. Ωστόσο, διαφορετικές υπηρεσίες καταλόγου μπορούν να μοιράζονται τον ίδιο κατάλογο. Για παράδειγμα, ας υποθέσουμε ότι διαθέτουμε δυο τηλεφωνικούς κατάλόγους, έναν White Pages κατάλογο και έναν Yellow Pages κατάλογο. Και οι δύο διαθέτουν τηλεφωνικούς αριθμούς αλλά με διαφορετικό τρόπο. Ο κατάλογος των «λευκών σελίδων» μας δίνει την δυνατότητα να βρούμε τον τηλεφωνικό αριθμό ενός ατόμου, ενώ ο κατάλογος των «κίτρινων σελίδων» μας επιτρέπει να αναζητούμε κατηγορίες πληροφοριών και να ανακτούμε ποικίλους τηλεφωνικούς αριθμούς.

Για την εφαρμογή των δύο υπηρεσιών μπορεί να χρησιμοποιηθεί ο ίδιος κατάλογος. Στο παράδειγμα αυτό, ο συγκεκριμένος κατάλογος θα περιείχε το μοντέλο των δεδομένων για να περιγράψει τον διαφορετικό τύπο χρηστών και πληροφοριών. Στην πραγματικότητα υπάρχουν δύο διαφορετικές υπηρεσίες καταλόγου. Η μια δίνει πρόσβαση στα δεδομένα των White Pages και η άλλη στα δεδομένα των Yellow Pages. Εντούτοις, και οι δύο μπορούν να χρησιμοποιούν τον ίδιο κατάλογο. Αυτό που συμβαίνει είναι ότι το μοντέλο δεδομένων της υπηρεσίας «λευκών σελίδων» επεκτείνεται ώστε να ταιριάζει με τις πιο σύνθετες ανάγκες της υπηρεσίας των «κίτρινων σελίδων».Το παράδειγμα αυτό δείχνει επίσης ότι οι υπηρεσίες καταλόγου συνήθως λειτουργούν με ένα συγκεκριμένο τρόπο. Παραδείγματος χάριν δεν μπορούμε να δώσουμε τον τηλεφωνικό αριθμό ενός ατόμου στην υπηρεσία White Pages και να αναμένουμε να μας δοθεί (με εύκολο τρόπο) το άτομο στο οποίο αντιστοιχεί ο αριθμός.

Κατά συνέπεια οι όροι «λευκές» και «κίτρινες» σελίδες χρησιμοποιούνται για να περιγράψουν το πώς χρησιμοποιείται ένας κατάλογος. Εάν το όνομα ενός αντικειμένου (άτομο, εκτυπωτής) είναι γνωστό, τότε τα χαρακτηριστικά του (τηλεφωνικός αριθμός, σελίδες ανά λεπτό) μπορούν να ανακτηθούν. Εάν το όνομα ενός μεμονωμένου αντικειμένου δεν είναι γνωστό, ο κατάλογος μπορεί να ψάξει στις αποθηκευμένες πληροφορίες του για μια λίστα αντικειμένων που καλύπτουν μια ορισμένη απαίτηση. Έτσι, οι κατάλογοι που αποθηκεύονται σε ένα ηλεκτρονικό υπολογιστή εν αντιθέσει με τους τηλεφωνικούς, είναι περισσότερο ευέλικτοι γιατί συνήθως μπορούν να εξερευνηθούν με συγκεκριμένα κριτήρια και όχι από ένα σύνολο προκαθορισμένων κατηγοριών.

1.1.1 Χαρακτηριστικά ενός Καταλόγου

Οι κατάλογοι έχουν πέντε σημαντικά χαρακτηριστικά:

- Η αποθήκευση των πληροφοριών προσαρμόζεται έτσι ώστε οι πληροφορίες να διαβάζονται συχνότερα παρά να ενημερώνονται.
  Οι κατάλογοι είναι κατάλληλοι για την εκτέλεση μεγάλου μήκους αναζητήσεων αλλά δεν ενδείκνυνται για την εφαρμογή λειτουργιών που απαιτούν συχνή ενημέρωση (π.χ συστήματα αεροπορικών κρατήσεων).
- Οι πληροφορίες αποθηκεύονται με τρόπο ιεραρχικό.
  Η δομή των πληροφοριών βασίζεται στις σχέσεις πρώτου ανιόντος-πρώτου κατιόντος (father-child).
- Οι πληροφορίες του καταλόγου βασίζονται στα χαρακτηριστικά.
  Ο κατάλογος αποτελείται από ένα σύνολο αντικειμένων (objects). Καθένα από τα αντικείμενα διαθέτει μια ομάδα χαρακτηριστικών (attributes) τα οποία περιέχουν πληροφορίες. Κατά συνέπεια, η πληροφορία διοχετεύεται στα χαρακτηριστικά των αντικειμένων τα οποία σχηματίζουν την υποδομή του καταλόγου.
- Οι κατάλογοι παρέχουν ένα ενοποιημένο namespace για όλους τους πόρους για τους οποίους διαθέτουν πληροφορία.
  Οι κοινές πληροφορίες εντοπίζονται και διαμοιράζονται από διαφορετικούς πελάτες καταλόγου επειδή κάθε εφαρμογή μπορεί να χρησιμοποιήσει την ίδια μέθοδο παραπομπής ενός αντικειμένου. Ένα ενοποιημένο namespace δίνει την δυνατότητα στα στοιχεία του δικτύου και των υπηρεσιών να ενοποιούνται με άλλους τύπους πληροφοριών, όπως χρήστες, εφαρμογές και servers.
- Οι κατάλογοι μπορούν να διανείμουν αποτελεσματικά την πληροφορία σε ένα κατανεμημένο σύστημα μέσω της πολλαπλότητας υλικού.
  Ο server καταλόγου έχει την ικανότητα να ελέγχει το είδος της πληροφορίας που κατανέμεται, τη χρονική στιγμή που συμβαίνει αυτό και σε ποιους κόμβους του συστήματος.

1.1.2 Directories vs. Databases

Ένας κατάλογος συχνά περιγράφεται ως Βάση Δεδομένων αλλά είναι μια εξειδικευμένη Βάση Δεδομένων η οποία έχει χαρακτηριστικά που την κάνουν να ξεχωρίζει από το γενικό σκοπό των Σχεσιακών Βάσεων Δεδομένων. Ένα ειδικό χαρακτηριστικό των καταλόγων είναι ότι προσπελάζονται (μέσω της ανάγνωσης ή της αναζήτησης) πολύ πιο συχνά απ' ότι ενημερώνονται (μέσω της εγγραφής). Εκατοντάδες άνθρωποι μπορούν να ανατρέξουν στα χαρακτηριστικά ενός τηλεφωνικού αριθμού ή χιλιάδες χρήστες εκτυπωτών ίσως να ανατρέξουν στα χαρακτηριστικά ενός συγκεκριμένου εκτυπωτή. Όμως ο τηλεφωνικός αριθμός ή τα χαρακτηριστικά του εκτυπωτή σπάνια αλλάζουν.

Επειδή οι κατάλογοι πρέπει να είναι σε θέση να υποστηρίζουν μεγάλο όγκο αιτημάτων ανάγνωσης, είναι βελτιστοποιημένοι για πρόσβαση ανάγνωσης. Η πρόσβαση εγγραφής μπορεί να επιτρέπεται στους διαχειριστές συστήματος ή στους δικαιούχους κάθε μέρους της πληροφορίας. Από την άλλη πλευρά, ο γενικός σκοπός της Βάσης Δεδομένων χρειάζεται να υποστηρίζει εφαρμογές όπως αεροπορικές κρατήσεις και τραπεζικές συναλλαγές με μεγάλο βαθμό ενημέρωσης.

Επειδή οι κατάλογοι προορίζονταν για την αποθήκευση σχετικά στατικών πληροφοριών και συνεχίζουν να παραμένουν σε αυτόν τον σκοπό, δεν είναι κατάλληλοι για αποθήκευση πληροφορίας που διαρκώς μεταβάλλεται. Για παράδειγμα, ο αριθμός των εργασιών που βρίσκονται προσωρινά στην ουρά ενός εκτυπωτή πιθανόν να μην πρέπει να αποθηκεύεται στον κατάλογο του εκτυπωτή επειδή οι συγκεκριμένες πληροφορίες θα έπρεπε να ενημερώνονται συχνά για να μπορεί να επαληθεύεται η ορθότητά τους. Αντί για αυτό, η καταχώρηση καταλόγου του εκτυπωτή θα μπορούσε να περιέχει την διεύθυνση δικτύου ενός εκτυπωτή server. Ο εκτυπωτής server θα μπορούσε να ερωτηθεί ώστε να γίνει γνωστό το μήκος της τρέχουσας ουράς. Η πληροφορία που βρίσκεται στον κατάλογο (η διεύθυνση του εκτυπωτή server) είναι στατική, ενώ ο αριθμός των εργασιών στην ουρά εκτύπωσης είναι δυναμικός.

Μια άλλη σημαντική διαφορά μεταξύ των καταλόγων και των -γενικού σκοπού- Βάσεων Δεδομένων είναι ότι οι κατάλογοι μπορεί να μην υποστηρίζουν δοσοληψίες. Οι δοσοληψίες είναι διαδικασίες που συντελούνται συνολικά ή καθόλου. Για παράδειγμα, κατά την μεταφορά χρηματικού ποσού από έναν τραπεζικό λογαριασμό σε έναν άλλο, τα χρήματα πρέπει να χρεωθούν πρώτα στον έναν λογαριασμό και έπειτα στον άλλο και όλα αυτά να γίνουν σε μια και μόνο δοσοληψία. Εάν ολοκληρωθεί μόνο το μισό μέρος αυτής της δοσοληψίας ή κάποιος παρέμβει στους λογαριασμούς κατά την διάρκεια μεταφοράς των χρημάτων, οι λογαριασμοί δεν θα είναι ισορροπημένοι. Συνήθως οι -γενικού σκοπού- Βάσεις Δεδομένων υποστηρίζουν τέτοιου είδους δοσοληψίες οι οποίες κάνουν δύσκολη την εφαρμογή τους. Επειδή οι κατάλογοι ασχολούνται περισσότερο με αιτήματα ανάγνωσης, οι δυσκολίες των δοσοληψιών μπορούν να αποφευχθούν. Εάν δυο άτομα ανταλλάξουν γραφεία, θα πρέπει να ενημερωθούν οι καταχωρήσεις καταλόγου τους με νέους τηλεφωνικούς αριθμούς, με τις καινούριες θέσεις των σταθμών εργασίας και ούτω καθεξής. Εάν ενημερωθεί μια καταχώρηση καταλόγου και στη συνέχεια μια άλλη, υπάρχει μια σύντομη χρονική περίοδος που ο κατάλογος θα δείξει ότι και τα δύο άτομα έχουν τον ίδιο αριθμό τηλεφώνου. Επειδή οι ενημερώσεις είναι σχετικά σπάνιες, τέτοιες ανωμαλίες θεωρούνται αποδεκτές.

Το είδος των πληροφοριών που αποθηκεύεται σε έναν κατάλογο συνήθως δεν απαιτεί ακριβή συνέπεια. Μπορεί να είναι ανεκτό το γεγονός ότι μια πληροφορία, όπως ένας τηλεφωνικός αριθμός, ίσως να βρίσκεται προσωρινά εκτός λειτουργίας. Επειδή οι κατάλογοι δεν είναι κατάλληλοι για δοσοληψίες, δεν είναι καλή ιδέα να χρησιμοποιούνται για την αποθήκευση πληροφοριών που είναι ευαίσθητες σε ασυνέπειες όπως οι ισορροπίες τραπεζικών λογαριασμών.

Επειδή οι -γενικού σκοπού- Βάσεις Δεδομένων πρέπει να υποστηρίζουν αυθαίρετες εφαρμογές όπως τον έλεγχο τραπεζικών εργασιών, επιτρέπουν να αποθηκεύονται αυθαίρετες συλλογές δεδομένων. Οι κατάλογοι μπορούν να περιορίζονται στον τύπο των δεδομένων που επιτρέπουν για αποθήκευση (αν και η αρχιτεκτονική δεν επιβάλλει έναν τέτοιο περιορισμό). Για παράδειγμα, ένας κατάλογος που ειδικεύεται στις πληροφορίες επικοινωνίας πελατών μπορεί να περιοριστεί μόνο στην αποθήκευση προσωπικών πληροφοριών όπως ονομάτων, διευθύνσεων και τηλεφωνικών αριθμών. Εάν ένας κατάλογος είναι επεκτάσιμος,

μπορεί να διαμορφωθεί με τρόπο που να αποθηκεύει ποικίλους τύπους πληροφοριών που τον καθιστούν πιο χρήσιμο σε πολλά και διαφορετικά προγράμματα.

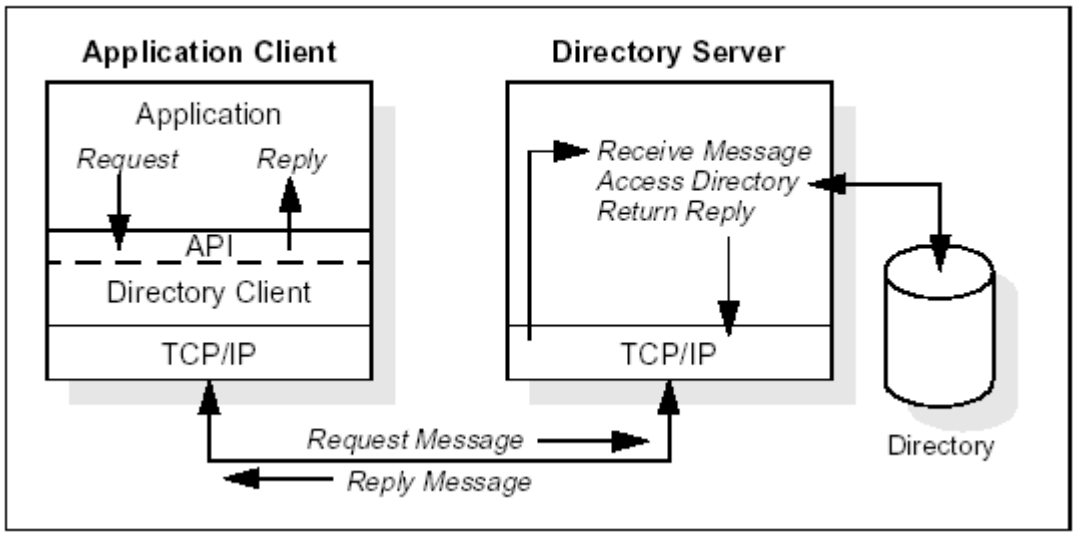
Μια άλλη σημαντική διαφορά μεταξύ των καταλόγων και των –γενικού σκοπού- Βάσεων Δεδομένων βρίσκεται στον τρόπο με τον οποίο προσεγγίζονται οι πληροφορίες. Οι περισσότερες Βάσεις Δεδομένων υποστηρίζουν μια ισχυρή τυποποιημένη μέθοδο πρόσβασης που ονομάζεται Δομημένη Γλώσσα Ερωτημάτων (SQL). Η SQL επιτρέπει πολύπλοκες ενημερώσεις και συναρτήσεις ερωτημάτων με κόστος το μέγεθος των προγραμμάτων και την πολυπλοκότητα των εφαρμογών. Από την άλλη πλευρά, οι κατάλογοι LDAP χρησιμοποιούν ένα απλουστευμένο και βελτιωμένο πρωτόκολλο πρόσβασης το οποίο μπορεί να εφαρμοσθεί σε μικρές και σχετικά απλές εφαρμογές.

Επειδή οι κατάλογοι δεν είναι προορισμένοι να παρέχουν τόσες λειτουργίες όσες προσφέρουν οι Βάσεις Δεδομένων, μπορούν να βελτιωθούν με οικονομικό τρόπο ώστε να παρέχουν περισσότερες λειτουργίες με άμεση πρόσβαση στα δεδομένα του καταλόγου σε μεγάλα κατανεμημένα συστήματα. Επειδή η εκτενής χρήση των καταλόγων είναι προορισμένη περισσότερο για ανάγνωση και όχι για δοσοληψίες μπορούν κατά συνέπεια να απλοποιηθούν ο πελάτης (client) και ο εξυπηρετητής (server) καταλόγου.

Πολλές διαφορές από τις οποίες αναφέρθηκαν μπορούν να οδηγήσουν στην υποψία ότι ένας κατάλογος δεν είναι κάτι περισσότερο από μια περιορισμένη σε λειτουργία Βάση Δεδομένων. Αυτό είναι πράγματι εν μέρει σωστό δεδομένου ότι ένας από τους σημαντικότερους σχεδιαστικούς στόχους μιας υπηρεσίας καταλόγου είναι ότι μπορεί να προσπελασθεί και να χρησιμοποιηθεί από σχετικά μικρές και απλές εφαρμογές. Επίσης συζητούνται προτάσεις στους οργανισμούς προτύπων για την πρόσθεση μερικών λειτουργιών στις μελλοντικές εκδόσεις του LDAP που ειδικεύονται στις Βάσεις Δεδομένων όπως η υποστήριξη των ενημερώσεων των δοσοληψιών.

1.1.3 Πελάτες και Εξυπηρετητές Καταλόγου

Οι κατάλογοι συνήθως προσπελάζονται με την χρησιμοποίηση του μοντέλου επικοινωνίας πελάτη / εξυπηρετητή (client/server). Μια εφαρμογή που επιθυμεί να διαβάσει ή να εγγράψει πληροφορία σε έναν κατάλογο δεν προσπελαύνει άμεσα τον κατάλογο. Αντί για αυτό καλεί μια λειτουργία, ένα API (Application Programming Interface), που αναγκάζει ένα μήνυμα να σταλεί σε μια άλλη διαδικασία. Η δεύτερη αυτή διαδικασία έχει πρόσβαση στην πληροφορία καταλόγου εξ ονόματος της εφαρμογής αίτησης. Έπειτα, τα αποτελέσματα της ανάγνωσης ή της εγγραφής επιστρέφονται στην εφαρμογή αίτησης (βλέπε σχήμα 1.1.3).



Σχήμα 1.1.3 Αλληλεπίδραση client/server καταλόγου

Το αίτημα εκτελείται από τον πελάτη καταλόγου και η διαδικασία αναζήτησης πληροφοριών σε έναν κατάλογο ονομάζεται εξυπηρετητής καταλόγου. Γενικά, οι εξυπηρετητές παρέχουν μια συγκεκριμένη υπηρεσία στους πελάτες. Μερικές φορές ένας εξυπηρετητής μπορεί να γίνει ο πελάτης άλλων εξυπηρετητών προκειμένου να συγκεντρώσει τις απαραίτητες πληροφορίες για να επεξεργαστεί ένα αίτημα.

Μια υπηρεσία καταλόγου είναι μια μόνο μορφή υπηρεσίας που μπορεί να είναι διαθέσιμη σε περιβάλλον client/server. Άλλα συνηθισμένα παραδείγματα υπηρεσιών είναι οι υπηρεσίες αρχείων, οι υπηρεσίες ηλεκτρονικού ταχυδρομείου, οι υπηρεσίες εκτύπωσης, οι υπηρεσίες ιστοσελίδων και ούτω καθεξής. Οι διαδικασίες του client και του server μπορούν να συντελούνται στο ίδιο μηχάνημα ή και όχι. Ένας server είναι σε θέση να εξυπηρετεί πολλούς πελάτες. Μερικοί servers μπορούν να επεξεργάζονται αιτήματα πελατών παράλληλα. Κάποιοι άλλοι βάζουν σε σειρά τα εισερχόμενα αιτήματα των clients για σειριακή επεξεργασία σε περίπτωση που είναι ήδη απασχολημένοι με την επεξεργασία άλλων αιτημάτων.

Ένα API καθορίζει την διεπαφή προγραμματισμού, μια ιδιαίτερη γλώσσα προγραμματισμού, που χρησιμοποιείται για την πρόσβαση σε μια υπηρεσία. Το format και τα περιεχόμενα των μηνυμάτων που ανταλλάσσονται μεταξύ του client και του server πρέπει να εμμένουν πάνω σε ένα συμφωνημένο πρωτόκολλο. Το LDAP καθορίζει ένα πρωτόκολλο μηνύματος που χρησιμοποιείται από τους clients και servers καταλόγου. Υπάρχει επίσης ένα σχετικό LDAP API για την γλώσσα προγραμματισμού C καθώς και τρόποι με τους οποίους μπορούμε να έχουμε πρόσβαση στο LDAP μέσω μιας εφαρμογής JAVA. Ο client δεν εξαρτάται από μια συγκεκριμένη εφαρμογή του server και ο server μπορεί να εφαρμόσει τον κατάλογο όπως αυτός επιλέξει.

1.1.4 Κατανεμημένοι Κατάλογοι

Οι όροι «τοπικός», «παγκόσμιος», «συγκεντρωτικός» και «κατανεμημένος» συχνά χρησιμοποιούνται για να περιγράψουν έναν κατάλογο ή μια υπηρεσία καταλόγου.

Γενικά, με τον όρο «τοπικός» εννοούμε κάτι το οποίο βρίσκεται κοντά ενώ με τον όρο «παγκόσμιος» κάτι το οποίο εξαπλώνεται κατά μήκος του περιβάλλοντος που μας ενδιαφέρει. Το περιβάλλον που μας ενδιαφέρει μπορεί να είναι μια εταιρεία, μια χώρα ή ολόκληρη η Γη. Οι όροι «τοπικός» και «παγκόσμιος» είναι τα δύο άκρα μιας συνέχειας. Δηλαδή κάτι μπορεί να είναι περισσότερο ή λιγότερο τοπικό ή παγκόσμιο από κάτι άλλο. Συγκεντρωτικό σημαίνει κάτι το οποίο βρίσκεται σε ένα μέρος και κατανεμημένο κάτι το οποίο βρίσκεται σε περισσότερα από ένα μέρη. Ανάλογα με το τοπικό και το παγκόσμιο, κάτι μπορεί να είναι κατανεμημένο σε μικρότερη ή σε μεγαλύτερη έκταση.

Η αποθηκευμένη πληροφορία σε έναν κατάλογο μπορεί να είναι τοπική ή σφαιρική. Για παράδειγμα, ένας κατάλογος ο οποίος αποθηκεύει τοπική πληροφορία μπορεί να περιέχει ονόματα, διευθύνσεις ηλεκτρονικού ταχυδρομείου, δημόσια κλειδιά κρυπτογράφησης και ούτω καθεξής, τα οποία αφορούν τα μέλη ενός τμήματος ή μια ομάδας εργαζομένων. Ένας κατάλογος που αποθηκεύει σφαιρική πληροφορία μπορεί να αποθηκεύσει πληροφορία για μια ολόκληρη εταιρεία. Στην προκείμενη περίπτωση μας ενδιαφέρει η εταιρεία. Οι πελάτες που έχουν πρόσβαση στις πληροφορίες καταλόγου μπορεί να είναι τοπικοί ή παγκόσμιοι. Οι τοπικοί μπορεί να βρίσκονται όλοι στο ίδιο κτίριο ή στο ίδιο τοπικό δίκτυο (LAN). Οι παγκόσμιοι μπορεί να είναι κατανεμημένοι κατά πλάτος της ηπείρου ή του πλανήτη.

Ο ίδιος κατάλογος μπορεί να είναι συγκεντρωτικός ή κατανεμημένος. Εάν ένας κατάλογος είναι συγκεντρωτικός, υπάρχει ένας server καταλόγου που παρέχει πρόσβαση στον κατάλογο ενώ αν είναι κατανεμημένος υπάρχουν περισσότεροι από έναν servers που δίνουν πρόσβαση στον κατάλογο. Όταν αναφερόμαστε σε έναν κατανεμημένο κατάλογο, συνήθως εννοούμε τους κατανεμημένους servers καταλόγου. Όταν ένας κατάλογος είναι κατανεμημένος η αποθηκευμένη πληροφορία του μπορεί να είναι χωρισμένη σε τμήματα ή να έχει αντίγραφα. Στην περίπτωση που η πληροφορία έχει την μορφή κατηγοριών, κάθε server καταλόγου αποθηκεύει ένα μοναδικό και μη επικαλυπτόμενο υποσύνολο της πληροφορίας. Δηλαδή κάθε καταχώρηση καταλόγου είναι αποθηκευμένη από έναν και μόνο έναν server. Όταν υπάρχουν αντίγραφα της πληροφορίας η ίδια καταχώρηση αποθηκεύεται σε περισσότερους από έναν servers. Σε έναν κατανεμημένο κατάλογο, κάποιες πληροφορίες μπορούν να είναι χωριστές και κάποιες να έχουν αντίγραφα.

Οι τρεις ‘’διαστάσεις’’ ενός καταλόγου – πεδίο της πληροφορίας, θέση των clients και διανομή των servers – είναι ανεξάρτητες μεταξύ τους. Παραδείγματος χάριν, οι πελάτες που είναι διασκορπισμένοι παγκοσμίως θα μπορούσαν να έχουν πρόσβαση σε έναν κατάλογο που περιέχει πληροφορία για ένα μόνο τμήμα και αυτός ο κατάλογος θα μπορούσε να αντιγραφθεί για πολλούς servers καταλόγου. Ή πελάτες που βρίσκονται σε μια μόνο θέση θα μπορούσαν να προσπελάσουν έναν κατάλογο ο οποίος περιέχει πληροφορίες για οποιονδήποτε στον κόσμο που είναι αποθηκευμένες από έναν μόνο server.

Το πεδίο των πληροφοριών που αποθηκεύεται σε έναν κατάλογο συχνά αποδίδεται ως μια απαίτηση εφαρμογής. Η διανομή των servers καταλόγου και ο τρόπος με τον οποίο τα δεδομένα χωρίζονται ή αντιγράφονται μπορούν συχνά να ρυθμίζονται για να επηρεάζουν την απόδοση και την διαθεσιμότητα του καταλόγου. Για παράδειγμα, ένας κατανεμημένος και αντιγραμμένος κατάλογος μπορεί να έχει καλύτερη απόδοση επειδή ένα αίτημα ανάγνωσης μπορεί να εξυπηρετηθεί από έναν κοντινό server. Ένας συγκεντρωτικός κατάλογος μπορεί να είναι λιγότερο διαθέσιμος επειδή σε περίπτωση αποτυχίας δεν θα μπορεί να δώσει πληροφορίες δεδομένου ότι είναι μοναδικός και δεν υπάρχει αντίγραφο του. Ωστόσο, ένας κατανεμημένος κατάλογος ίσως να είναι πιο δύσκολο να διατηρηθεί, εξ αιτίας των πολλών θέσεων

που πιθανόν βρίσκονται κάτω από τον έλεγχο πολλών διαχειριστών και που πρέπει να παραμένουν ενημερωμένες και σε κατάσταση λειτουργίας. Ο σχεδιασμός και η συντήρηση μιας υπηρεσίας καταλόγου μπορεί να είναι πολύπλοκες διαδικασίες.

1.2 Ο Κατάλογος ως Υποδομή

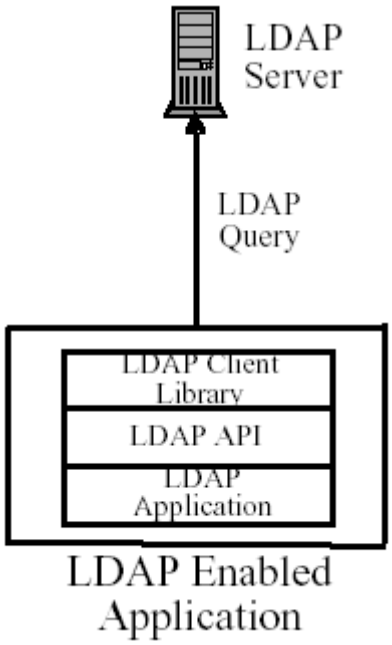
Ένας κατάλογος ο οποίος προσπελαύνεται από όλες τις εφαρμογές αποτελεί ένα ζωτικής σημασίας μέρος της υποδομής που υποστηρίζει ένα κατανεμημένο σύστημα. Μια υπηρεσία καταλόγου προσφέρει μια ενιαία λογική όψη των χρηστών, των πόρων, και άλλων αντικειμένων που συνθέτουν ένα κατανεμημένο σύστημα. Αυτό επιτρέπει στους χρήστες και στις εφαρμογές να έχουν πρόσβαση στους δικτυακούς πόρους με τρόπο διαφανή. Δηλαδή, το σύστημα αντιμετωπίζεται σαν ένα σύνολο και όχι σαν μια συλλογή από ανεξάρτητα μέρη. Τα αντικείμενα μπορούν να προσεγγίζονται με το όνομα ή με μια λειτουργία χωρίς την γνώση προσδιοριστικών όπως οι host διευθύνσεις, ονόματα αρχείων εξυπηρετητών και διευθύνσεις ηλεκτρονικού ταχυδρομείου.

1.2.1 Διαθέσιμες Εφαρμογές Καταλόγου

Μια διαθέσιμη εφαρμογή καταλόγου είναι η εφαρμογή που χρησιμοποιεί μια υπηρεσία καταλόγου για να βελτιώσει την λειτουργικότητα της, την χρήση της και την διαχείρισή της. Σήμερα πολλές εφαρμογές κάνουν χρήση των πληροφοριών που θα μπορούσαν να αποθηκευθούν σε ένα κατάλογο. Για παράδειγμα, ας θεωρήσουμε μια εφαρμογή ημερολογίου που χρησιμοποιείται με σκοπό την οργάνωση συνεδριάσεων του προσωπικού μιας εταιρείας σε διαφορετικές αίθουσες διασκέψεων. Στην χειρότερη περίπτωση, η εφαρμογή ημερολογίου δεν χρησιμοποιεί καμία υπηρεσία καταλόγου. Εάν αυτό συνέβαινε, ο χρήστης που θα προσπαθούσε να προγραμματίσει ένα meeting θα έπρεπε να θυμάται τον αριθμό κάθε αίθουσας που θα ήταν κατάλληλη για το meeting. Και ακόμα θα αναρωτιόταν αν η αίθουσα είναι αρκετά μεγάλη, αν διαθέτει τον απαραίτητο ακουστικό και οπτικό εξοπλισμό και ούτω καθεξής. Επίσης ο χρήστης θα έπρεπε να θυμάται τα ονόματα και τις διευθύνσεις e-mail κάθε συμμετέχοντος που χρειάζεται να λάβει μια σημείωση συνεδρίασης. Είναι προφανές ότι μια τέτοια εφαρμογή θα ήταν δύσκολο να χρησιμοποιηθεί. Εάν οι πληροφορίες για τις αίθουσες διάσκεψης (μέγεθος, τοποθεσία, ειδικός εξοπλισμός) και οι πληροφορίες προσωπικού (ονόματα, διευθύνσεις e-mail, τηλεφωνικοί αριθμοί και λοιπά) μπορούσαν να προσπελάζονται μέσω μιας υπηρεσίας καταλόγου, η εφαρμογή θα ήταν πολύ πιο εύκολο να χρησιμοποιηθεί. Επιπλέον, θα βελτιωνόταν η λειτουργικότητα της εφαρμογής. Παραδείγματος χάριν, θα μπορούσε να παρουσιάζεται στον χρήστη μια λίστα με όλες τις διαθέσιμες αίθουσες συνεδριάσεων καθώς και το μέγεθος και οι απαιτήσεις εξοπλισμού τους.

Όμως οι υπεύθυνοι ανάπτυξης των διαθέσιμων εφαρμογών καταλόγου βρίσκονται αντιμέτωποι με ένα πρόβλημα. Τι θα συμβεί εάν δεν λάβουν υπόψη ότι μια υπηρεσία καταλόγου θα υπάρξει σε όλα τα περιβάλλοντα; Εάν υπάρχει μια υπηρεσία καταλόγου, αυτή μπορεί να ειδικεύεται σε ένα συγκεκριμένο λειτουργικό

σύστημα δικτύου (NOS), καθιστώντας την εφαρμογή μη εκτελέσιμη. Μπορεί η υπάρχουσα υπηρεσία καταλόγου να διευρυνθεί για να αποθηκεύσει τον τύπο των πληροφοριών που χρειάζονται από την εφαρμογή; Λόγω αυτών των ανησυχιών, οι υπεύθυνοι ανάπτυξης εφαρμογών συχνά υιοθετούν την μέθοδο ανάπτυξης του δικού τους καταλόγου που θα υποστηρίζει μια συγκεκριμένη εφαρμογή.



1.2.2 Τα Πλεονεκτήματα ενός Κοινού Καταλόγου

Ένας κατάλογος που υποστηρίζει μια συγκεκριμένη εφαρμογή αποθηκεύει μόνο την πληροφορία που χρειάζεται αυτή η εφαρμογή και η οποία δεν μπορεί να προσπελαστεί από άλλες εφαρμογές. Επειδή είναι δύσκολο να στηθεί μια πλήρης λειτουργικά υπηρεσία καταλόγου, οι κατάλογοι ειδικών εφαρμογών είναι πολύ λίγοι. Πιθανόν να αποθηκεύουν μόνο ένα συγκεκριμένο τύπο πληροφορίας και ίσως να μην διαθέτουν δυνατότητες αναζήτησης ούτε την υποστήριξη της αντιγραφής και διαμοίρασης της πληροφορίας και να μην έχουν ένα πλήρες σύνολο εργαλείων διαχείρισης. Ένας κατάλογος μιας συγκεκριμένης εφαρμογής μπορεί να είναι τόσο απλός όσο ένα σύνολο αρχείων κειμένου ή θα μπορούσε να αποθηκευθεί και να προσεγγίζεται με έναν ατεκμηρίωτο, ιδιόκτητο τρόπο.

Σε έναν τέτοιο περιβάλλον, κάθε εφαρμογή δημιουργεί και διαχειρίζεται το δικό της κατάλογο. Αυτό γρήγορα γίνεται ένας εφιάλτης διαχείρισης. Η ίδια διεύθυνση e-mail που αποθηκεύεται από την εφαρμογή ημερολογίου μπορεί επίσης να αποθηκευθεί από μια εφαρμογή ταχυδρομείου και από μια εφαρμογή που ειδοποιεί τους χειριστές συστήματος για προβλήματα εξοπλισμού. Η προσπάθεια να διατηρούνται ενήμερα και συγχρονισμένα τα αντίγραφα των πληροφοριών είναι δύσκολη, ειδικά όταν αναμιγνύονται διαφορετικές διεπαφές χρηστών και διαφορετικά συστήματα διαχείρισης.

Αυτό που χρειάζεται είναι ένας κοινός, ανεξάρτητος από εφαρμογές, κατάλογος. Αν οι υπεύθυνοι ανάπτυξης εφαρμογών μπορούσαν να βεβαιωθούν για την ύπαρξη μιας υπηρεσίας καταλόγου, τότε οι κατάλογοι ειδικών εφαρμογών δεν θα

ήταν απαραίτητοι. Ωστόσο, ένας κοινός κατάλογος πρέπει να βασίζεται σε ένα ανοιχτό πρότυπο το οποίο να υποστηρίζεται από πολλούς προμηθευτές σε πολλές πλατφόρμες και που πρέπει να προσεγγίζεται μέσω ενός πρότυπου API. Είναι απαραίτητο να είναι επεκτάσιμο έτσι ώστε να μπορεί να κρατήσει τους τύπους των δεδομένων που χρειάζονται από τις αυθαίρετες δοκιμές. Και ακόμα πρέπει να παρέχει πλήρη λειτουργικότητα χωρίς να απαιτεί υπερβολικούς πόρους από τα μικρότερα συστήματα. Δεδομένου ότι περισσότεροι χρήστες και εφαρμογές θα έχουν πρόσβαση στον κοινό κατάλογο και θα εξαρτώνται από αυτόν, πρέπει επίσης να είναι εύρωστος, ασφαλής και με δυνατότητες εξέλιξης.

Όταν μια τέτοια υποδομή καταλόγου βρίσκεται σε ισχύ, οι υπεύθυνοι ανάπτυξης εφαρμογών μπορούν να αφιερώσουν τον χρόνο τους στην ανάπτυξη εφαρμογών και όχι στην δημιουργία καταλόγων που υποστηρίζουν συγκεκριμένες εφαρμογές. Με αυτόν τον τρόπο, οι υπεύθυνοι ανάπτυξης εφαρμογών που βασίζονται στην επικοινωνιακή δομή του TCP/IP και στην απομακρυσμένη διαδικασία κλήσης (RPC), θα είναι σε θέση να στηριχτούν σε ισχυρές, πλήρους λειτουργίας υπηρεσίες καταλόγου. Το LDAP είναι το πρωτόκολλο που πρέπει να χρησιμοποιηθεί για την πρόσβαση σε αυτή την κοινή υποδομή καταλόγου. Όπως το HTTP (Hypertext Transfer Protocol) και το FTP (File Transfer Protocol), έτσι και το LDAP είναι ένα απαραίτητο μέρος της ακολουθίας πρωτοκόλλων του Internet.

Όταν οι εφαρμογές έχουν πρόσβαση σε έναν πρότυπο κοινό κατάλογο παρά σε έναν κατάλογο που υποστηρίζει εξειδικευμένες εφαρμογές, τότε η περιττή και δαπανηρή διαχείριση μπορεί να εξαλειφθεί και οι κίνδυνοι ασφάλειας να είναι περισσότερο ελεγχόμενοι. Οι εφαρμογές ημερολογίου, ταχυδρομείου και ενημερώσεων των χειριστών μπορούν να προσπελαύνουν τον ίδιο κατάλογο για να ανακτήσουν ένα δεδομένο όπως για παράδειγμα μια διεύθυνση e-mail. Θα εμφανιστούν νέες χρήσεις του καταλόγου πληροφοριών και θα αναπτυχθεί μια σύμπραξη καθώς περισσότερες εφαρμογές θα επωφεληθούν από τον κοινό κατάλογο.

1.3 Πρότυπα και Ιστορία του LDAP

Κατά τη δεκαετία του 1970, η ολοκλήρωση των επικοινωνιών και των τεχνολογιών πληροφορικής οδήγησε στην ανάπτυξη νέων τεχνολογιών επικοινωνίας. Πολλά από τα ιδιωτικά συστήματα που αναπτύχθηκαν ήταν μη συμβατά με άλλα συστήματα. Έγινε προφανές ότι χρειάζονταν πρότυπα για να επιτρέψουν στους εξοπλισμούς και στα συστήματα διαφορετικών προμηθευτών να επικοινωνήσουν μεταξύ τους. Για τον καθορισμό τέτοιων προτύπων αναπτύχθηκαν δυο κορυφαίες ανεξάρτητες προσπάθειες τυποποίησης.

1.3.1 ΤΟ OSI και το Internet

Μια κίνηση προτύπων έγινε από την CCITT (*Comite Consultatif International Telephonique et Telegrafique,* ή *Consultative Committee on International Telephony and Telegraphy*) και τον ISO (*International Standards Organization*). Η CCITT έχει γίνει από τότε η ITU-T (*International Telecommunications Union- Telecommunication Standardization Sector*). Η προσπάθεια αυτή οδήγησε στο πρότυπο αναφοράς (ISO

7498) του OSI (Open Systems Interconnect), το οποίο καθόρισε ένα μοντέλο δεδομένων επικοινωνίας επτά επιπέδων με φυσική μεταφορά στο χαμηλότερο στρώμα και των πρωτοκόλλων εφαρμογής στα ανώτερα επίπεδα.

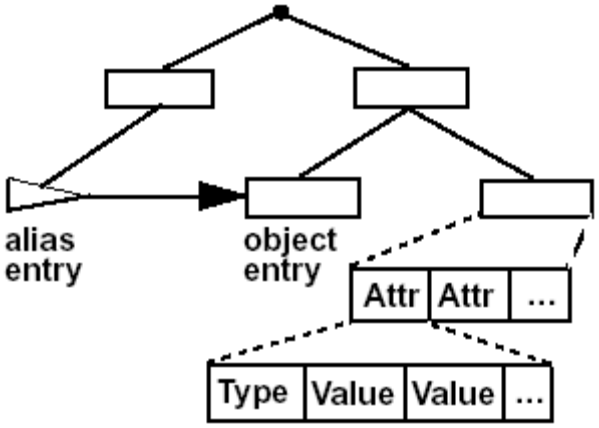
Η άλλη κίνηση προτύπων έγινε γύρω από το Internet και αναπτύχθηκε από την έρευνα που υποστηρίχθηκε από το DARPA (*the Defense Advanced Research Projects Agency*) στις Ηνωμένες Πολιτείες. Ο *Internet Architecture Board* (IAB) και η IETF (*Internet Engineering Task Force*) αναπτύσσουν πρότυπα για το Internet υπό μορφή εγγράφων που ονομάζονται RFCs (Request for Comments) τα οποία αφού πρώτα εγκριθούν, εφαρμοσθούν και χρησιμοποιηθούν για ένα χρονικό διάστημα, τελικά γίνονται πρότυπα (STDs). Πριν μια πρόταση υλοποιηθεί σε RFC καλείται προσχέδιο διαδικτύου (Internet Draft).

Οι δυο διαδικασίες προτύπων προσεγγίζουν την τυποποίηση από δυο διαφορετικές προοπτικές. Η προσέγγιση του OSI άρχισε από μια ξεκάθαρη βάση και καθόρισε πρότυπα χρησιμοποιώντας μια επίσημη διαδικασία επιτροπής χωρίς την απαίτηση εφαρμογών. Το Internet χρησιμοποιεί μια λιγότερο επίσημη προσέγγιση εφαρμοσμένης μηχανικής όπου οποιοσδήποτε μπορεί να προτείνει και να σχολιάσει πάνω στα RFCs ενώ απαιτούνται οι εφαρμογές για να ελεγχθεί ό,τι είναι εφικτό. Τα πρωτόκολλα OSI αναπτύχθηκαν αργά και επειδή τρέχουν ολόκληρη τη σωρό πρωτοκόλλου δεν έχουν ευρέως επεκταθεί ειδικά στους υπολογιστές γραφείου και στην μικρή αγορά υπολογιστών. Στο μεταξύ, το TCP/IP και το Internet αναπτύσσονταν και εφαρμόζονταν ραγδαία. Επίσης, κάποιοι προμηθευτές δικτύου ανέπτυξαν ιδιωτικά πρωτόκολλα δικτύου και άλλα προϊόντα.

1.3.2 X.500: Το Πρότυπο Υπηρεσίας Καταλόγου

Εντούτοις, τα πρωτόκολλα του OSI αντιμετώπισαν σημαντικά ζητήματα στα μεγάλα κατανεμημένα συστήματα που αναπτύσσονταν για την αγορά των ηλεκτρονικών υπολογιστών και του Internet. Ένα τέτοιο σημαντικό πεδίο ήταν οι υπηρεσίες καταλόγου. Η CCITT δημιούργησε το 1988 το πρότυπο X.500 το οποίο έγινε *ISO 9594, Data Communications Network Directory, Recommendations X.500-X.521* το 1990, αν και ακόμα αναφέρεται συνήθως ως X.500.

Το X.500 οργανώνει τις καταχωρήσεις καταλόγου σε ένα ιεραρχικό namespace που είναι ικανό να υποστηρίζει μεγάλες ποσότητες πληροφορίας. Επίσης προσφέρει ισχυρές δυνατότητες αναζήτησης ώστε να γίνεται ευκολότερη η ανάκτηση πληροφοριών. Λόγω της λειτουργίας του και της ικανοτητάς του να εξελίσσεται, το X.500 συχνά χρησιμοποιείται μαζί με πρόσθετες ενότητες (modules) για την διαλειτουργικότητα ανάμεσα σε υπηρεσίες καταλόγου που δεν είναι συμβατές μεταξύ τους. Το X.500 ορίζει ότι η επικοινωνία ανάμεσα στον πελάτη καταλόγου και στον εξυπηρετητή καταλόγου γίνεται με την χρήση του πρωτοκόλλου DAP (Directory Access Protocol). Ωστόσο, ως ένα πρωτόκολλο στρώματος εφαρμογής, το DAP απαιτεί ολόκληρη τη στοίβα του OSI για να λειτουργήσει. Η υποστήριξη της στοίβας OSI χρειάζεται περισσότερους πόρους από όσους είναι διαθέσιμοι σε πολλά μικρά υπολογιστικά περιβάλλοντα. Επομένως έγινε επιθυμητή μια διεπαφή σε έναν X.500 server καταλόγου που θα χρησιμοποιεί λιγότερους πόρους ή αλλιώς προέκυψε η ανάγκη για ένα ‘’ελαφρύτερο’’ πρωτόκολλο.



Σχήμα 1.3.2 Πληροφοριακό μοντέλο X.500. Το μοντέλο X.500 βρίσκεται στο κέντρο και περιτριγυρίζεται από καταχωρήσεις οι οποίες απαρτίζονται από χαρακτηριστικά που έχουν συντακτικούς τύπους και μια ή περισσότερες τιμές. Οι καταχωρήσεις είναι οργανωμένες υπό τη μορφή δένδρου. Οι καταχωρήσεις με ψευδώνυμο (alias) χρησιμοποιούνται για την δημιουργία μη- ιεραρχικών σχέσεων.

1.3.3 LDAP: Η «Ελαφριά» Πρόσβαση στο Internet

Το LDAP αναπτύχθηκε ως μια ελαφριά εναλλακτική λύση για το DAP. Το LDAP απαιτεί την ελαφρύτερη και δημοφιλέστερη στοίβα πρωτοκόλλου του TCP/IP από αυτή του OSI. Επίσης απλοποιεί κάποιες λειτουργίες του X.500 και παραλείπει μερικά εσωτερικά χαρακτηριστικά γνωρίσματα του.

Δυο πρόδρομοι του LDAP παρουσιάστηκαν ως RFCs από την IETF, το DAS (*Directory Assistance Service) (RFC 1202)* και το *DIXIE Protocol Specification (RFC 1249)*. Και τα δυο ήταν ενημερωτικά RFC τα οποία δεν προτάθηκαν ως πρότυπα. Η Υπηρεσία Βοήθειας Καταλόγου (DAS) καθόρισε μια μέθοδο με την οποία ένας πελάτης καταλόγου μπορούσε να επικοινωνήσει με έναν αντιπρόσωπο (proxy) πάνω σε έναν ενεργό host OSI ο οποίος θέτει αιτήματα X.500 εκ μέρους του πελάτη. Το DIXIE είναι παρόμοιο με το DAS, αλλά προσφέρει μια πιο άμεση μετάφραση του DAP.

Η πρώτη έκδοση του LDAP ορίσθηκε ως το Ελαφρύ Πρωτόκολλο Πρόσβασης X.500 (RFC 1487) το οποίο αντικαταστάθηκε από το Ελαφρύ Πρωτόκολλο Πρόσβασης Καταλόγου LDAP (RFC 1777). Το LDAP βελτιώνει περαιτέρω τις ιδέες και τα πρωτόκολλα των DAS και DIXIE. Είναι πιο ουδέτερη εφαρμογή και μειώνει την δυσκολία των πελατών να ενθαρρύνουν την επέκταση των εφαρμογών καταλόγου. Ένα μεγάλο μέρος της εργασίας για το DIXIE και για το LDAP εκτελέσθηκε στο πανεπιστήμιο του Μίσιγκαν το οποίο παρέχει εφαρμογές αναφοράς του LDAP και διατηρεί ιστοσελίδες σχετικές με το LDAP και καταλόγους διευθύνσεων.

Το RFC 1777 καθορίζει το ίδιο το πρωτόκολλο LDAP το οποίο μαζί με

- την αναπαράσταση στοιχειοσειράς των τυποποιημένων συντακτικών των χαρακτηριστικών (*The String Representation of Standard Attribute Syntaxes, RFC 1778)*

- μια αναπαράσταση στοιχειοσειράς των διακριτών ονομάτων *(A String Representation of Distinguished Names, RFC 1779)*

- ένα LDAP URL Format *(RFC 1959)*

- μια αναπαράσταση στοιχειοσειράς των φίλτρων αναζήτησης LDAP *(A String Representation of LDAP Search Filters, RFC 1960)*

καθορίζει την έκδοση 2 του LDAP (LDAP v. 2).

Η δεύτερη έκδοση έχει φθάσει στην κατάσταση προσχεδίου σύμφωνα με την διαδικασία τυποποίησης της IETF, δηλαδή ένα βήμα πριν να γίνει πρότυπο. Αν και θα μπορούσαν να γίνουν αλλαγές σε ένα προσχέδιο προτύπου, επιδιώκεται πρώτα μια ουσιώδης δοκιμή του. Πολλοί προμηθευτές έχουν εφαρμόσει προϊόντα τα οποία υποστηρίζουν την έκδοση 2 του LDAP. Άλλοι εφαρμόζουν προϊόντα που υποστηρίζουν εξ ολοκλήρου ή μερικώς την έκδοση 3.

Η έκδοση 3 έχει καθοριστεί από το Lightweight Directory Access Protocol (v3) (RFC 2251). Σχετικά RFCs που είναι πρόσφατα ή ενημερωμένα για την έκδοση 3 είναι:

- *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions (RFC 2252)*
- *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names (RFC 2253)*

- *The String Representation of LDAP Search Filters (RFC 2254)*

- *The LDAP URL Format (RFC 2255)*

- *A Summary of the X.500 (96) User Schema for use with LDAPv3 (RFC 2256)*

Το RFC 2251 είναι προτεινόμενο πρότυπο, ένα βήμα πριν γίνει προσχέδιο. Είναι πιθανόν να γίνουν κάποιες μικρές αναθεωρήσεις αλλά επιδιώκεται η δοκιμή από διάφορες ομάδες. Το LDAPv3 επεκτείνει το LDAPv2 στα ακόλουθα σημεία:

Παραπομπές	Ένας server ο οποίος δεν αποθηκεύει τα ζητούμενα δεδομένα μπορεί να παραπέμψει τον client σε έναν άλλο server.
Ασφάλεια	Επικύρωση με την χρησιμοποίηση του μηχανισμού Simple Authentication and Security Layer (SASL).
Διεθνοποίηση	Υποστήριξη UTF-8 για τους διεθνείς χαρακτήρες.
Επεκτασιμότητα	Νέοι τύποι αντικειμένων και λειτουργιών μπορούν να καθορίζονται δυναμικά και το σχήμα (schema) να δημοσιεύεται με τυποποιημένο τρόπο.

Το LDAP ορίζει το πρωτόκολλο επικοινωνίας μεταξύ του πελάτη καταλόγου και του server καταλόγου, όμως δεν καθορίζει μια διεπαφή προγραμματισμού για τον πελάτη. Το *LDAP Application Program Interface* (RFC 1823) καθορίζει ένα API γλώσσας C για να υπάρχει πρόσβαση σε έναν κατάλογο που χρησιμοποιεί το LDAP έκδοσης 2. Αυτό είναι ένα ενημερωτικό RFC, το οποίο σημαίνει ότι δεν είναι ένα επίσημο πρότυπο. Ωστόσο, έχει γίνει ένα *de facto* πρότυπο. Ένα τυποποιημένο πρωτόκολλο και η διάθεση ενός κοινού API για διαφορετικές πλατφόρμες είναι οι κυριότεροι λόγοι που κάνουν ευρέως αποδεκτό το LDAP.

1.3.4 Σύγκριση DAP- LDAP

Η απόδοση του LDAP είναι ικανοποιητική στις περισσότερες εφαρμογές. Στην ενότητα αυτή γίνεται σύγκριση των DAP και LDAP σε τέσσερις τομείς: στον απαιτούμενο χρόνο απόκρισης των ερωτημάτων, στο μέγεθος των ερωτημάτων, στην ταχύτητα κωδικοποίησης PDU και στο μέγεθος και την πολυπλοκότητα των εφαρμογών που συντελούνται από την μεριά του client. Για τις συγκρίσεις αυτές χρησιμοποιήθηκε η εφαρμογή LDAP του Πανεπιστημίου Μίσιγκαν και η εφαρμογή DAP του ISODE. Στο X.500, ο κατάλογος είναι κατανεμημένος μεταξύ πολλών servers και ονομάζεται DSA (Directory System Agent). Ανεξάρτητα με ποιον server είναι συνδεδεμένος ο client, βλέπει την ίδια άποψη του καταλόγου. Εάν ένας server δεν είναι σε θέση να απαντήσει σε ένα αίτημα του client, τότε μπορεί να δρομολογήσει το αίτημα σε έναν άλλο server (*διαδικασία αλυσίδας*) ή να παραπέμψει τον client σε άλλον server (βλ. 2.2.2.2). Έτσι χρησιμοποιήθηκε ο ίδιος DSA για όλες τις μετρήσεις των ερωτημάτων αποκτώντας με αυτό τον τρόπο μια βασική γραμμή σύγκρισης. Ο πίνακας 1.3.4.α' δείχνει την απόδοση μιας σειράς τυπικών ερωτημάτων των DAP και LDAP. Τα τεστ διεξήχθησαν σε ένα μηχάνημα στο οποίο έτρεχαν οι clients των DAP και LDAP, ο LDAP server και ο DSA. Όπως φαίνεται στον πίνακα, η καθυστέρηση που σημείωσε το LDAP είναι ελάχιστη. Η καθυστέρηση αυτή θα μπορούσε να εξαλειφθεί συνολικά από μια εγγενή εφαρμογή DSA που αποβάλλει την ενδιάμεση μετάφραση κωδικοποίησης, αποκωδικοποίησης και την μετάφραση πρωτοκόλλου.

Query	DAP	LDAP
Unauthenticated bind	30	68
Authenticated bind	34	56
Simple search (one entry)	32	41
Simple search (50 entries)	293	353

Πίνακας 1.3.4.α'. Σύγκριση των χρόνων των ερωτημάτων των DAP και LDAP. Οι αναζητήσεις εκτελέσθηκαν με την χρήση του ίδιου DSA. Οι χρόνοι είναι σε ms.

Ο πίνακας 1.3.4.β' δείχνει το μέγεθος των ερωτημάτων. Όπως φαίνεται, τα ερωτήματα LDAP είναι ουσιαστικά μικρότερα από τα ισοδύναμα DAP. Οι εξοικονομήσεις οφείλονται στο απλουστευμένο DN και στις κωδικοποιήσεις των τιμών. Τα μεγέθη των ερωτημάτων είναι επίσης μειωμένα λόγω της απουσίας των ελέγχων υπηρεσίας σε κάθε λειτουργία.

Query	DAP	LDAP
Unauthenticated bind	192	14
Authenticated bind	409	138
Simple search request	237	105
Single search result	547	355

Πίνακας 1.3.4.β'. Σύγκριση των μεγεθών των ερωτημάτων DAP και LDAP. Τα ερωτήματα LDAP είναι σημαντικά μικρότερα από τα αντίστοιχα DAP. Τα μεγέθη των ερωτημάτων είναι σε bytes.

Οι πίνακες 1.3.4.γ' και 1.3.4.δ' παρουσιάζουν τον χρόνο αποκωδικοποίησης και κωδικοποίησης τυπικών DAP και LDAP PDUs. Δείχνουν ότι το LDAP έχει ένα μέτριο πλεονέκτημα απόδοσης για απλά PDUs και ένα ουσιαστικό πλεονέκτημα για σύνθετα PDUs, ειδικά για εκείνα που περιέχουν πολλά διακριτά ονόματα (DNs) όπου η αντιπροσώπευση της στοιχειοσειράς LDAP είναι ένα μεγάλο κέρδος.

PDU Complexity	DAP	LDAP
Simple	550	110
Medium	7,925	714
Complex	38,393	2,702

Πίνακας 1.3.4.γ'. Σύγκριση των χρόνων αποκωδικοποίησης των DAP και LDAP. Τα στοιχεία πρωτοκόλλου LDAP είναι ευκολότερο να αποκωδικοποιηθούν ειδικά για σύνθετα PDUs. Το σύνθετο PDU περιείχε ένα χαρακτηριστικό με περισσότερα από 600 DNs. Περίπου μισός από τον χρόνο αποκωδικοποίησης DAP ξοδεύτηκε σε έναν διπλό έλεγχο για να εξασφαλισθεί ότι μια ιδιότητα διαθέτει μόνο μια από κάθε τιμή.

PDU Complexity	DAP	LDAP
Simple	24	6
Medium	1,084	324
Complex	2656	989

Πίνακας 1.3.4.δ' Σύγκριση των χρόνων κωδικοποίησης DAP και LDAP. Τα στοιχεία πρωτοκόλλου LDAP είναι κωδικοποιημένα πιο αποτελεσματικά, ειδικά για πολύπλοκα PDUs.

Τέλος, έγινε σύγκριση του μεγέθους της εφαρμογής και της πολυπλοκότητας του κώδικα. Μια τέτοια σύγκριση είναι δύσκολο να γίνει διότι πρέπει να αξιολογηθεί το ευρύ φάσμα των δεξιοτήτων και των στόχων των προγραμματιστών που εργάζονται για την παραγωγή των εφαρμογών. Ωστόσο μερικά ευνοϊκά συμπεράσματα για το LDAP μπορούν να προέλθουν από το συντριπτικό πλεονέκτημα που έχει σε αυτόν τον τομέα, όπως φαίνεται στον πίνακα 1.3.4.ε'.

Metric	DAP	LDAP
Total size (DE)	1,484,568	334,552
Text	958,464	221,184
Data	385,024	73,728
BSS	141,080	39,640
Semicolon count	46,746	1,989
If count	9369	568

Πίνακας 1.3.4.ε' Σύγκριση της πολυπλοκότητας εφαρμογής των DAP και LDAP. Ο DE client, ο οποίος μπορεί να δημιουργηθεί είτε με τη χρήση του DAP είτε του LDAP, χρησιμοποιείται για να συγκρίνει το μέγεθος εφαρμογής. Ο semicolon count που προσεγγίζει τον αριθμό προτάσεων και ο αριθμός των προτάσεων «if» που προσεγγίζει τον αριθμό των paths του κώδικα αποτελούν τα δυο μέτρα πολυπλοκότητας. Η σύγκριση έγινε μεταξύ του ISODE- 8.0 και της εφαρμογής LDAP του Πανεπιστημίου Μίσιγκαν.

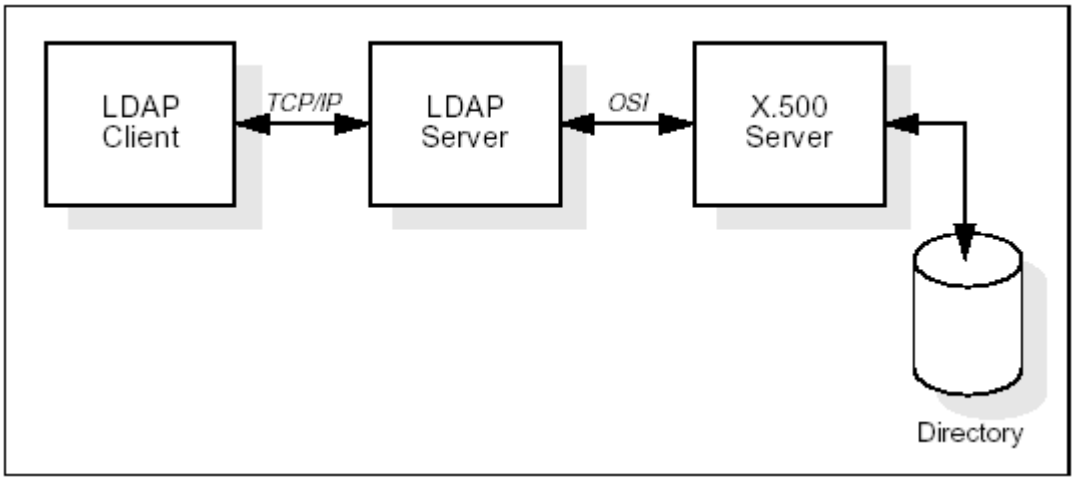
Για την σύγκριση μεγέθους επιλέχθηκε ο πελάτης ερευνών καταλόγου (Directory Enquiries Client) ο οποίος μπορεί να μεταγλωττισθεί ώστε να χρησιμοποιήσει είτε το DAP είτε το LDAP για την επικοινωνία X.500. Συγκρίθηκε επίσης η πολυπλοκότητα κώδικα του ISODE DAP και των βιβλιοθηκών των LDAP clients. Χρησιμοποιήθηκαν δύο μέτρα πολυπλοκότητας. Το πρώτο, ο συνολικός αριθμός του χαρακτήρα ‘’ ; ’’ που προσεγγίζει τον αριθμό των προτάσεων και το δεύτερο, ο συνολικός αριθμός των προτάσεων «if» που προσεγγίζει τον αριθμό των paths της κωδικοποίησης. Στον υπολογισμό των συγκεκριμένων μέτρων, καταβλήθηκε προσπάθεια να συμπεριληφθούν μόνο οι μερίδες του κώδικα που απαιτούνται για να υπάρχει πρόσβαση σον X.500.

1.4 LDAP: Πρωτόκολλο ή Κατάλογος;

Το LDAP ορίζει ένα πρωτόκολλο επικοινωνίας, δηλαδή, καθορίζει την μεταφορά και το format των μηνυμάτων που χρησιμοποιούνται από έναν πελάτη για να έχει πρόσβαση στα δεδομένα ενός καταλόγου της μορφής του X.500. Το LDAP δεν καθορίζει από μόνο του την υπηρεσία καταλόγου. Πολλοί άνθρωποι μιλούν για τους καταλόγους LDAP. Άλλοι λένε ότι το LDAP είναι απλώς ένα πρωτόκολλο και όχι ένας κατάλογος. Τι είναι όμως κατάλογος LDAP;

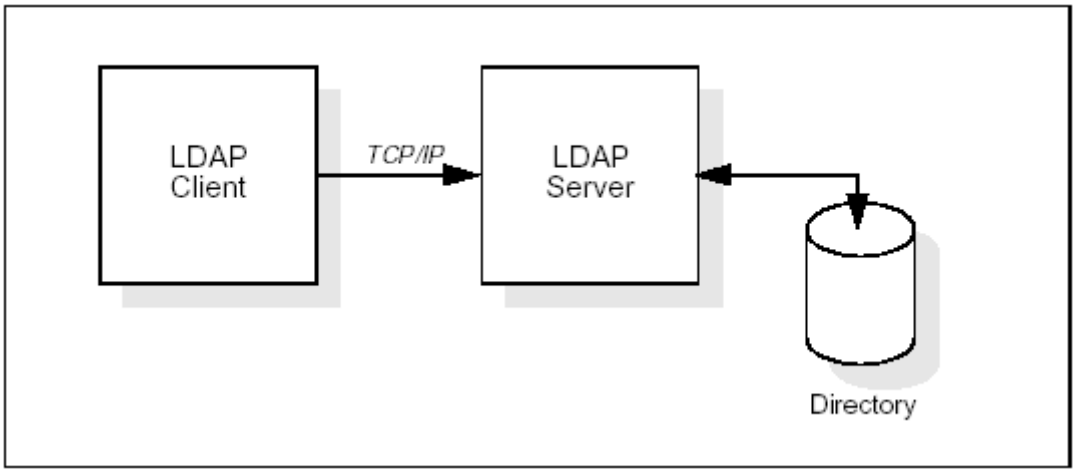
Ένα πρόγραμμα εφαρμογής πελάτη εισάγει ένα μήνυμα LDAP μέσω της κλήσης ενός LDAP API. Αλλά ένας εξυπηρετητής καταλόγου X.500 δεν καταλαβαίνει τα μηνύματα LDAP. Στην πραγματικότητα, ο πελάτης LDAP και ο εξυπηρετητής X.500 χρησιμοποιούν διαφορετικά πρωτόκολλα επικοινωνίας (TCP/IP εν αντιθέσει με το OSI). Ο πελάτης LDAP επικοινωνεί με μια διαδικασία πύλης (gateway) που διαβιβάζει αιτήματα στον εξυπηρετητή καταλόγου X.500. Η πύλη αυτή είναι γνωστή ως εξυπηρετητής LDAP (LDAP server). Ο LDAP server εξυπηρετεί τα αιτήματα του

LDAP client. Το επιτυγχάνει αυτό με το να γίνεται ο ίδιος πελάτης στον X.500 server. Ο LDAP server πρέπει να επικοινωνεί χρησιμοποιώντας και το TCP/IP και το OSI.



Σχήμα 1.4.α' Ο LDAP Server ως πύλη σε έναν X.500 Server

Καθώς μεγαλώνει η χρήση του LDAP και τα οφέλη του είναι προφανή, τα άτομα τα οποία δεν διέθεταν εξυπηρετητές X.500 ή που δεν είχαν τα κατάλληλα περιβάλλοντα για να τους υποστηρίξουν, θέλησαν να στήσουν καταλόγους που θα είχαν πρόσβαση σε αυτούς πελάτες LDAP. Έτσι λοιπόν γιατί να μην υπάρχει ο LDAP server αποθηκευμένος με δυνατότητα πρόσβασης στον κατάλογο από μόνος του (βλ. σχήμα 1.4.β') αντί να συμπεριφέρεται ως πύλη στους X.500 servers; Αυτό εξαλείφει οποιαδήποτε ανάγκη για το OSI. Βέβαια αυτό κάνει τον LDAP server πολύπλοκο αφού πρέπει να αποθηκεύει και να ανακτά τις καταχωρήσεις καταλόγου. Οι συγκεκριμένοι LDAP servers συχνά αποκαλούνται αυτόνομοι servers επειδή δεν εξαρτώνται από τον server καταλόγου X.500. Από την στιγμή που το LDAP δεν υποστηρίζει όλες τις δυνατότητες του X.500, ένας αυτόνομος LDAP server χρειάζεται μόνο να υποστηρίζει τις δυνατότητες που απαιτούνται από το LDAP.



Σχήμα 1.4.β' Αυτόνομος LDAP Server

Το RFC 1777 (LDAPv2) αναφέρει την παροχή πρόσβασης στον κατάλογο X.500. Το RFC 2251 (LDAPv3) συζητά την πρόσβαση σε καταλόγους που υποστηρίζουν το μοντέλο X.500. Αυτή η αλλαγή στη γλώσσα απεικονίζει την ιδέα ότι

ένας LDAP server μπορεί να εφαρμόσει ο ίδιος τον κατάλογο ή να αποτελέσει πύλη για έναν X.500 κατάλογο.

Από την πλευρά του πελάτη, οποιοσδήποτε server που εφαρμόζει το πρωτόκολλο LDAP είναι ένας LDAP server καταλόγου, εφόσον ο server εφαρμόζει τον κατάλογο ή είναι πύλη σε έναν server X.500. Ο κατάλογος ο οποίος προσεγγίζεται μπορεί να αποκαλείται LDAP κατάλογος, εφόσον ο κατάλογος εφαρμόζεται από έναν αυτόνομο LDAP server ή από έναν X.500 server.

1.5 Η Πορεία του LDAP

Το LDAP έχει εξελιχθεί για να ικανοποιήσει την ανάγκη παροχής πρόσβασης σε μια υποδομή ενός κοινού καταλόγου. Το LDAP είναι μια ανοιχτή βιομηχανία προτύπου που υποστηρίζεται από πολλούς προμηθευτές συστήματος σε πολλές πλατφόρμες. Ενσωματώνεται σε προϊόντα λογισμικού και γρήγορα γίνεται επιλογή ως πρωτόκολλο πρόσβασης καταλόγου. Επιτρέπει στα προϊόντα διαφορετικών προμηθευτών και πλατφόρμων να επικοινωνούν και να παρέχουν μια παγκόσμια υποδομή καταλόγου, όπως έγινε με το HTTP που συνέβαλε στην ανάπτυξη του Παγκόσμιου Ιστού. Τα πρόσφατα προϊόντα LDAP υποστηρίζουν τουλάχιστον την έκδοση 2. Ήδη, πολλά υποστηρίζουν μέρη της έκδοσης 3 ή και ολόκληρη.

Οι υπεύθυνοι ανάπτυξης εφαρμογών μπορούν να επωφεληθούν από το LDAP για να δημιουργήσουν εφαρμογές καταλόγου επόμενης γενιάς. Ενώ ο X.500 έχει παραδοσιακά αναπτυχθεί μόνο σε μεγάλους οργανισμούς όπου μπορεί να δεσμεύσει τους απαραίτητους πόρους για να τον υποστηρίξουν, το LDAP είναι επίσης κατάλληλο για μικρούς οργανισμούς. Για παράδειγμα, μια μικρή εταιρεία μπορεί να θέλει να ανταλλάξει έγγραφα με τους πελάτες και τους προμηθευτές της χρησιμοποιώντας ηλεκτρονική ανταλλαγή δεδομένων (Electronic Data Interchange). Η EDI απαιτεί και οι δυο πλευρές να συμφωνούν ως προς τον τύπο των εγγράφων που θα ανταλλαγούν, τις ιδιαιτερότητες της επικοινωνίας και ούτω καθεξής. Οι εταιρείες θα μπορούσαν να εκδώσουν τα χαρακτηριστικά της δικής τους EDI σε -δημόσιας πρόσβασης- LDAP καταλόγους για να διευκολύνουν την ηλεκτρονική ανταλλαγή. Μια κοινή υποδομή καταλόγου ενθαρρύνει νέες χρήσεις. Τα DEN (Directory Enabled Networks) παίρνουν την πρωτοβουλία και προτείνουν οι πληροφορίες για την κατασκευή δικτύων, για τα πρωτόκολλα, για τα χαρακτηριστικά των routers και ούτω καθεξής, να αποθηκεύονται σε κατάλογο LDAP. Η διαθεσιμότητα αυτής της πληροφορίας με ένα κοινό format από πολλούς προμηθευτές εξοπλισμού θα οδηγήσει στην έξυπνη διαχείριση και στην παροχή των δικτυακών πόρων. Τα παραδείγματα αυτά δείχνουν τις ποικίλες χρήσεις των εφαρμογών καταλόγου που υποστηρίζονται από μια κοινή υποδομή καταλόγου στην οποία πετυχαίνουμε πρόσβαση μέσω του LDAP.

1.6 Προϊόντα Εξυπηρετητών Καταλόγου (Directory Servers)

OpenLDAP 2.3

Ο OpenLDAP αποτελεί ένα προϊόν μιας συλλογικής προσπάθειας για την ανάπτυξη ενός ισχυρού και ανοιχτού κώδικα LDAP το οποίο είναι κατάλληλο για εφαρμογές και εργαλεία ανάπτυξης εφαρμογών. Το έργο αυτό διαχειρίζεται μια παγκόσμια κοινότητα εθελοντών που χρησιμοποιεί το Internet με σκοπό την επικοινωνία, τον σχεδιασμό και την ανάπτυξη του OpenLDAP Suite και της πρόσφατης τεκμηρίωσής του.

Το OpenLDAP Project πρόσφατα ανακοίνωσε την κυκλοφορία του OpenLDAP 2.3 (Ιούνιος 2005). Το λογισμικό OpenLDAP 2.3 αποτελεί μια ακολουθία του LDAP v3 που υποστηρίζει servers, χρησιμότητες και εργαλεία ανάπτυξης εφαρμογών.

Η έκδοση αυτή υποστηρίζει τις περισσότερες πλατφόρμες UNIX (και UNIX-like) όπως Darwin, FreeBSD, Linux, NetBSD καθώς και τα πιο εμπορικά UNIX συστήματα. Η έκδοση αυτή δοκιμάζεται επίσης (μερικώς ή εξ' ολοκλήρου) και σε άλλες πλατφόρμες όπως στις Apple MacOS X, IBM zOS, και Microsoft Windows 2000.

Ο OpenLDAP 2.3 περιέχει τις εξής σημαντικές επαυξήσεις:

-Slapd(8) enhancements

- Updated slapd "overlay" interface, and several example (and mostly experimental) overlays.
- Updated LDAP "sync" Engine with replication support, provider now an "overlay"
- Numerous access control enhancements, including experimental "don't disclose on error" capability
- Configuration Backend

- LDAPv3 extensions, including:

- LDAP Password Policy
- LDAP Component Matching (requires OpenLDAP-snacc)
- LDAP Modify Increment

Επίσης περιλαμβάνει τα εξής συστατικά:

- slapd - a stand-alone LDAP directory server
- slurpd - a stand-alone LDAP replication server
- -lldap - a LDAP client library
- -llber - a lightweight BER/DER encoding/decoding library
- LDIF tools - data conversion tools for use with slapd
- LDAP tools - A collection of command line LDAP utilities
- Admin Guide, Manual Pages - associated documentation
- SNACC - ASN.1 development tools for OpenLDAP

Επιπροσθέτως, υπάρχουν μερικά συμβαλλόμενα συστατικά:

- LDAPC++ - a LDAP C++ SDK
- Various slapd modules and slapi plugins

Sun ONE Directory Server 5.2

Ο Sun ONE Directory Server 5.2 είναι ένας ισχυρός και με δυνατότητες εξέλιξης κατανεμημένος server ο οποίος βασίζεται στο LDAP. Το λογισμικό του είναι μέρος του Sun Open Net Environment (Sun ONE) το οποίο βασίζεται σε ανοιχτά πρότυπα όπως Java και παρέχει ένα σημαντικά ευέλικτο περιβάλλον για την δημιουργία ενός πλήθους υπηρεσιών κατ' απαίτηση (services on demand). Ο συγκεκριμένος server προσφέρει πληροφορίες για έναν μεγάλο αριθμό εφαρμογών υποστηρίζοντας δυο πρωτόκολλα για την πρόσβαση στον παγκόσμιο κατάλογο: το LDAP και το DSML (Directory Services Markup Language).

Ο Sun ONE Directory Server 5.2 υποστηρίζει τις ακόλουθες πλατφόρμες:

Πλατφόρμες	Αρχιτεκτονική
Sun Solaris™ 9 Operating Environment	SPARC® processors in 32-bit and 64-bit mode Supported x86 processors
Sun Solaris 8 Operating Environment	UltraSPARC® processors in 32-bit and 64-bit mode
Sun Linux 5.0	Sun LX50 servers
Hewlett Packard HP-UX 11i	PA-RISC 2.0 processors in 32-bit and 64-bit mode
IBM AIX 5.1	PowerPC processors
Microsoft Windows 2000 Server, SP 3 Microsoft Windows 2000 Advanced Server, SP 3	Pentium II or later IA-32 processors
Red Hat Linux 7.2	Pentium II or later IA-32 processors

Ο Sun ONE Directory Server 5.2 υποστηρίζει πολλά πρότυπα εκ των οποίων ορισμένα παρουσιάζονται στον παρακάτω πίνακα:

Πρότυπο	Περιγραφή
DSML v2	Directory Services Markup Language, an open standard for web services using Simple Object Access Protocol (SOAP) and Hypertext Transfer Protocol (HTTP) to provide access to directory information
LDAP search filters	The String Representation of LDAP Search Filters, RFC 2254, for searching on attribute value presence, equality, inequality, substring, approximate, greater than, less than, and Boolean combinations thereof
LDAP URLs	An LDAP URL Format, RFC 2255
LDAP v3	Lightweight Directory Access Protocol (v3), RFC 2251, the Internet standard protocol for accessing directory information
LDIF	LDAP Data Interchange Format, RFC 2849, for exchanging directory information
SASL	Simple Authentication and Security Layer, RFC 2222, providing for extensible authentication

SNMP v2	Simple Network Management Protocol for network management and monitoring
SSL/TLS	Secure Sockets Layer and Transport Layer Security, industry standards for securing data on the network
X.509 v3	Standard for public key certificate formats enabling secure authentication

IBM Tivoli Directory Server

Ο IBM Tivoli Directory Server παρέχει μια ισχυρή υποδομή ταυτότητας LDAP όπου αποτελεί το ίδρυμα για την ανάπτυξη περιεκτικών εφαρμογών διαχείρισης ταυτότητας και προηγμένων αρχιτεκτονικών λογισμικού όπως οι υπηρεσίες Ιστού.

Οι πλατφόρμες που υποστηρίζει ο IBM Tivoli Directory Server είναι: AIX, Solaris, Microsoft, IBM eServer iSeries, pSeries και zSeries.

Χαρακτηριστικά γνωρίσματα	**Πλεονεκτήματα**	**Οφέλη**
Broad platform support	Runs on AIX, Solaris, Microsoft Windows 2000, and HP-UX, as well as Linux distributions for Intel and IBM eServer iSeries, pSeries and zSeries platforms	Provides great flexibility to fit into the enterprise. Consistent GUI-driven cross-platform install experience
DB2 UDB 8.1 Data store	Robust e-business foundation	Proven performance, reliability and data integrity
Robust Replication	Single-master multiple replication and multiple-master replication are supported as is cascaded, gateway and partial replication	Allows configuration of servers to match the topography of the network to ensure data availability and to maximize server response time
DSML V2 support	Extends the reach of the directory to web services. Expose the directory and deliver it to web services through XML coding.	An enterprise's customers could, for example, make changes to directory data such as phone numbers or street addresses themselves over the Internet rather than calling in to customer service
Password strength features	Enable the pre-expiration of passwords, the definition of password rules, maintenance of password history and failed attempt account	Leverage these password features along with ACL protection features to enhance directory security

Comprehensive and extensible schema	Dynamically updatable schema includes person objects and attributes and the Dynamic Enabled Networking (DEN)schema for devices and support for Java object storage	Using the IBM schema helps avoid the time-consuming design of schemata, which can slow the deployment of the LDAP directory
Management	The Web browser-based directory administration console can be run on the server or remotely. IBM Directory Server configuration settings may also be viewed or updated via LDAP.Audit, Change and Error logs are provided to allow quick problem detection and prevention.	Offers great flexibility in managing the operation of IBM Directory Server

Novell eDirectory 8.7.3

Ο Novell eDirectory είναι μια υψηλής απόδοσης και ασφαλής υπηρεσία καταλόγου. Μπορεί να αποθηκεύσει και να διαχειρισθεί εκατομμύρια αντικείμενα, όπως χρήστες, εφαρμογές, συσκευές δικτύου και δεδομένα. Παρέχει συγκεντρωτική διαχείριση ταυτότητας, υποδομή, ασφάλεια διαδικτύου και δυνατότητες εξέλιξης σε όλους τους τύπους εφαρμογών που τρέχουν πίσω και πέρα από το firewall. Ο Novell eDirectory 8.7.3 παρέχει διαδικτυακές και ασύρματες δυνατότητες διαχείρισης, επιτρέποντας την πρόσβαση και διαχείριση του καταλόγου, των χρηστών, των δικαιωμάτων πρόσβασης και των δικτυακών πόρων.

Ο Novell eDirectory υποστηρίζει το πρότυπο κατάλογου LDAPv3 και παρέχει υποστήριξη για υπηρεσίες TLS/SSL που βασίζονται στον πηγαίο κώδικα OpenSSL. Τρέχει σε Linux, NetWare, Windows, Solaris, AIX και HP-UX.

Οι απαιτήσεις συστήματος για τον Novell eDirectory 8.7.3 είναι:

NetWare

- NetWare 5.1 Support Pack 6 or later
- NetWare 6 Support Pack 3 or later
- NetWare 6.5 Support Pack 1 (eDirectory 8.7.3 is only supported through the NetWare 6.5 Support Pack 1 installation)

If you are using RCONSOLE, you will need a ConsoleOne® administrator workstation with the following:

- 200 MHZ or faster processor
- 64 MB RAM (128 recommended)
- Novell Client™ for Windows NT/2000/XP version 4.9 or later or Novell Client for Windows 95/98 version 3.4 or later

Windows

One of the following:

- Windows NT Server 4.0 with Service Pack 6 or later
- Windows 2000 Server with Service Pack 4 or later
- Windows Server 2003

Important**:** Windows XP is not a supported Novell eDirectory 8.7.3 platform.

- An assigned IP address
- A Pentium 200 with a minimum of 64 MB RAM (128 MB recommended) and a monitor color palette set to a number higher than 16

(Optional) One or more workstations running one of the following:

- Novell Client for Windows 95/98 version 3.4 or later
- Novell Client for Windows NT/2000/XP version 4.9 or later
- Administrative rights to the NT/2000 server and to all portions of the eDirectory tree that contain domain-enabled User objects. For an installation into an existing tree, you need administrative rights to the root of the tree so that you can extend the schema and create objects.

Linux

One of the following:

- SUSE LINUX Enterprise Server 9 (IR3 required)
- SUSE LINUX Enterprise Server 8
- Red Hat Linux 7.3, 8.0, 9.0, or Red Hat Advanced Server 2.1
  Note: Ensure that the latest glibc patches are applied from Red Hat Errata on Red Hat systems.
- 128 MB RAM minimum
- 90 MB of disk space for the eDirectory server
- 25 MB of disk space for the eDirectory administration utilities
- 74 MB of disk space for every 50,000 users

Ensure that gettext is installed

Solaris

One of the following:

- Solaris 8 on Sun SPARC (with patch 108827-20 or later)
- Solaris 9 on Sun SPARC
- All of the latest recommended sets of patches are available on the SunSolve webpage. If you do not update your system with the latest patches before installing eDirectory, you will get the patchadd error.
- 128 MB RAM minimum
- 120 MB of disk space for the eDirectory server
- 32 MB of disk space for the eDirectory administration utilities
- 74 MB of disk space for every 50,000 users

ΚΕΦΑΛΑΙΟ 2: ΑΡΧΙΤΕΚΤΟΝΙΚΗ LDAP

Το LDAP είναι βασισμένο στο μοντέλο client/server και έχει εξελιχθεί ως ένα ελαφρύ πρωτόκολλο για την πρόσβαση στις πληροφορίες των υπηρεσιών καταλόγου X.500. Από τότε έχει γίνει πιο ανεξάρτητο από το X.500 και οι servers που υποστηρίζουν αποκλειστικά το LDAP και όχι το DAP, είναι πλέον κοινοί. Η επιτυχία του LDAP έχει αυξηθεί και αυτό οφείλεται στα ακόλουθα χαρακτηριστικά που το καθιστούν απλούστερο στην εφαρμογή σε σχέση με το X.500 και το DAP:

- Το LDAP τρέχει στο TCP/IP και όχι στη στοίβα του πρωτοκόλλου OSI. Το TCP/IP είναι λιγότερο απαιτητικό σε πόρους και είναι ευρέως διαθέσιμο, ειδικά στα υπολογιστικά συστήματα γραφείου.
- Το λειτουργικό μοντέλο του LDAP είναι πιο απλό. Παραλείπει τα αντίγραφα και τα σπανίως χρησιμοποιούμενα εσωτερικά χαρακτηριστικά γνωρίσματα. Όλα αυτά καθιστούν το LDAP ευκολότερο στην κατανοησή και στην εφαρμογή του.
- Το LDAP χρησιμοποιεί στοιχειοσειρές (strings) για να αναπαραστήσει τα δεδομένα και όχι πολύπλοκα δομημένα συντακτικά όπως η ASN.1 (Abstract Syntax Notation One).

2.1 Επισκόπηση Αρχιτεκτονικής LDAP

Το LDAP καθορίζει το περιεχόμενο των μηνυμάτων που ανταλλάσσονται μεταξύ του LDAP client και του LDAP server. Τα μηνύματα διευκρινίζουν τις λειτουργίες που ζητούνται από τον πελάτη (αναζήτηση, τροποποίηση, διαγραφή κ.λ.π.), τις απαντήσεις από την μεριά του server, και το format των δεδομένων που μεταφέρονται στα μηνύματα. Τα μηνύματα LDAP μεταφέρονται πάνω στο TCP/IP. Επειδή το TCP/IP είναι πρωτόκολλο που προσανατολίζεται με σύνδεση, υπάρχουν λειτουργίες που γίνονται για να καθιερώσουν και να διακόψουν μια σύνοδο μεταξύ του πελάτη και του εξυπηρετητή.

Ωστόσο, για τον σχεδιαστή του καταλόγου LDAP, δεν έχει τόσο σημασία η δομή των μηνυμάτων που στέλνονται και λαμβάνονται. Αυτό που έχει σημασία είναι το λογικό μοντέλο το οποίο καθορίζεται από αυτά τα μηνύματα και από τους τύπους των δεδομένων π.χ. πώς οργανώνεται ο κατάλογος, ποιες λειτουργίες μπορούν να γίνουν, με ποιο τρόπο προστατεύονται οι πληροφορίες και ούτω καθεξής.

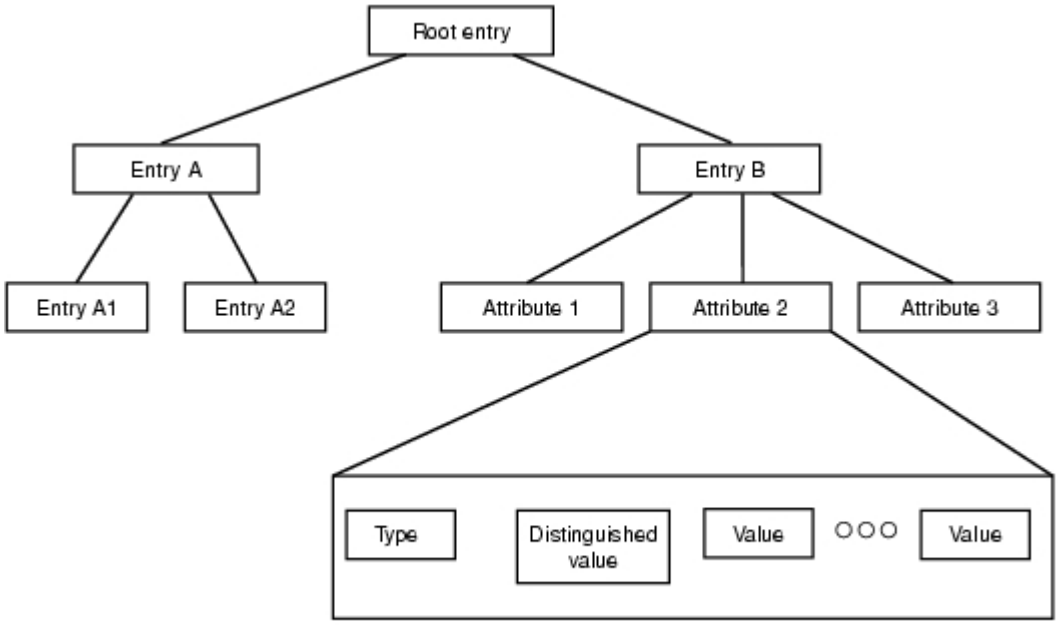
Η γενική αλληλεπίδραση μεταξύ των LDAP server και client παίρνει την ακόλουθη μορφή:

- Ο client καθιερώνει μια σύνοδο με έναν LDAP server που είναι γνωστή ως *δέσμευση* του server. Ο client διευκρινίζει το host name ή την IP διεύθυνση και τον port number του TCP/IP όπου σε αυτόν ‘’ακούει’’ ο LDAP server. Ο client μπορεί να παρέχει ένα user name και ένα κωδικό πρόσβασης για να επικυρώνεται κατάλληλα με τον server. Ή μπορεί να καθιερώσει μια ανώνυμη σύνοδο με προεπιλεγμένα δικαιώματα πρόσβασης (by default). Επίσης ο client και ο server μπορούν να καθιερώσουν μια σύνοδο που να εφαρμόζει ισχυρές μεθόδους κρυπτογράφησης δεδομένων.

- Έπειτα ο client εκτελεί λειτουργίες πάνω στα δεδομένα του καταλόγου. Αυτό επιτρέπει στις πληροφορίες καταλόγου να διαχειρίζονται και να ερωτώνται. Επίσης το LDAP υποστηρίζει την αναζήτηση δεδομένων που ικανοποιούν αυθαίρετα κριτήρια τα οποία έχουν καθοριστεί από τον χρήστη. Η διαδικασία αναζήτησης είναι πολύ συνηθισμένη για τον LDAP. Ένας χρήστης μπορεί να διευκρινίσει ποιο μέρος του καταλόγου θέλει να αναζητηθεί και ποια πληροφορία να επιστραφεί. Ένα φίλτρο αναζήτησης που χρησιμοποιεί όρους BOOLEAN διευκρινίζει ποιο στοιχείο του καταλόγου ταιριάζει με το κριτήριο αναζήτησης.
- Όταν ο client ολοκληρώσει την υποβολή ερωτημάτων κλείνει την σύνοδο με τον server. Αυτό είναι γνωστό ως *αποδέσμευση* (unbinding).

Αν και δεν καθορίζεται από το πρωτόκολλο LDAP ούτε από την αρχιτεκτονική του, υπάρχει ένα γνωστό LDAP API που επιτρέπει στις εφαρμογές να αλληλεπιδρούν εύκολα με τους LDAP servers. Το API μπορεί να θεωρηθεί ως μια προέκταση της αρχιτεκτονικής του LDAP. Παρόλο που το LDAP API της γλώσσας C είναι μόνο ένα ενημερωτικό RFC και η πιο πρόσφατη ενημερωσή του είναι ένα προσχέδιο Internet, έχει καταφέρει de facto να φθάσει σε κατάσταση προτύπου επειδή υποστηρίζεται από όλους τους σημαντικούς προμηθευτές LDAP. Η φιλοσοφία του LDAP API είναι να διατηρεί τα πράγματα απλά. Αυτό σημαίνει ότι μπορεί, με χαμηλό γενικά κόστος, να προστεθεί υποστήριξη καταλόγου στις υπάρχουσες εφαρμογές.

Επειδή το LDAP προορίστηκε αρχικά ως μια ελαφριά εναλλακτική λύση του DAP για την πρόσβαση σε καταλόγους X.500, ακολουθεί το μοντέλο X.500. Ο κατάλογος αποθηκεύει και οργανώνει δομές δεδομένων γνωστές ως ***καταχωρήσεις*** (*entries*). Μια καταχώρηση καταλόγου συνήθως περιγράφει ένα **αντικείμενο** όπως ένα άτομο, έναν εκτυπωτή, έναν server και τα λοιπά. Κάθε καταχώρηση έχει ένα όνομα γνωστό ως ***διακριτό όνομα*** (*Distinguished Name(DN))* που την προσδιορίζει μοναδικά. Το DN αποτελείται από μια ακολουθία μερών που ονομάζονται *σχετικά διακριτά ονόματα* (*RDNs*) τα οποία μοιάζουν κατά πολύ με το όνομα αρχείου που περιέχει τα paths ονομάτων καταλόγου σε πολλά λειτουργικά συστήματα όπως των UNIX και WINDOWS. Οι καταχωρήσεις μπορούν να ταξινομούνται σε μια ιεραρχική μορφή δέντρου βασισμένη στα διακριτά ονοματά τους. Το δέντρο των καταχωρήσεων καταλόγου ονομάζεται Δέντρο Πληροφοριών Καταλόγου (Directory Information Tree (**DIT**)).



Σχήμα 2.1 Δομή DIT

Κάθε καταχώρηση περιέχει ένα ή περισσότερα ***χαρακτηριστικά*** (*attributes*) τα οποία περιγράφουν την καταχώρηση. Κάθε χαρακτηριστικό έχει έναν τύπο και μια τιμή. Για παράδειγμα, η καταχώρηση καταλόγου ενός ατόμου μπορεί να έχει το χαρακτηριστικό telephoneNumber. Το συντακτικό του χαρακτηριστικού telephoneNumber διευκρινίζει ότι ο τηλεφωνικός αριθμός πρέπει να είναι μια ακολουθία από αριθμούς που μπορεί να περιέχουν διαστήματα και παύλες. Η τιμή του χαρακτηριστικού θα είναι ο τηλεφωνικός αριθμός του ατόμου π.χ 512-555-1212.

Μια καταχώρηση καταλόγου περιγράφει κάποιο **αντικείμενο** (*object*). Μια *κατηγορία αντικειμένου* είναι μια γενική περιγραφή που συχνά αναφέρεται ως *φόρμα* αντικειμένου σε αντιδιαστολή με την περιγραφή ενός συγκεκριμένου αντικειμένου. Παραδείγματος χάριν, η κατηγορία (**class**) αντικειμένου *person* έχει ένα χαρακτηριστικό surname ενώ το αντικείμενο που περιγράφει το John Smith έχει ένα χαρακτηριστικό surname με την τιμή Smith.

Οι κατηγορίες αντικειμένου που μπορεί να αποθηκεύσει ένας server καταλόγου καθώς και τα χαρακτηριστικά που περιέχονται σ' αυτές, περιγράφονται στο **σχήμα** (*schema*). Το σχήμα καθορίζει το είδος των κατηγοριών αντικειμένου που επιτρέπεται να αποθηκευθούν στον κατάλογο, ποια χαρακτηριστικά πρέπει να έχουν, ποια από αυτά είναι προαιρετικά και τον συντακτικό τύπο αυτών. Για παράδειγμα, θεωρούμε ένα σχήμα που ορίζει μια κατηγορία αντικειμένου με το όνομα person. Το σχήμα της κατηγορίας person απαιτεί ότι ένα άτομο θα έχει το χαρακτηριστικό surname το οποίο θα είναι ένα string χαρακτήρων, διευκρινίζοντας ότι η καταχώρηση ενός ατόμου μπορεί προαιρετικά να έχει και το χαρακτηριστικό telephoneNumber όπου είναι ένα string αριθμών, κενών και παυλών.

Το LDAP ορίζει λειτουργίες για την προσπέλαση και τροποποίηση καταχωρήσεων καταλόγου όπως:

- Αναζήτηση για καταχωρήσεις που ικανοποιούν τα κριτήρια που καθορίζονται από τον χρήστη
- Προσθήκη καταχώρησης
- Διαγραφή καταχώρησης
- Τροποποίηση καταχώρησης

- Τροποποίηση του διακριτού ή του σχετικού διακριτού ονόματος μιας καταχώρησης
- Σύγκριση καταχώρησης

Το LDAP είναι τεκμηριωμένο σε πολλά IETF RFCs. Παρατίθενται τα RFCs της έκδοσης 3 με μια μικρή περιγραφή για να γίνει μια επισκόπηση των εγγράφων που καθορίζουν την αρχιτεκτονική του LDAP.

1. RFC 2251 Lightweight Directory Access Protocol (v3)

Περιγράφει το πρωτόκολλο LDAP που είναι σχεδιασμένο ώστε να παρέχει ελαφριά πρόσβαση σε καταλόγους που υποστηρίζουν το μοντέλο X.500. Το ελαφρύ πρωτόκολλο προορίστηκε για εφαρμογή σε περιβάλλοντα περιορισμένων πόρων όπως οι browsers και τα μικρά υπολογιστικά συστήματα γραφείου. Αυτό το RFC αποτελεί τον πυρήνα της οικογένειας των LDAP RFCs. Περιγράφει το πώς ονομάζονται οι καταχωρήσεις με τα διακριτά ονόματα, καθορίζει το format των μηνυμάτων που ανταλλάσσονται μεταξύ του πελάτη και του server, απαριθμεί τις λειτουργίες που μπορούν να εκτελεσθούν από την μεριά του πελάτη και διευκρινίζει ότι τα δεδομένα αναπαριστώνται με την χρησιμοποίηση της κωδικοποίησης χαρακτήρα UTF-8. Το RFC 2251 διευκρινίζει ότι οι καταχωρήσεις καταλόγου που περιγράφονται στο σχήμα πρέπει να είναι οι ίδιες αναγνώσιμες έτσι ώστε ένας πελάτης να μπορεί να προσδιορίσει τον τύπο των αντικειμένων που αποθηκεύει ο κατάλογος. Καθορίζει τον τρόπο με τον οποίο ένας πελάτης μπορεί να προσφύγει σε έναν άλλον LDAP server εάν ένας server δεν περιέχει την ζητούμενη πληροφορία. Περιγράφει το πώς οι μεμονωμένες λειτουργίες με τις κατάλληλες ρυθμίσεις μπορούν να επεκταθούν και πώς οι πρόσθετες λειτουργίες μπορούν να προσδιοριστούν χρησιμοποιώντας επεκτάσεις. Επίσης συζητάει για το πώς οι πελάτες μπορούν να ζητήσουν επικύρωση από τους servers και προαιρετικά να χρησιμοποιήσουν το Simple Authentication and Security Layer (SASL) για να επιτρέψουν πρόσθετους μηχανισμούς επικύρωσης.

2. RFC 2252 Lightweight Directory Access Protocol (v3): *Attribute Syntax Definitions*

Το LDAP χρησιμοποιεί οκταδικά strings για να αναπαραστήσει τις τιμές των χαρακτηριστικών για τη μεταφορά στο πρωτόκολλο LDAP. Αυτό το RFC καθορίζει το πώς αναπαριστώνται τιμές όπως οι ακέραιοι αριθμοί, διευθύνσεις e-mail και τα λοιπά. Για παράδειγμα, ο ακέραιος 123 αναπαριστάται από το string ‘’123’’. Οι ορισμοί αυτοί ονομάζονται συντακτικά χαρακτηριστικών (attribute syntaxes). Το συγκεκριμένο RFC περιγράφει το πώς κωδικοποιείται ένα χαρακτηριστικό με ένα συγκεκριμένο συντακτικό τύπο όπως ‘’τηλεφωνικός αριθμός’’. Επίσης καθορίζει τους κανόνες που χρειάζονται για να καθορισθεί αν οι τιμές ικανοποιούν τα κριτήρια αναζήτησης.

Αυτοί οι τύποι χαρακτηριστικών και ο συντακτική τους μορφή χρησιμοποιούνται για να χτίσουν το σχήμα που περιγράφει τις κατηγορίες αντικειμένων. Ένα σχήμα αναφέρει τα χαρακτηριστικά που πρέπει να έχει μια καταχώρηση καταλόγου. Κάθε καταχώρηση καταλόγου έχει ένα objectclass χαρακτηριστικό το οποίο απαριθμεί το σχήμα (ένα ή περισσότερα) που περιγράφει την καταχώρηση. Παραδείγματος χάριν, μια καταχώρηση καταλόγου μπορεί να περιγραφθεί από τις κατηγορίες αντικειμένου residentialPerson και organizationalPerson. Αν ένα χαρακτηριστικό objectclass περιλαμβάνει την τιμή extensibleObject, τότε μπορεί να περιέχει οποιοδήποτε χαρακτηριστικό.

3. RFC 2253 Lightweight Directory Access Protocol (v3): *UTF-8 String Representation of Distinguished Names*

Τα διακριτά ονόματα (DNs) είναι τα μοναδικά αναγνωριστικά των καταχωρήσεων καταλόγου τα οποία μερικές φορές καλούνται πρωτεύοντα κλειδιά (primary keys). Το X.500 χρησιμοποιεί την ASN.1 για να κωδικοποιήσει τα διακριτά ονόματα. Το LDAP κωδικοποιεί τα διακριτά ονόματα ως strings. Αυτό το RFC ορίζει τον τρόπο με τον οποίο αναπαριστώνται τα DNs με την μορφή strings. Μια αναπαράσταση string είναι εύκολο να κωδικοποιηθεί και να αποκωδικοποιηθεί και επιπλέον είναι ανθρωπίνως αναγνώσιμη. Ένα DN αποτελείται από μια ακολουθία σχετικών διακριτών ονομάτων (RDNs) που χωρίζονται με κόμματα. Η ακολουθία των RDNs κάνουν τα DN ονόματα να είναι οι πρόγονοι μιας καταχώρησης καταλόγου στην ρίζα ενός DIT. Κάθε RDN έχει συγκροτηθεί από την τιμή ενός χαρακτηριστικού που ανήκει σε μια καταχώρηση. Για παράδειγμα, το DN cn=John Smith, ou=Austin, o=IBM, c=US αντιπροσωπεύει μια καταχώρηση που αφορά ένα άτομο με το όνομα (cn: common name) John Smith που ανήκει στην οργανωτική μονάδα (ou: organizational unit) Austin στον οργανισμό (o: organization) IBM στην χώρα (c: country) US.

4. RFC 2254 *The String Representation of LDAP Search Filters*

Τα φίλτρα αναζήτησης του LDAP παρέχουν έναν ισχυρό μηχανισμό αναζήτησης καταχωρήσεων καταλόγου που ικανοποιούν ορισμένα κριτήρια. Το πρωτόκολλο LDAP ορίζει την δικτυακή αναπαράσταση ενός φίλτρου αναζήτησης. Μια τέτοια αναπαράσταση μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές ή στον πηγαίο κώδικα ενός προγράμματος για να ορίσει τα κριτήρια αναζήτησης. Οι τιμές των χαρακτηριστικών συγκρίνονται με την χρησιμοποίηση τελεστών όπως ‘’=’’, ‘’>’’. Οι Boolean τελεστές μπορούν να χρησιμεύσουν στην κατασκευή πιο σύνθετων φίλτρων. Παραδείγματος χάριν, το φίλτρο αναζήτησης (| (sn=Smith) (cn=Jo*)) ψάχνει για εγγραφές που είτε έχουν ένα χαρακτηριστικό surname Smith είτε ένα common name που αρχίζει από Jo.

5. RFC 2255 *The LDAP URL Format*

Οι URLs (Uniform Resource Locators) χρησιμοποιούνται για να αναγνωρίζουν ιστοσελίδες, αρχεία και άλλες πηγές στο Internet. Ένα LDAP URL διευκρινίζει ότι μια αναζήτηση LDAP θα εκτελεσθεί σε συγκεκριμένο LDAP server. Ένα LDAP URL αντιπροσωπεύει με σταθερό και τυποποιημένο τρόπο την επιστρεφόμενη πληροφορία ως το αποτέλεσμα της αναζήτησης.

6. RFC 2256 *A Summary of the X.500 (96) User Schema for use with LDAPv3*

Πολλά σχήματα και χαρακτηριστικά που συνήθως προσεγγίζονται από τους πελάτες καταλόγου είναι ήδη καθορισμένα από το X.500. Αυτό το RFC κάνει μια επισκόπηση των χαρακτηριστικών και των κατηγοριών αντικειμένου που πρέπει οι LDAP servers να αναγνωρίζουν. Για παράδειγμα, καθορίζονται χαρακτηριστικά όπως cn (common name), description και postalAddress. Επίσης καθορίζονται κατηγορίες αντικειμένου όπως country, organizationalUnit, groupOfNames και applicationEntity.

Τα RFCs που αναφέρθηκαν παραπάνω διαμορφώνουν τον πυρήνα του LDAP προδιαγραφής έκδοσης 3. Εκτός από αυτά τα RFCs, η IETF διαθέτει έναν αριθμό προτεινόμενων επεκτάσεων του LDAPv3 που μπορούν να εφαρμοσθούν από τους προμηθευτές. Εντούτοις, οι επεκτάσεις αυτές βρίσκονται ακόμα σε κατάσταση προσχεδίων αλλά μπορεί να αλλάξουν. Η ακόλουθη λίστα συνοψίζει κάποιες από τις προτεινόμενες επεκτάσεις:

- Υποχρεωτική εφαρμογή επικύρωσης
  Μια προσπάθεια ώστε να υπάρξει μια τουλάχιστον πρότυπη, ασφαλής μέθοδο επικύρωσης σε όλους τους servers και clients (όχι μόνο στο LDAP), παρά ξεχωριστές μέθοδοι για κάθε πρωτόκολλο που τρέχει πάνω στο TCP/IP.
- Επεκτάσεις για Δυναμικές Υπηρεσίες Καταλόγου
  Αυτή είναι μια επέκταση πρωτοκόλλου που επιτρέπει στους πελάτες να αλληλεπιδρούν πιο αξιόπιστα με τους servers όταν τα περιεχόμενα του καταλόγου αλλάζουν.
- Χρήση των Κωδικών Γλώσσας στο LDAP
  Περιγράφει την προσθήκη των κωδικών φυσικής γλώσσας στα χαρακτηριστικά που αποθηκεύονται στον κατάλογο LDAP.
- Επέκταση του LDAPv3 για την ασφάλεια του στρώματος μεταφοράς
  Καθορίζει την ενσωμάτωση του μηχανισμού ασφάλειας στρώματος μεταφοράς (TLS) μέσα στο LDAP.
- Επέκταση ελέγχου LDAP για τον χειρισμό σελιδοποιημένων αποτελεσμάτων
  Περιγράφει μια επέκταση ελέγχου για σελιδοποίηση των αποτελεσμάτων αναζήτησης. Αυτό αποτελεί ειδική αξία για τους απλούς, -περιορισμένης λειτουργίας- πελάτες έτσι ώστε να μπορούν να ζητούν τα αποτελέσματα αναζήτησης να επιστρέφονται σε μικρότερα τμήματα (σελίδες) κάθε φορά.
- Παραπομπές και αναφορές πληροφορίας σε καταλόγους LDAP
  Καθορίζει το πώς οι παραπομπές και οι αναφορές πληροφορίας μπορούν να αποθηκευθούν ως χαρακτηριστικά και πώς μπορούν να χρησιμοποιηθούν.
- Επέκταση ελέγχου LDAP για την ταξινόμηση των αποτελεσμάτων αναζήτησης στον server.
  Επιτρέπει την ταξινόμηση των αποτελεσμάτων αναζήτησης στον server απ' ότι στον client. Αυτό μπορεί να είναι επιθυμητό για την κατασκευή απλούστερων -περιορισμένης λειτουργίας- πελατών.
- Η διεπαφή προγραμματισμού του LDAP
  Καθορίζει το API της γλώσσας C στο LDAP. Οι περισσότεροι προμηθευτές ήδη ενσωματώνουν αυτή την επέκταση ή τουλάχιστον ένα μέρος αυτής.

2.2 Τα Πρότυπα LDAP

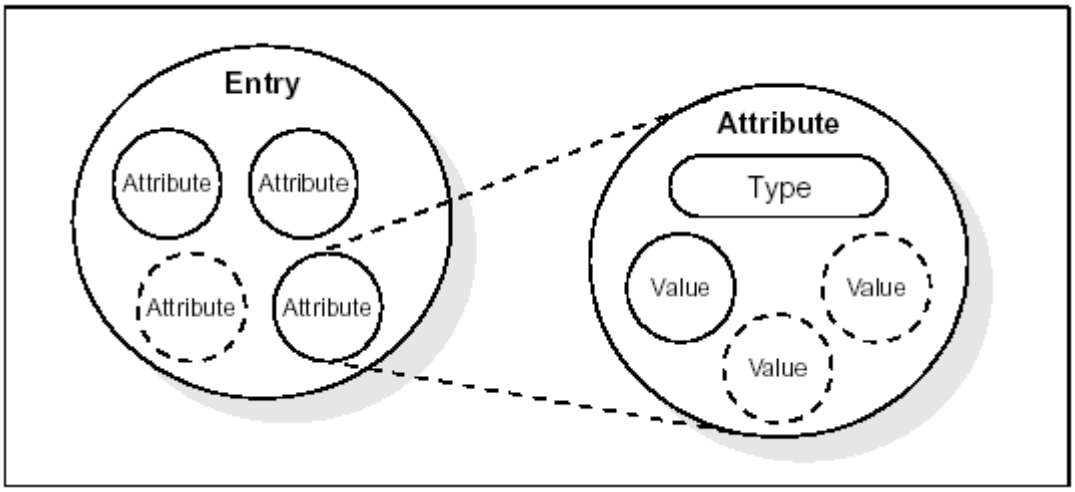
Το LDAP μπορεί να γίνει καλύτερα κατανοητό αν εξετάσουμε τα τέσσερα μοντέλα στα οποία είναι βασισμένο:

- **Πληροφορία**
  Περιγράφει την δομή των πληροφοριών που αποθηκεύονται σε έναν κατάλογο LDAP.

- **Ονομασία**
  Περιγράφει τον τρόπο με τον οποίο η πληροφορία οργανώνεται και αναγνωρίζεται στον κατάλογο LDAP.
- **Λειτουργικότητα**
  Περιγράφει τις λειτουργίες που μπορούν να εκτελεσθούν πάνω στις αποθηκευμένες πληροφορίες καταλόγου LDAP.
- **Ασφάλεια**
  Περιγράφει το πώς η αποθηκευμένη πληροφορία καταλόγου LDAP μπορεί να προστατεύεται από μη επικυρωμένες προσβάσεις.

2.2.1 Το Πληροφοριακό Μοντέλο

Η βασική μονάδα αποθηκευμένων πληροφοριών καταλόγου ονομάζεται *καταχώρηση*. Οι καταχωρήσεις αναπαριστούν αντικείμενα όπως άτομα, servers, οργανισμούς και ούτω καθεξής. Αποτελούνται από μια συλλογή χαρακτηριστικών που περιέχουν πληροφορίες για τα αντικείμενα. Κάθε χαρακτηριστικό έχει έναν τύπο και μια ή περισσότερες τιμές. Ο τύπος του χαρακτηριστικού συνδέεται με έναν συντακτικό τύπο. Ο συντακτικός τύπος διευκρινίζει τα είδη των τιμών που μπορούν να αποθηκευθούν. Για παράδειγμα, μια καταχώρηση μπορεί να έχει το χαρακτηριστικό facsimileTelephoneNumber. Ο συντακτικός τύπος μαζί με αυτόν τον τύπο του χαρακτηριστικού διευκρινίζει ότι οι τιμές είναι τηλεφωνικοί αριθμοί οι οποίοι αναπαρίστανται από εκτυπώσιμα strings. Είναι πιθανό μια καταχώρηση καταλόγου ενός οργανισμού να περιέχει διαφορετικές τιμές για το παραπάνω χαρακτηριστικό. Αυτό σημαίνει ότι ένας οργανισμός ή άτομο που αντιπροσωπεύεται από την οντότητα θα έχει πολλούς αριθμούς fax. Η σχέση μεταξύ καταχώρησης καταλόγου, των χαρακτηριστικών και των τιμών της φαίνεται στο σχήμα 2.2.1.



Σχήμα 2.2.1 Καταχωρήσεις, χαρακτηριστικά και τιμές

Εκτός από τον καθορισμό των δεδομένων που αποθηκεύονται ως τιμή σε ένα χαρακτηριστικό, ένας συντακτικός τύπος καθορίζει επίσης το πώς αυτές οι τιμές συμπεριφέρονται κατά την διάρκεια των διαδικασιών αναζήτησης και άλλων

λειτουργιών καταλόγου. Παραδείγματος χάριν, το χαρακτηριστικό telephoneNumber έχει έναν συντακτικό τύπο ο οποίος καθορίζει:

- την λεξικογραφική σειρά
- ότι τα κενά και οι παύλες αγνοούνται κατά την διάρκεια των συγκρίσεων
- ότι οι τιμές πρέπει να είναι strings χαρακτήρων

Για παράδειγμα, χρησιμοποιώντας τους σωστούς ορισμούς, οι τηλεφωνικοί αριθμοί ‘’512-838-6008’’, ‘’512838-6008’’ και ‘’5128386008’’ θεωρούνται ότι είναι ίδιοι. Μερικοί από τους συντακτικούς τύπους που έχουν ορισθεί για το LDAP παρατίθενται στον ακόλουθο πίνακα.

Συντακτικός τύπος	**Περιγραφή**
bin	Δυαδική πληροφορία.
ces	case exact string, γνωστό επίσης ως ‘’string καταλόγου’’, είναι σημαντικό κατά την διάρκεια των συγκρίσεων.
cis	case ignore string. Δεν είναι σημαντικό κατά την διάρκεια των συγκρίσεων.
tel	Τηλεφωνικός αριθμός. Οι αριθμοί συμπεριφέρονται σαν κείμενο, αλλά τα κενά και οι παύλες παραλείπονται.
dn	Διακριτό όνομα.
Generalized Time	Το έτος, ο μήνας, η ημέρα και ο χρόνος αναπαριστώνται ως μια εκτυπώσιμη σειρά.
Postal Address	Διεύθυνση ταχυδρομείου με γραμμές που χωρίζονται από τον χαρακτήρα ‘’$’’.

Πίνακας 2.2.1.α’ Μερικοί από τους συντακτικούς τύπους χαρακτηριστικών LDAP.

Ο πίνακας που ακολουθεί παρουσιάζει κάποια κοινά χαρακτηριστικά. Μερικά από αυτά έχουν ψευδώνυμα (aliases) τα οποία μπορούν να χρησιμοποιηθούν οπουδήποτε χρησιμοποιείται το πλήρες όνομα του χαρακτηριστικού. Παραδείγματος χάριν, το cn χρησιμοποιείται όταν αναφερόμαστε στο χαρακτηριστικό commonName.

Χαρακτηριστικό, Alias	**Συντακτικός τύπος**	**Περιγραφή**	**Παράδειγμα**
commonName, cn	cis	Κοινό όνομα μιας καταχώρησης	John Smith
surname, sn	cis	Το επίθετο ενός ατόμου	Smith
telephoneNumber	tel	Τηλεφωνικός αριθμός	512-838-6008
organizationalUnitName, ou	cis	Όνομα μιας μονάδας οργανισμού	itso
owner	dn	Διακριτό όνομα του ατόμου στο οποίο ανήκει η καταχώρηση	cn= John Smith o= IBM, c=US
organization, o	cis	Όνομα ενός οργανισμού	IBM
jpegPhoto	bin	Φωτογραφία με JPEG format	Φωτογραφία του John Smith

Πίνακας 2.2.1.β’ Κοινά χαρακτηριστικά LDAP

Οι περιορισμοί μπορούν να συνδυασθούν με τους τύπους των χαρακτηριστικών για να περιορίσουν τον αριθμό των τιμών που αποθηκεύονται στο χαρακτηριστικό ή να περιορίσουν το συνολικό μέγεθος μιας τιμής. Για παράδειγμα, ένα χαρακτηριστικό το οποίο περιέχει μια φωτογραφία θα μπορούσε να μειώσει την χωρητικοτητά του στα 10 KB έτσι ώστε να αποτραπεί η χρήση αδικαιολόγητων ποσοτήτων αποθηκευτικού χώρου. Ή ένα χαρακτηριστικό που συνήθιζε να αποθηκεύει έναν social security number θα μπορούσε να περιοριστεί στην φύλαξη μιας μόνο τιμής.

Τα σχήματα καθορίζουν τον τύπο των αντικειμένων που αποθηκεύονται στον κατάλογο. Επιπλέον απαριθμούν τα χαρακτηριστικά κάθε τύπου αντικειμένου και ξεχωρίζουν ποια από αυτά είναι υποχρεωτικά και ποια προαιρετικά. Παραδείγματος χάριν, στο σχήμα person, απαιτείται το χαρακτηριστικό surname (sn) αλλά το description είναι προαιρετικό. Ακόμα ο έλεγχος του σχήματος εξασφαλίζει ότι τα χαρακτηριστικά που δεν βρίσκονται στο σχήμα δεν είναι αποθηκευμένα στην καταχώρηση. Τα προαιρετικά χαρακτηριστικά μπορούν να συμπληρωθούν οποιαδήποτε στιγμή. Το σχήμα επίσης καθορίζει την μεταφορά και την υποκατηγοριοποίηση των αντικειμένων καθώς και την ακριβή θέση εμφάνισης τους στην ιεραρχία του DIT.
Ο πίνακας 2.2.1.γ' δείχνει μερικά κοινά σχήματα (κατηγορίες αντικειμένων και τα απαιτούμενα χαρακτηριστικά τους). Σε πολλές περιπτώσεις, μια καταχώρηση μπορεί να περιέχει περισσότερες από μια κατηγορίες αντικειμένου:

Κατηγορία αντικειμένου	**Περιγραφή**	**Απαιτούμενα χαρακτηριστικά**
InetOrgPeson	Ορίζει καταχωρήσεις για ένα άτομο	commonName (cn) surname (cn) objectClass
organizationalUnit	Ορίζει καταχωρήσεις για μονάδες οργανισμού	ou objectClass
organization	Ορίζει καταχωρήσεις για οργανισμούς	o objectClass

Πίνακας 2.2.1.γ' Κατηγορίες αντικειμένων και τα απαιτούμενα χαρακτηριστικά τους

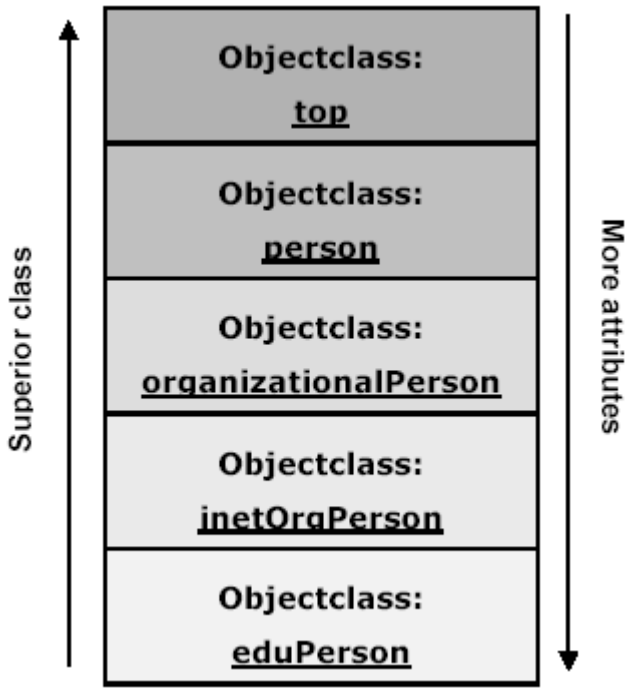
Αν και ο κάθε server μπορεί από μόνος του να προσδιορίσει το δικό του σχήμα, λόγω της διαλειτουργικότητας είναι αναμενόμενο ότι πολλά σχήματα τυποποιούνται (RFC 2252 και RFC 2256). Υπάρχουν κάποιες φορές που θα προκύψει η ανάγκη ενός νέου σχήματος από έναν συγκεκριμένο server ή από μια επιχείρηση. Στην έκδοση 3 του LDAP, ένας server υποχρεούται να επιστρέφει πληροφορίες που αφορούν τον ίδιο και το σχήμα που χρησιμοποιεί. Επομένως ένα πρόγραμμα μπορεί να ζητήσει από τον server να προσδιορίσει τα περιεχόμενα του σχήματος. Αυτή η πληροφορία του server αποθηκεύεται στο ειδικό μηδενικού μήκους DN.

Τα αντικείμενα μπορούν να προέλθουν από άλλα αντικείμενα μέσω μιας διαδικασίας που ονομάζεται κληρονομικότητα (inheritance). Σε αυτή την περίπτωση η «κληρονόμος» objectClass κληρονομεί κάποια χαρακτηριστικά της «κληρονομούμενης» objectClass και αυτό είναι γνωστό ως *υποκατηγοριοποίηση* (subclassing ή objectClass inheritance). Για παράδειγμα ας υποθέσουμε ότι το αντικείμενο που ονομάζεται person έχει οριστεί να περιέχει ένα surname και ούτω καθεξής. Μια κατηγορία αντικειμένου με το όνομα organizationalPerson μπορούσε να ορισθεί ως μια υποκατηγορία της κατηγορίας person. Η κατηγορία

organizationalPerson θα έχει τα ίδια χαρακτηριστικά με την person και μπορεί να προσθέσει και άλλα όπως τα title και officenumber. Η person θα αποκαλείται η ανώτατη (superior) της organizationalPerson. Μια ειδική κατηγορία αντικειμένου που ονομάζεται ανώτερη (top), δεν έχει άλλες που να βρίσκονται πιο πάνω από αυτήν. Η ανώτερη κατηγορία περιλαμβάνει το υποχρεωτικό χαρακτηριστικό objectClass. Τα χαρακτηριστικά της ανώτερης εμφανίζονται σε όλες τις καταχωρήσεις καταλόγου όπως έχουν καθοριστεί (προαιρετικά ή υποχρεωτικά).

Κάθε καταχώρηση καταλόγου έχει ένα ειδικό χαρακτηριστικό που ονομάζεται objectClass. Η τιμή του χαρακτηριστικού objectClass είναι μια λίστα δυο ή περισσότερων ονομάτων σχημάτων. Τα σχήματα αυτά καθορίζουν το είδος των αντικειμένων που αντιπροσωπεύει η καταχώρηση. Μια από τις τιμές πρέπει να είναι είτε top είτε alias. Το alias χρησιμοποιείται όταν η καταχώρηση παίζει ρόλο alias (ψευδωνύμου) σε μια άλλη καταχώρηση. Σε διαφορετική περίπτωση χρησιμοποιείται η top. Το χαρακτηριστικό objectClass προσδιορίζει τα χαρακτηριστικά που πρέπει και μπορεί να έχει η καταχώρηση.

Η ειδική κατηγορία αντικειμένου extensibleObject επιτρέπει σε κάθε χαρακτηριστικό να αποθηκεύεται στην καταχώρηση. Αυτό μπορεί να είναι πιο βολικό από το να ορισθεί μια νέα κατηγορία αντικειμένου ώστε να προστεθεί ένα ειδικό χαρακτηριστικό σε κάποιες καταχωρήσεις, αλλά επίσης ανοίγει το αντικείμενο ώστε να είναι ικανό να περιέχει οτιδήποτε (το οποίο μπορεί να μην είναι καλό σε ένα δομημένο σύστημα).



Σχήμα 2.2.1.δ' Διαδικασία κληρονομικότητας (objectClass inheritance)

Στο σχήμα 2.2.1.δ' φαίνεται περιγραφικά η διαδικασία κληρονομικότητας για πέντε standard objectClasses. Όπως παρατηρούμε η objectClass eduPerson κληρονομεί την κλάση inetOrgPerson. Στη συνέχεια η inetOrgPerson κληρονομεί την κλάση Person μέσω της ενδιάμεσης κλάσης organizationalPerson. Εμπεριέχει συμπληρωματικά attributes, ώστε να μπορεί να περιέχει δεδομένα που έχουν ευρεία χρήση στο Internet και μέσα σε οργανισμούς. Η objectClass organizationalPerson

κληρονομεί την Person και η person κληρονομεί την αφηρημένη κλάση top. Τέλος, η top αποτελεί μια ιδιαίτερη κλάση από την οποία όλες οι άλλες άμεσα ή έμμεσα κληρονομούν χαρακτηριστικά της. Γενικότερα, η κλάση από την οποία μια άλλη κλάση κληρονομεί χαρακτηριστικά ονομάζεται superior ή superclass. Θα μπορούσε να ειπωθεί ότι η κλάση top είναι η superclass της κλάσης Person.

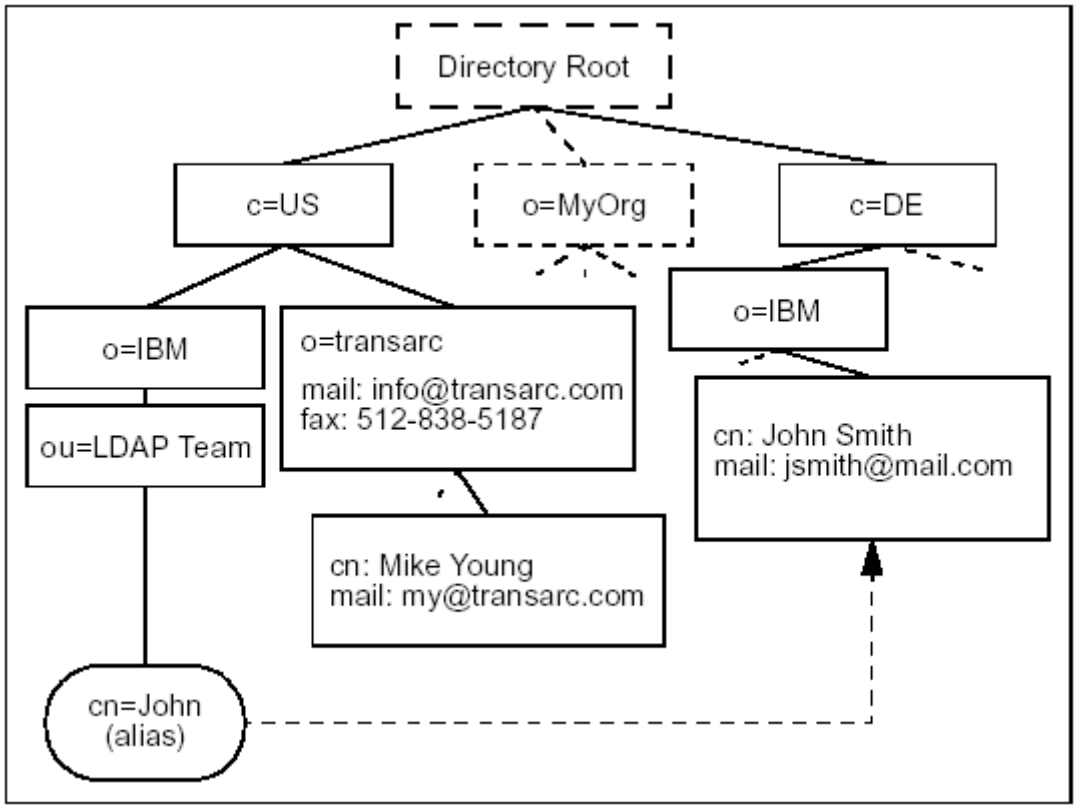
Όταν μια κλάση κληρονομεί μια άλλη, κληρονομεί το σύνολο των υποχρεωτικών χαρακτηριστικών, το σύνολο των προαιρετικών χαρακτηριστικών, καθώς και τον τύπο της κλάσης την οποία κληρονομεί. Εξυπακούεται ότι μια κλάση κληρονομώντας μια άλλη, αναγκαστικά κληρονομεί και τα χαρακτηριστικά της superclass της.

Συγκεκριμένα για την κλάση eduPerson, αναφέρεται ότι αποτελεί καρπό της συνεργασίας πανεπιστημιακών ιδρυμάτων, κυβερνητικών φορέων και ιδιωτών, στα πλαίσια του internet2 και του EDUCAUSE. Η κλάση eduPerson είναι μια βοηθητική κλάση για τις Υπηρεσίες Καταλόγου των πανεπιστημιακών ιδρυμάτων και εμπεριέχει χαρακτηριστικά τα οποία συναντά κανείς στην πλειοψηφία αυτών. Απώτερος στόχος είναι να αποτελέσει η συγκεκριμένη κλάση προδιαγραφή για τις Υπηρεσίες Καταλόγου πανεπιστημιακών ιδρυμάτων και αυτό επειδή είναι σχεδιασμένη με τέτοιο τρόπο, ώστε να διευκολύνει την διαπανεπιστημιακή επικοινωνία κάνοντας χρήση εξελιγμένων υπηρεσιών.

2.2.2 Το Μοντέλο Ονομασίας

Το μοντέλο ονομασίας LDAP καθορίζει τον τρόπο με τον οποίο οι καταχωρήσεις αναγνωρίζονται και οργανώνονται. Οι καταχωρήσεις είναι οργανωμένες υπό μορφή δέντρου που ονομάζεται Directory Information Tree (DIT) και τακτοποιούνται μέσα στο DIT με βάση το DN τους. Ένα DN είναι ένα μοναδικό όνομα που σαφώς προσδιορίζει μια μοναδική καταχώρηση. Τα DN είναι δημιουργημένα από μια ακολουθία σχετικών διακριτών ονομάτων (RDNs). Κάθε RDN που βρίσκεται σε ένα DN αντιστοιχεί σε ένα κλαδί του DIT που ξεκινάει από τη ρίζα του δέντρου ως την καταχώρηση του καταλόγου. Κάθε RDN προέρχεται από τα χαρακτηριστικά της καταχώρησης καταλόγου. Στην απλή και συνηθισμένη περίπτωση, ένα RDN έχει την μορφή <όνομα χαρακτηριστικού> = <τιμή>. Ένα DN αποτελείται από την ακολουθία των RDNs που χωρίζονται με κόμματα.

Ένα παράδειγμα ενός DIT φαίνεται στο σχήμα 2.2.2.α'. Το παράδειγμα είναι πολύ απλό αλλά μπορεί να χρησιμοποιηθεί και για την επεξήγηση μερικών βασικών εννοιών. Κάθε πλαίσιο αντιπροσωπεύει μια καταχώρηση καταλόγου. Η καταχώρηση ρίζας είναι εννοιολογική και στην πραγματικότητα δεν υπάρχει. Τα χαρακτηριστικά παρατίθενται μέσα σε κάθε καταχώρηση. Η λίστα των χαρακτηριστικών που παρουσιάζεται δεν είναι ολοκληρωμένη. Για παράδειγμα, η καταχώρηση για την χώρα DE (c=DE) θα μπορούσε να έχει το χαρακτηριστικό description (περιγραφή) με την τιμή Germany (Γερμανία).



Σχήμα 2.2.2.α'. Παράδειγμα ενός DIT

Η οργάνωση των καταχωρήσεων στο DIT περιορίζεται από τους αντίστοιχους ορισμούς των κατηγοριών αντικειμένου. Είναι συνηθισμένο να ακολουθούμε ένα γεωγραφικό ή οργανωτικό σχήμα. Παραδείγματος χάριν, οι καταχωρήσεις που αντιπροσωπεύουν χώρες θα τοποθετούνταν στην κορυφή του DIT. Κάτω από τις χώρες θα βρίσκονταν διεθνείς οργανισμοί, πόλεις, επαρχίες και ούτω καθεξής. Κάτω από αυτό το επίπεδο, οι καταχωρήσεις μπορεί να αφορούν άτομα που βρίσκονται σε αυτούς τους οργανισμούς ή περαιτέρω υποδιαιρέσεις του οργανισμού. Τα χαμηλότερα επίπεδα των καταχωρήσεων του DIT μπορούν να αναπαριστούν οποιοδήποτε αντικείμενο όπως άτομα, εκτυπωτές, εξυπηρετητές εφαρμογών και τα λοιπά. Το βάθος ή το εύρος του DIT δεν είναι απαγορευτικό και μπορεί να σχεδιαστεί ώστε να ταιριάζει με τις απαιτήσεις των εφαρμογών.

Οι καταχωρήσεις ονομάζονται σύμφωνα με τη θέση τους στο DIT. Η καταχώρηση που βρίσκεται χαμηλά δεξιά στο Σχήμα 2.2.2.α' έχει το DN cn=John Smith, o=IBM, c=DE. Σημειώνεται ότι συνήθως τα DNs διαβάζονται από τα φύλλα προς τη ρίζα σε αντίθεση με τα ονόματα αρχείου συστήματος που συνήθως διαβάζονται από την ρίζα προς τα φύλλα. Όπως αναφέρθηκε, τα DNs αποτελούνται από την ακολουθία των RDNs. Κάθε RDN είναι κατασκευασμένο από ένα χαρακτηριστικό (ή χαρακτηριστικά) της καταχώρησης που ονομάζει. Παραδείγματος χάριν, το DN cn=John Smith, o=IBM, c=DE έχει κατασκευαστεί από την πρόσθεση του RDN cn=John Smith στο DN o=IBM, c=DE της προηγούμενης καταχώρησης. Σημειώνεται το cn=John Smith είναι χαρακτηριστικό στην καταχώρηση cn=John Smith, o=IBM, c=DE. Το DN μιας καταχώρησης προσδιορίζεται όταν δημιουργείται. Επίσης θα ήταν νόμιμο να έχει δημιουργηθεί μια καταχώρηση με το DN mail=jsmith@mail.com, o=IBM, c=DE.

Το DIT περιγράφεται ως μια μορφή δέντρου αλλά δεν είναι δέντρο. Αυτό συμβαίνει εξ αιτίας των ψευδωνύμων (aliases). Τα ψευδώνυμα επιτρέπουν στη δομή

δένδρου να είναι παρακάμψιμη. Αυτό μπορεί να είναι χρήσιμο αν μια καταχώρηση ανήκει σε περισσότερους από έναν οργανισμούς ή αν ένα χρησιμοποιούμενο DN είναι πολύπλοκο. Μια άλλη συνηθισμένη χρήση των ψευδωνύμων είναι όταν οι καταχωρήσεις μετακινούνται μέσα στο DIT και επιθυμούμε πρόσβαση για να συνεχίσουμε να εργαζόμαστε όπως πριν. Στο Σχήμα 2.2.2.α' το cn=John, ou=LDAP Team, o=IBM, c=US είναι ένα alias για το cn= John Smith, o= IBM, c=DE. Τα ψευδώνυμα δεν χρειάζεται να δείχνουν στις καταχωρήσεις φύλλων στο DIT. Για παράδειγμα, το o=Book, c=US θα μπορούσε να ήταν ψευδώνυμο για το ou=ITSO, o=IBM, c=US.

2.2.2.1 Το Συντακτικό των DNs

Τα DNs χρησιμοποιούνται ως πρωτεύοντα κλειδιά στις καταχωρήσεις καταλόγου. Το LDAP ορίζει μια string αναπαράσταση των DNs. Το Σχήμα 2.2.2.1.α' δείχνει την επίσημη γραμματική των DNs. Σημειώνεται ότι τα RDNs μπορεί να γίνουν πιο πολύπλοκα σε αντίθεση με τα παραδείγματα που περιγράφηκαν παραπάνω. Ένα RDN μπορεί να αποτελείται από ποικίλα χαρακτηριστικά συνοδευόμενα από το ''+'' όπως στο DN cn=John Smith+1=Stuttgart, o=IBM, c=DE. Εάν οι τιμές των χαρακτηριστικών περιέχουν ειδικούς χαρακτήρες πρέπει να γίνεται διαφυγή αυτών με την χρήση του χαρακτήρα ''\'' πριν από αυτούς. Το ακόλουθο DN περιέχει ένα κόμμα

o=Transarc\, Inc., c=US.

Τα DNs στην έκδοση 3 του LDAP είναι περισσότερο περιοριστικά απ' ότι στην έκδοση 2. Παραδείγματος χάριν, στο LDAPv2 ο ειδικός χαρακτήρας '' ; '' μπορεί επίσης να χωρίζει τα RDNs. Στο LDAPv3 πρέπει να είναι αποδεκτή η παλαιότερη σύνταξη αλλά όμως δεν πρέπει να δημιουργούνται DNs που δεν προσαρμόζονται με το νεότερο συντακτικό τύπο. Η ακριβής γραμματική για τον συντακτικό τύπο ενός DN φαίνεται στον Πίνακα 2.2.2.1.α'.

```
distinguishedName = [name]                          , may be empty string

name         = name-component *(“,” name-component)

name-component = attributeTypeAndValue * (“+” attributeTypeAndValue)

attributeTypeAndValue = attributeType ”=” attributeValue

attributeType = (ALPHA 1* keychar) / oid

keychar        = ALPHA / DIGIT / “-“

oid              = 1*DIGIT * (“.” 1*DIGIT)

attributeValue = string

string           = *(stringchar / pair)
                     / “#” hexstring
                     QUOTATION * ( quotechar / pair ) QUOTATION ; only from v2
```

```
quotechar     = <any character except "\" or QUOTATION >

special         = "," / "=" / "+" / "<" / ">" / "#" / ";"
pair          = "\" ( special /  "\"  / QUOTATION / hexpair )

stringchar  = <any character except one of special, "\" or QUOTATION >

hexstring   = 1*hexpair

hexpair      = hexchar hexchar

hexchar      = DIGIT / "A" / "B" / "C" / "D" / "E" / "E" / "F"
                 /  "a" / "b" / "c" / "d " / "e" / "f "

ALPHA      = <any ASCII alphabetic character>          ; (decimal 65-90 and 97-122)

DIGIT        = <any ASCII decimal digit>                  ; (decimal 48-57)

QUOTATION= <the ASCII double quotation mark character ' " ' decimal 34>
```

Πίνακας 2.2.2.1.α' Γραμματική DN

Οι τύποι των χαρακτηριστικών που χρησιμοποιήθηκαν στο RDN μπορούν να αναπαρίστανται από ένα δεκαδικό string που κωδικοποιεί το αναγνωριστικό του αντικειμένου του. Για παράδειγμα, το cn=John μπορούσε επίσης να είχε γραφθεί ως 2.5.4.2=John. Ωστόσο, τα ονόματα των χαρακτηριστικών που χρησιμοποιούνται συχνά, έχουν μια αναπαράσταση string που προφανώς είναι πιο εύκολο να κατανοηθεί. Ο Πίνακας 2.2.2.1.β' απαριθμεί μερικούς από τους πιο κοινούς τύπους χαρακτηριστικών και τις στοιχειοσειρές τους.

Attribute Type	**String**
CommonName	CN
LocalityName	L
StateOrProvinceName	ST
OrganizationName	O
OrganizationalUnitName	OU
CountryName	C
StreetAddress	STREET
DomainComponent	DC
Userid	UID

Πίνακας 2.2.2.1.β' Τύποι και στοιχειοσειρές χαρακτηριστικών

2.2.2.2 Επιθέματα και Παραπομπές

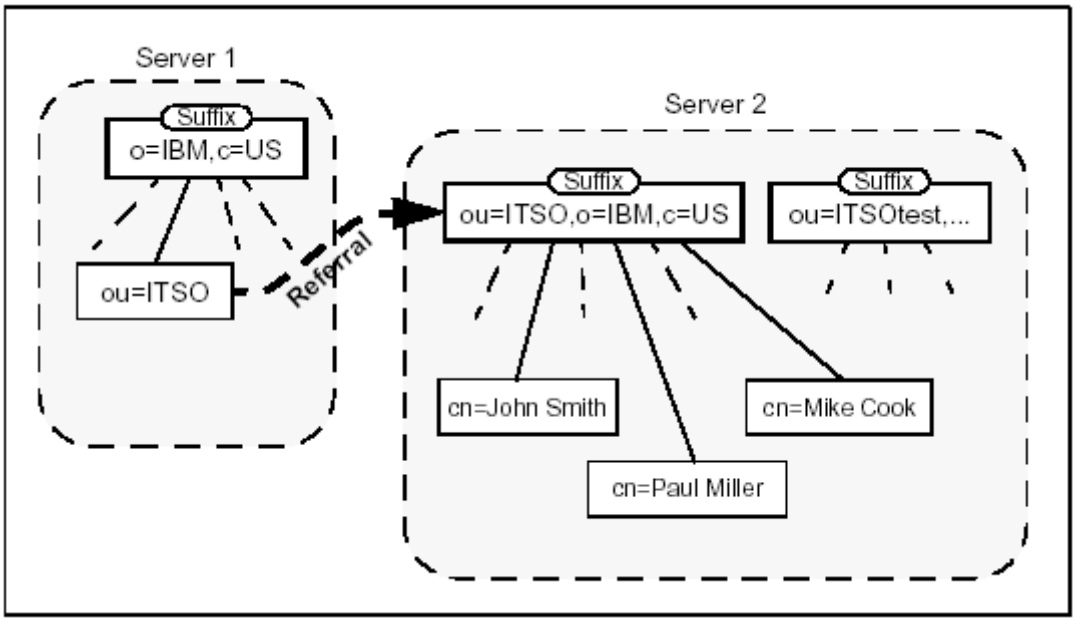
Ένας αυτόνομος LDAP server ίσως να μην αποθηκεύει ολόκληρο το DIT. Ένας server μπορεί να αποθηκεύει τις καταχωρήσεις ενός συγκεκριμένου τμήματος και όχι τις καταχωρήσεις για αυτούς που προηγούνταν του τμήματος. Παραδείγματος χάριν, ένας server μπορεί να αποθηκεύσει τις καταχωρήσεις για το

τμήμα ITSO της IBM. Ο ανώτερος κόμβος του DIT που αποθηκεύεται από τον server θα είναι το ou=ITSO, o=IBM, c=US. Ο server δεν θα αποθηκεύσει καταχωρήσεις για το c=US ή για o=IBM, c=US. Η ανώτατη καταχώρηση που αποθηκεύεται από έναν server ονομάζεται *επίθεμα*. Κάθε καταχώρηση λήγει σε ένα επίθεμα (σημειώνεται ότι στην σύνταξη των DNs οι –ανωτέρου επιπέδου- καταχωρήσεις βρίσκονται στο τέλος).

Ένας server μπορεί να υποστηρίξει ποικίλα επιθέματα. Για παράδειγμα, εκτός από την αποθήκευση πληροφοριών για το ITSO τμήμα, ο ίδιος server θα μπορούσε να αποθηκεύσει πληροφορίες για το τμήμα πωλήσεων στο Transarc. Σε αυτή την περίπτωση ο server θα είχε τα επιθέματα ou=ITSO, o=IBM, c=US και ou=sales, o=Transarc, c=US.

Από την στιγμή που ο server δεν αποθηκεύει ολόκληρο το δένδρο, οι servers θα πρέπει να συνδέονται με κάποιον τρόπο μεταξύ τους ώστε να σχηματίσουν ένα κατανεμημένο κατάλογο που θα περιέχει ολόκληρο το DIT. Αυτό κατορθώνεται με τις *παραπομπές*. Συνεχίζοντας το παράδειγμα, ένας άλλος server ίσως να αποθηκεύσει την καταχώρηση o=IBM, c=US αλλά όχι πληροφορίες για το τμήμα ITSO. Εάν κάποιος αναζητήσει πληροφορία σε αυτόν τον κατάλογο για το τμήμα ITSO, δεν θα βρει καμία. Ωστόσο ο server μπορεί να αποθηκεύσει μια παραπομπή στον LDAP server που θα περιέχει την πληροφορία. Η παραπομπή αυτή συμπεριφέρεται ως δείκτης ο οποίος ακολουθεί την επιθυμητή πληροφορία όπου αυτή βρίσκεται. Ένα τέτοιο παράδειγμα φαίνεται στο Σχήμα 2.2.2.2.α' όπου το βέλος της παραπομπής δείχνει την λογική σύνδεση της παραπομπής και όχι την τεχνική εφαρμογή.

Μια παραπομπή είναι μια καταχώρηση της παραπομπής objectClass. Έχει ένα χαρακτηριστικό, το ref, του οποίου η τιμή είναι το LDAP URL της καταχώρησης που παραπέμπεται σε έναν άλλο LDAP server.

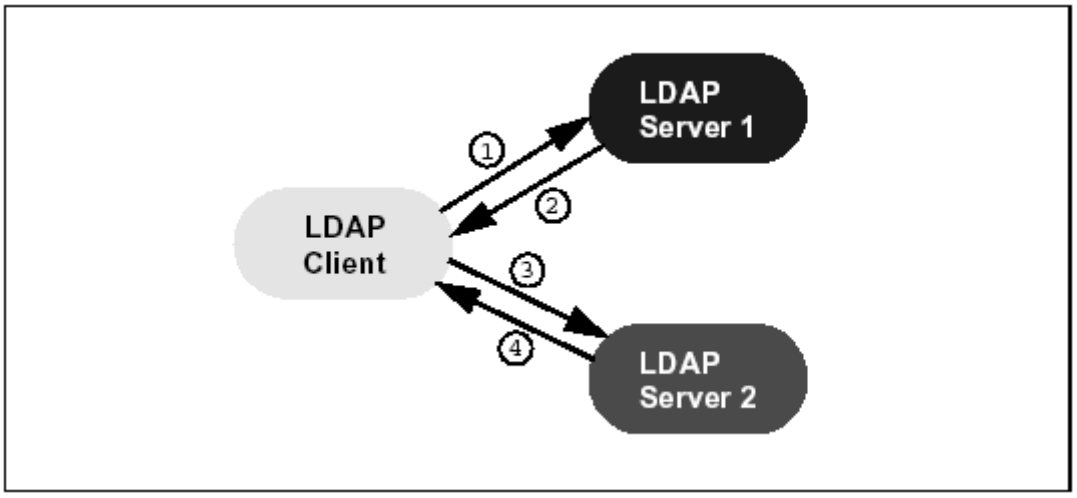


Σχήμα 2.2.2.2.α' Παράδειγμα DIT με παραπομπές και επιθέματα

Όταν ένας client στέλνει ένα αίτημα στον LDAP server, η απόκριση στον client μπορεί να είναι μια παραπομπή. Έπειτα ο client μπορεί να επιλέξει να ακολουθήσει την παραπομπή με το να ρωτήσει τον άλλον server ο οποίος περιέχεται στην παραπομπή που επιστρέφεται από τον πρώτο LDAP server. Οι παραπομπές δεν

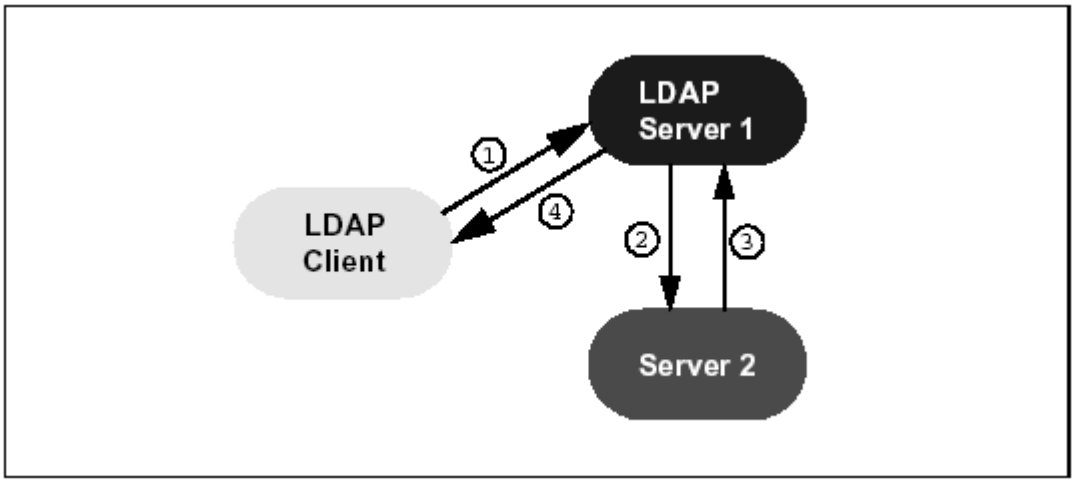
ακολουθούνται από τους servers. Αυτό μπορεί να βελτιώσει την απόδοση του server με το να τεθεί εκτός φόρτωσης η διαδικασία σύνδεσης άλλων servers με τον client.

Το Σχήμα 2.2.2.2.β' απεικονίζει έναν client που ακολουθεί μια παραπομπή. Ένας LDAP client ζητάει πληροφορία από τον LDAP server 1 (1). Το αίτημα απαντάται με μια παραπομπή στον LDAP server 2 (2). Έπειτα ο client συνδέεται με τον LDAP server 2 (3). Ο LDAP server 2 παρέχει τα δεδομένα που έχουν ζητηθεί στον client (4).



Σχήμα 2.2.2.2.β' Παραπομπή που ακολουθείται από τον client

Στο Σχήμα 2.2.2.2.γ' απεικονίζεται η διαδικασία *αλυσίδας*. Ένας LDAP client ζητάει πληροφορία από τον LDAP server 1 (1). Ο LDAP server 1 βρίσκει μια παραπομπή στον server 2 και προωθεί το αίτημα (2).Ο server 2 δίνει τα ζητούμενα δεδομένα στον LDAP server 1 (3). Κατόπιν ο LDAP server 1 επιστρέφει τα αποτελέσματα στον client (4). Σημειώνεται ότι η συγκεκριμένη εξήγηση και το Σχήμα 2.2.2.2.γ' υπάρχουν για την επεξήγηση περιπτώσεων στις οποίες δεν συμπεριλαμβάνεται η διαδικασία αλυσίδας στις προδιαγραφές της έκδοσης 2 ή 3 του LDAP.



Σχήμα 2.2.2.2.γ' Διαδικασία αλυσίδας των server

Το LDAP API επιτρέπει στον προγραμματιστή να διευκρινίζει πότε πρέπει οι παραπομπές να ακολουθούνται αυτόματα ή να επιστρέφονται στο πρόγραμμα. Εάν αυτές ακολουθούνται αυτόματα τότε η βιβλιοθήκη του LDAP client (ούτε ο server

ούτε το πρόγραμμα εφαρμογής) ακολουθεί την παραπομπή. Το γεγονός αυτό δεν απαιτεί επιπλέον κωδικοποίηση και είναι διαφανές στον προγραμματιστή. Για να αποφευχθούν οι μεγάλου μήκους αναζητήσεις ή παραπομπές που (λανθασμένα) σχηματίζουν ένα βρόχο, ο προγραμματιστής μπορεί να μειώσει τον αριθμό των παραπομπών που ακολουθούν ένα αίτημα.

Εάν η παραπομπή επιστρέφεται στο πρόγραμμα, τότε πρέπει να συμπεριλαμβάνεται η κωδικοποίηση για να φαίνεται ότι η παραπομπή έχει επιστρέψει. Η παραπομπή μπορεί να αξιολογηθεί και να αποφασισθεί αν θα ακολουθηθεί ή όχι. Αυτό είναι αρκετά πολύπλοκο αλλά δίνει στον προγραμματιστή καλύτερη δυνατότητα επιλογής για το ποιες παραπομπές πρέπει να ακολουθούνται και ποιες όχι.

Οι παραπομπές επιτρέπουν σε ένα DIT να μοιρασθεί και να κατανεμηθεί σε πολλούς servers. Επίσης τα μέρη του DIT μπορούν να αντιγραφούν. Αυτό μπορεί να βελτιώσει την απόδοση και την διαθεσιμότητα.

Η έκδοση 2 του LDAP δεν καθορίζει επίσημα τις παραπομπές αλλά η έκδοση 3 τις συμπεριλαμβάνει. Ούτε η έκδοση 2 ούτε η 3 καθορίζουν την αλυσίδα, όμως δεν είναι απαγορευτική αν οι προμηθευτές επιθυμήσουν να την εφαρμόσουν. Οι προμηθευτές, για παράδειγμα, μπορούν να επιλέξουν να εφαρμόσουν έναν μηχανισμό αλυσίδας τύπου X.500 ή να παρέχουν λειτουργικότητα μέσω κατανεμημένων Βάσεων Δεδομένων.

2.2.2.3 Πληροφορία του Server

Ένας LDAP server έκδοσης 3 πρέπει να παρέχει πληροφορίες για τον εαυτό του. Η ειδική καταχώρηση που ονομάζεται ρίζα DSE με ένα –μηδενικού μήκους (κενό)- DN περιέχει χαρακτηριστικά που περιγράφουν τον server. Τα χαρακτηριστικά αυτά μπορούν να ανακτηθούν για να αποκαλυφθούν βασικές πληροφορίες για τον server και το DIT που αποθηκεύει. Η πληροφορία για τον server περιλαμβάνει:

- Τα επιθέματα που αποθηκεύει ο server
- Το DN μιας ειδικής καταχώρησης που περιέχει μια λίστα όλων των objectClass και του σχήματος των χαρακτηριστικών που είναι γνωστά στον server.
- Την έκδοση ή τις εκδόσεις του LDAP που υποστηρίζονται
- Μια λίστα με επιπλέον λειτουργίες και ελέγχους
- Μια λίστα των υποστηριζόμενων μηχανισμών ελέγχου SASL
- Μια λίστα των εναλλακτικών LDAP servers

Καθώς το LDAP επεκτείνεται, οι επιπρόσθετες πληροφορίες για τον server θα αποθηκεύονται στην ρίζα DSE.

2.2.3 Το Λειτουργικό Μοντέλο

Το LDAP ορίζει τις λειτουργίες προσπέλασης και τροποιήσης των καταχωρήσεων καταλόγου. Η ενότητα αυτή παρουσιάζει τις λειτουργίες του LDAP με τρόπο ανεξάρτητο από γλώσσες προγραμματισμού. Οι λειτουργίες του LDAP μπορούν να χωρισθούν στις παρακάτω κατηγορίες:

Query	Περιλαμβάνει την αναζήτηση και την σύγκριση λειτουργιών που χρησιμοποιούνται για την ανάκτηση πληροφοριών από τον κατάλογο
Update	Περιλαμβάνει την πρόσθεση, την διαγραφή, την τροποποίηση και την τροποποίηση των RDN που χρησιμοποιούνται για να ενημερώσουν την αποθηκευμένη πληροφορία καταλόγου
Authentication	Περιλαμβάνει τις διαδικασίες σύνδεσης, αποσύνδεσης και ματαίωσης που γίνονται για να συνδεθούμε με έναν LDAP server ή να αποσυνδεθούμε από αυτόν, κατοχυρώνοντας έτσι τα δικαιώματα προσπέλασης και την προστασία των πληροφοριών

Η πιο συνηθισμένη ενέργεια είναι η αναζήτηση. Η διαδικασία της αναζήτησης είναι πολύ ευέλικτη και έχει μερικές από τις πιο περίπλοκες επιλογές.

2.2.3.1 Αναζήτηση (search)

Η λειτουργία της αναζήτησης επιτρέπει σε έναν client να ζητήσει από έναν LDAP server να ψάξει σε κάποια τμήματα του DIT για πληροφορίες που θα ικανοποιούν τα κριτήρια αναζήτησης του χρήστη με σκοπό να διαβάσει και να απαριθμήσει τα αποτελέσματα. Δεν υπάρχουν ξεχωριστές διαδικασίες για την ανάγνωση και την εμφάνιση αποτελεσμάτων. Και οι δύο είναι συνδεδεμένες με την αναζήτηση. Η αναζήτηση μπορεί να είναι γενική ή πολύ ειδική. Αυτή επιτρέπει σε κάποιον να διευκρινίσει το σημείο από όπου ξεκινάει η αναζήτηση μέσα στο DIT, τί βάθος θα έχει η αναζήτηση στο DIT, ποια χαρακτηριστικά πρέπει να έχει μια καταχώρηση για να θεωρηθεί ότι ταιριάζει με τα κριτήρια και ποια χαρακτηριστικά πρέπει να επιστραφούν για τις ταιριασμένες καταχωρήσεις.

Μερικά παραδείγματα αναζητήσεων είναι:

- Βρες την διεύθυνση ταχυδρομείου για το cn=John Smith, o=IBM, c=DE.
- Βρες όλες τις καταχωρήσεις που είναι απόγονοι της καταχώρησης ou=ITSO, o=IBM, c=US.
- Βρες τις διευθύνσεις ηλεκτρονικού ταχυδρομείου και τους τηλεφωνικούς αριθμούς όλων όσων εργάζονται στην IBM που το επιθετό τους περιέχει τους χαρακτήρες ‘’miller’’ και διαθέτουν επίσης αριθμό fax.

Για να εκτελεστεί μια αναζήτηση πρέπει να καθορισθούν οι παρακάτω παράμετροι:

- **Βάση**

  Ένα DN που ορίζει το σημείο εκκίνησης- το οποίο αποκαλείται αντικείμενο βάσης- της αναζήτησης. Το αντικείμενο βάσης είναι ένας κόμβος μέσα στο DIT.

- **Πεδίο**

  Καθορίζει την τοποθεσία μέσα στο δέντρο που θα γίνει η αναζήτηση με αφετηρία το αντικείμενο βάσης. Υπάρχουν τρεις επιλογές: baseObject,

singleLevel και wholeSubtree. Εάν είναι καθορισμένη η baseObject, εξετάζεται μόνο το αντικείμενο βάσης. Εάν καθορίζεται η singleLevel εξετάζονται μόνο τα άμεσα ‘’παιδιά’’ του αντικειμένου βάσης. Το ίδιο το αντικείμενο βάσης δεν εξετάζεται. Στην περίπτωση της wholeSubtree, εξετάζεται το αντικείμενο βάσης και όλοι οι απογονοί του.

- **Φίλτρα αναζήτησης**

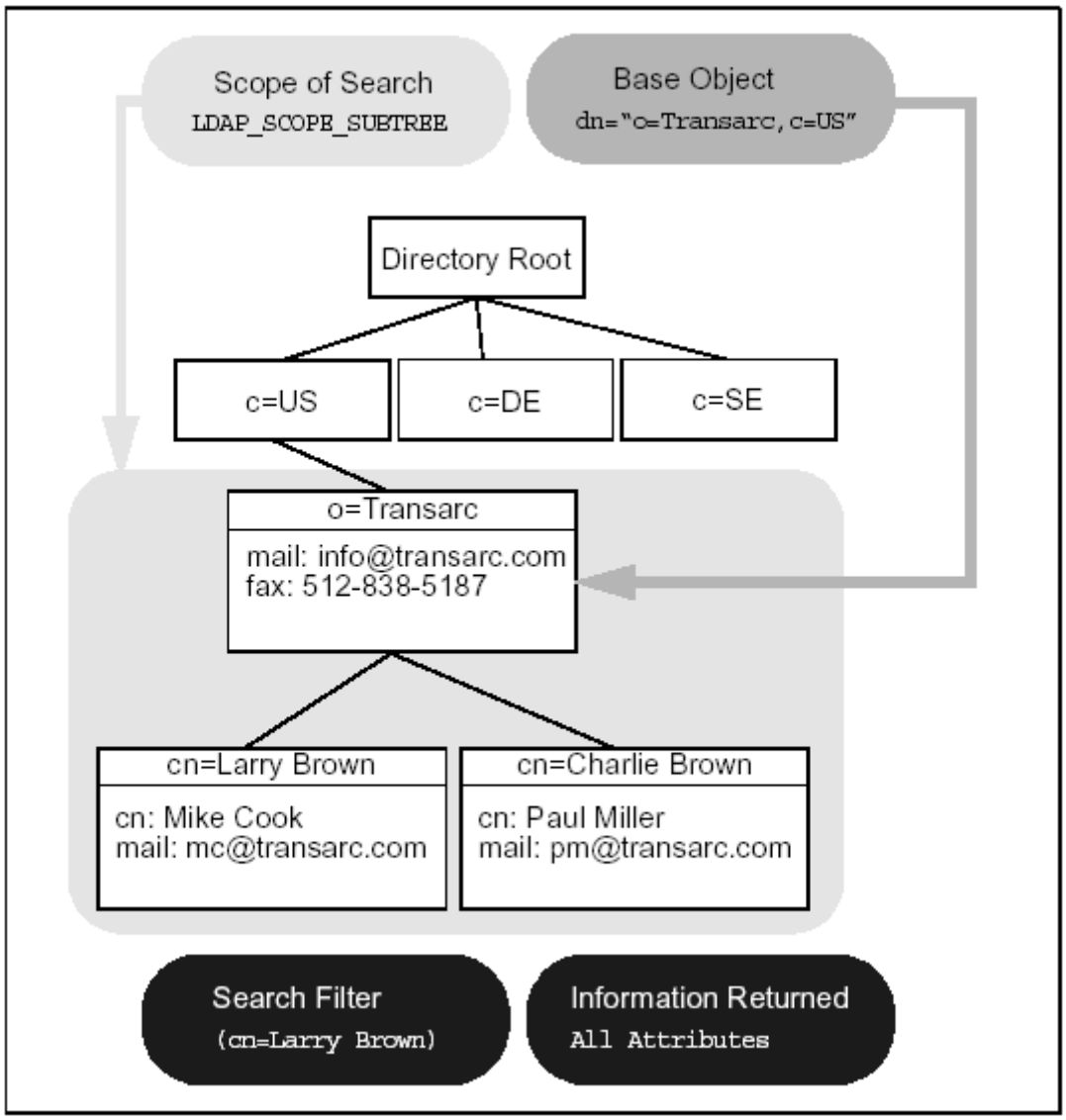
  Ορίζει τα κριτήρια που πρέπει να ταιριάξει μια καταχώρηση για να επιστραφεί από την αναζήτηση. Το φίλτρο αναζήτησης είναι ένας Boolean συνδυασμός των διεκδικούμενων τιμών χαρακτηριστικού. Μια τιμή που πρόκειται να γίνει τιμή χαρακτηριστικού εξετάζει την υπάρχουσα τιμή για το αν είναι μεγαλύτερη, μικρότερη, ίση και ούτω καθεξής. Παραδείγματος χάριν, ένα φίλτρο αναζήτησης μπορεί να ψάχνει για καταχωρήσεις που να περιέχουν τη λέξη ‘’printer’’ ή να ανήκουν στον οργανισμό ITSO.

- **Καταχωρήσεις επιστροφής**

  Διευκρινίζει ποια χαρακτηριστικά ανακτώνται από τις καταχωρήσεις που ταιριάζουν με τα κριτήρια αναζήτησης. Από τη στιγμή που μια καταχώρηση έχει πολλά χαρακτηριστικά, αυτό επιτρέπει στον χρήστη να βλέπει μόνο αυτά που τον ενδιαφέρουν. Φυσιολογικά, ο χρήστης ενδιαφέρεται για την τιμή των χαρακτηριστικών. Ωστόσο, είναι δυνατόν να επιστρέφονται μόνο οι τύποι των χαρακτηριστικών και όχι οι τιμές τους.

- **Όρια**

  Οι αναζητήσεις μπορεί να είναι πολύ γενικές, εξετάζοντας μεγάλα υπόδεντρα και προκαλώντας την επιστροφή πολλών καταχωρήσεων ως αποτελέσματα. Ο χρήστης μπορεί να θέσει όρια μεγέθους και χρόνου για να αποφύγει την δύστροπη αναζήτηση καταναλώνοντας πολλές πηγές. Το όριο μεγέθους περιορίζει των αριθμό καταχωρήσεων που επιστρέφονται από την αναζήτηση. Το χρονικό όριο περιορίζει τον συνολικό χρόνο αναζήτησης. Οι servers είναι ελεύθεροι να επιβάλλουν πιο περιοριστικά όρια ακόμα και αν δεν έχουν ζητηθεί από την πλευρά του client.



Σχήμα 2.2.3.1 Παράμετροι αναζήτησης

2.2.3.2 Παραπομπές και Αναφορές Συνέχειας

Εάν ο server δεν περιέχει το αντικείμενο βάσης, θα επιστρέψει μια παραπομπή στον server που το έχει, εάν αυτό είναι δυνατόν. Από την στιγμή που θα βρεθεί το αντικείμενο βάσης οι αναζητήσεις singleLevel και wholeSubtree ίσως συναντήσουν άλλες παραπομπές. Αυτές οι παραπομπές επιστέφονται στα αποτελέσματα της αναζήτησης μαζί με άλλες παρόμοιες καταχωρήσεις. Οι συγκεκριμένες παραπομπές ονομάζονται αναφορές συνέχειας επειδή δείχνουν πού μπορεί να συνεχιστεί μια αναζήτηση.

Παραδείγματος χάριν, όταν αναζητάμε σε ένα υπόδεντρο οποιονδήποτε που ονομάζεται Smith, μπορεί να επιστραφεί μια αναφορά συνέχειας σε έναν άλλο server πιθανώς μαζί με πολλές άλλες ταιριαστές καταχωρήσεις. Δεν εγγυάται ότι υπάρχει καταχώρηση με το όνομα Smith στον άλλο server, αλλά ότι η αναφορά συνέχειας δείχνει σε ένα υπόδενδρο που θα μπορούσε να περιέχει μια τέτοια καταχώρηση. Εξαρτάται από τον client να ακολουθήσει την αναφορά αν το επιθυμεί. Αφού μόνο η

έκδοση 3 του LDAP καθορίζει τις παραπομπές, οι αναφορές συνέχειας δεν υποστηρίζονται από τις προηγούμενες εκδόσεις.

2.2.3.3 Συντακτικός Τύπος του Φίλτρου Αναζήτησης

Το φίλτρο αναζήτησης ορίζει τα κριτήρια που πρέπει να ταιριάξει μια καταχώρηση για να επιστραφεί από μια αναζήτηση. Το βασικό συστατικό ενός φίλτρου αναζήτησης είναι η διεκδίκηση τιμής χαρακτηριστικού της μορφής:

attribute operator value

Παραδείγματος χάριν, για να αναζητήσουμε ένα άτομο που ονομάζεται John Smith το φίλτρο αναζήτησης θα ήταν cn=John Smith. Σε αυτή την περίπτωση το cn είναι το χαρακτηριστικό, = είναι ο τελεστής και John Smith είναι η τιμή. Ο πίνακας 2.2.3.3.α' απαριθμεί τους τελεστές για τα φίλτρα αναζήτησης.

Τελεστής	Περιγραφή	Παράδειγμα
=	Επιστρέφει καταχωρήσεις των οποίων το χαρακτηριστικό είναι ίσο με την τιμή.	cn=John Smith βρίσκει την καταχώρηση με το όνομα John Smith
>=	Επιστρέφει καταχωρήσεις των οποίων το χαρακτηριστικό είναι μεγαλύτερο από ή ίσο με την τιμή.	sn>=smith βρίσκει όλες τις καταχωρήσεις από το smith μέχρι και το z*
<=	Επιστρέφει καταχωρήσεις των οποίων το χαρακτηριστικό είναι μικρότερο από ή ίσο με την τιμή.	sn<=smith βρίσκει όλες τις καταχωρήσεις από το a* ως το smith
=*	Επιστρέφει καταχωρήσεις που έχουν ένα σύνολο τιμών για το χαρακτηριστικό.	sn=* βρίσκει όλες τις καταχωρήσεις που έχουν το χαρακτηριστικό sn
~=	Επιστρέφει καταχωρήσεις των οποίων η τιμή του χαρακτηριστικού περίπου ταιριάζει με την τιμή που έχει ορισθεί. Τυπικά, αυτός είναι ένας αλγόριθμος που ταιριάζει λέξεις που ηχούν το ίδιο.	sn~=smit ίσως να βρει την καταχώρηση ''sn=smith''

Πίνακας 2.2.3.3.α' Τελεστές αναζήτησης φίλτρου

Ο χαρακτήρας '' * '' ταιριάζει κάθε υποστοιχειοσειρά και μπορεί να χρησιμοποιηθεί με τον τελεστή '' = ''. Παραδείγματος χάριν, cn=J*Smi* θα ταίριαζε με τα John Smith και Jan Smithy.

Τα φίλτρα αναζήτησης μπορούν να συνδυαστούν με τελεστές Boolean για να σχηματισθούν πιο πολύπλοκα φίλτρα. Ο συντακτικός τύπος για τον συνδυασμό φίλτρων είναι:
("&" or "|" (filter1) (filter2) (filter3)…)
("!" (filter))

Οι Boolean τελεστές φαίνονται στον πίνακα 2.2.3.3.β'.

Τελεστής Boolean	Περιγραφή
&	Επιστρέφει τις καταχωρήσεις που ταιριάζουν με όλα τα καθορισμένα κριτήρια αναζήτησης φίλτρου
|	Επιστρέφει καταχωρήσεις που ταιριάζουν με ένα ή με περισσότερα κριτήρια φίλτρου.
!	Επιστρέφει καταχωρήσεις για τις οποίες το φίλτρο δεν είναι αληθές. Ο συγκεκριμένος τελεστής μπορεί να εφαρμοσθεί για ένα φίλτρο μόνο. Το (!(filter1)) είναι έγκυρο, αλλά το (!(filter1) (filter2)) δεν είναι.

Πίνακας 2.2.3.3.β' Τελεστές Boolean

Για παράδειγμα, τα (| (sn=Smith) (sn=Miller) ταιριάζουν με καταχωρήσεις που έχουν το επίθετο Smith ή το Miller. Οι Boolean τελεστές μπορούν επίσης να τοποθετηθούν κατά την εξής σύνταξη: (| (sn=Smith) (& (ou=Austin) (sn=Miller)), η οποία ταιριάζει οποιαδήποτε καταχώρηση με τα επίθετα Smith ή Miller τα οποία επίσης διαθέτει η οργανωτική μονάδα Austin που είναι ένα χαρακτηριστικό.

2.2.3.4 Σύγκριση

Όπως φανερώνει και το όνομά της, η λειτουργία της σύγκρισης συγκρίνει μια καταχώρηση για μια τιμή χαρακτηριστικού. Αν η καταχώρηση έχει αυτή την τιμή, η σύγκριση επιστρέφει την τιμή TRUE. Αλλιώς επιστρέφει την FALSE. Αν και η σύγκριση είναι απλούστερη από μια αναζήτηση, είναι περίπου η ίδια με την αναζήτηση πεδίου βάσης με ένα φίλτρο αναζήτησης χαρακτηριστικού που είναι ίσο με την τιμή (attribute=value). Η διαφορά βρίσκεται στο ότι αν η καταχώρηση δεν έχει καθόλου το χαρακτηριστικό (το χαρακτηριστικό δεν εμφανίζεται), η αναζήτηση θα επιστρέψει ότι δεν βρήκε κάτι. Από την άλλη πλευρά, η σύγκριση θα επιστρέψει FALSE. Αυτό δείχνει ότι η καταχώρηση υπάρχει αλλά δεν διαθέτει χαρακτηριστικά που ταιριάζουν με τα κριτήρια της καθορισμένης τιμής.

2.2.3.5 Διαδικασίες Ενημέρωσης (UPDATE)

Οι λειτουργίες ενημέρωσης τροποποιούν τα περιεχόμενα του καταλόγου. Ο πίνακας 2.2.3.5 συνοψίζει τις συγκεκριμένες λειτουργίες.

Λειτουργία	Περιγραφή
Add	Εισάγει νέες καταχωρήσεις μέσα στον κατάλογο
Delete	Διαγράφει υπάρχουσες καταχωρήσεις από τον κατάλογο. Μόνο οι κόμβοι των φύλλων μπορούν να διαγραφούν.
Modify	Αλλάζει τα χαρακτηριστικά και τις τιμές που περιέχονται σε μια υπάρχουσα καταχώρηση. Επιτρέπει να προστεθούν νέα χαρακτηριστικά και να διαγραφούν ή να τροποποιηθούν ισχύοντα χαρακτηριστικά.

Modify DN	Αλλάζει το λιγότερο σημαντικό στοιχείο ενός DN ή μετακινεί ένα υπόδεντρο καταχωρήσεων σε νέα τοποθεσία του DIT. Οι καταχωρήσεις δεν μπορούν να μετακινηθούν κατά μήκος των ορίων του server.

Πίνακας 2.2.3.5 Λειτουργίες ενημέρωσης

2.2.3.6 Λειτουργίες Επικύρωσης (AUTHENTICATION)

Οι λειτουργίες επικύρωσης χρησιμοποιούνται για να ξεκινήσουν ή να ολοκληρώσουν μια σύνοδο ανάμεσα σε έναν LDAP client και σε έναν LDAP server. Η σύνοδος μπορεί να διασφαλισθεί σε πολλά επίπεδα αρχίζοντας από μια επισφαλή ανώνυμη σύνοδο, δηλαδή μια ανεπικύρωτη σύνοδο κατά την οποία ο client γνωστοποιεί τον εαυτό του μέσω ενός password και, καταλήγοντας σε μια ασφαλή, κρυπτογραφημένη σύνοδο χρησιμοποιώντας μηχανισμούς SASL. Το SASL προστέθηκε στην έκδοση 3 του LDAP για να υπερνικήσει την αδύναμη επικύρωση του LDAPv2 (κάποιοι προμηθευτές, ωστόσο, έχουν προσθέσει ισχυρότερες μεθόδους επικύρωσης όπως το Kerberos στην LDAPv2). Στον πίνακα 2.2.3.6 συνοψίζονται οι λειτουργίες επικύρωσης.

Λειτουργία	**Περιγραφή**
Bind	Αρχικοποιεί μια σύνοδο LDAP μεταξύ client και server. Επιτρέπει στον client να αποδείξει την ταυτοτητά του με το να επικυρώσει τον εαυτό του στον server.
Unbind	Ολοκληρώνει την σύνοδο client/server.
Abandon	Επιτρέπει σε έναν client να ζητήσει από τον server να εγκαταλείψει μια λειτουργία.

Πίνακας 2.2.3.6 Λειτουργίες επικύρωσης

2.2.3.7 Έλεγχοι και Εκτεταμένες Λειτουργίες

Οι έλεγχοι και οι εκτεταμένες λειτουργίες επιτρέπουν στο πρωτόκολλο LDAP να επεκταθεί χωρίς να αλλάξει το ίδιο το πρωτόκολλο. Οι έλεγχοι τροποποιούν τη συμπεριφορά μιας λειτουργίας και οι εκτεταμένες λειτουργίες προσθέτουν νέες λειτουργίες στο πρωτόκολλο LDAP. Η λίστα των ελέγχων και των επεκτάσεων που υποστηρίζονται από έναν LDAP server μπορούν να προμηθευτούν από την εξέταση του κενού DN στον server.

Οι έλεγχοι μπορούν να καθορισθούν για την επέκταση κάθε λειτουργίας και προστίθενται στο τέλος του μηνύματος της λειτουργίας του πρωτοκόλλου. Παρέχονται ως παράμετροι στις συναρτήσεις του API. Στο μέλλον, οι τυποποιημένοι έλεγχοι ίσως να καθορίζονται στα σχετικά LDAP RFCs.

Ένας έλεγχος έχει μια -δεκαδικής στοιχειοσειράς- ταυτότητα αντικειμένου που χρησιμεύει για την αναγνωρισή του, μια αυθαίρετη τιμή ελέγχου που φυλάσσει παραμέτρους για τον έλεγχο, και ένα επίπεδο κρισιμότητας. Όταν το επίπεδο κρισιμότητας είναι TRUE ο server πρέπει να υποστηρίζει τον έλεγχο ή αν δεν τον υποστηρίξει τότε πρέπει να απορρίψει ολόκληρη την λειτουργία. Εάν το επίπεδο

κρισιμότητας είναι FALSE, ο server που δεν υποστηρίζει τον έλεγχο πρέπει να εκτελέσει την λειτουργία όπως θα γινόταν αν δεν είχε καθορισθεί έλεγχος. Για παράδειγμα, ένας έλεγχος μπορεί να παρατείνει την διαδικασία διαγραφής προκαλώντας μια εγγραφή λογιστικού ελέγχου η οποία θα καταγράφεται σε ένα αρχείο που προσδιορίζεται από την πληροφορία της τιμής του ελέγχου.

Μια εκτεταμένη λειτουργία επιτρέπει να καθορισθεί μια εντελώς νέα λειτουργία. Το μήνυμα του πρωτοκόλλου της εκτενούς λειτουργίας περιέχει μια ταυτότητα αντικειμένου δεκαδικού string που χρησιμοποιείται για την αναγνώριση της εκτενούς λειτουργίας και ένα αυθαίρετο string των δεδομένων της συγκεκριμένης λειτουργίας.

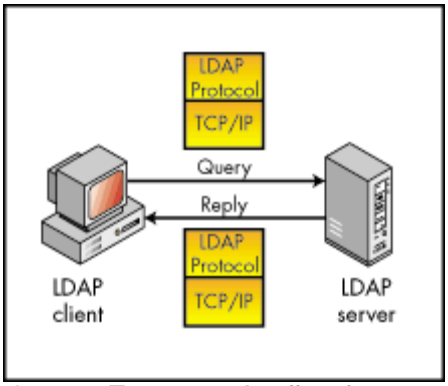
2.2.4 Το Μοντέλο Ασφάλειας

Η ασφάλεια των πληροφοριών που αποθηκεύονται στον κατάλογο είναι αντικείμενο σημαντικής μελέτης. Μερικοί κατάλογοι προορίστηκαν να προσεγγίζονται δημόσια μέσω του Internet αλλά ο κάθε χρήστης δεν θα πρέπει απαραίτητα να εκτελέσει κάποια λειτουργία. Ο κατάλογος μιας εταιρείας που εξυπηρετεί το δικό της Intranet μπορεί να αποθηκευθεί πίσω από ένα firewall ώστε να κρατά μακριά το ευρύ κοινό από την πρόσβασή του σε αυτόν, όμως απαιτείται περισσότερος έλεγχος ασφάλειας μέσα στο ίδιο το Intranet. Για παράδειγμα, καθένας θα ήταν σε θέση να αναζητήσει το e-mail ενός υπαλλήλου αλλά μόνο ο υπάλληλος ή ο διαχειριστής συστήματος θα μπορούσε να το αλλάξει. Ίσως τα μέλη του τμήματος προσωπικού να έχουν την άδεια να αναζητήσουν τον τηλεφωνικό αριθμό ενός υπαλλήλου αλλά οι συνεργάτες τους να μην έχουν την άδεια αυτή. Μπορεί η πληροφορία να χρειασθεί να κρυπτογραφηθεί πριν διατεθεί σε όλο το δίκτυο. Μια πολιτική ασφάλειας καθορίζει ποιος και τί είδους πρόσβαση έχει και σε ποια πληροφορία. Η πολιτική ασφάλειας είναι καθορισμένη από τον οργανισμό ο οποίος διατηρεί τον κατάλογο. Ένας κατάλογος θα πρέπει να υποστηρίζει τις βασικές προϋποθέσεις που χρειάζονται για να εφαρμοσθεί μια πολιτική ασφάλειας. Ο κατάλογος μπορεί να μην είναι σε θέση να παρέχει άμεσα τις συγκεκριμένες δυνατότητες ασφάλειας, αλλά μπορεί να συνεργασθεί με μια έμπιστη υπηρεσία ασφάλειας δικτύου η οποία θα παρέχει τις βασικές υπηρεσίες ασφάλειας. Πρώτον, χρειάζεται μια μέθοδος επικύρωσης των χρηστών. Η επικύρωση επιβεβαιώνει ότι οι χρήστες είναι πράγματι αυτοί που δηλώνουν ότι είναι. Ένα όνομα χρήστη (user name) και ένας κωδικός (password) αποτελούν ένα βασικό σχέδιο επικύρωσης. Μόλις οι χρήστες επικυρώνονται πρέπει να διευκρινισθεί αν έχουν την έγκριση ή την άδεια να εκτελέσουν την αιτούμενη ενέργεια πάνω στο συγκεκριμένο αντικείμενο.
Η επικύρωση συχνά βασίζεται στους καταλόγους ελέγχου πρόσβασης (ACLs). Ένα ACL είναι μια λίστα των εγκρίσεων που ίσως συνδέονται με τα αντικείμενα και τις ιδιότητες στον κατάλογο. Ένα ACL απαριθμεί τον τύπο πρόσβασης που επιτρέπεται σε κάθε χρήστη. Προκειμένου να γίνουν οι ACLs μικρότεροι και πιο εύχρηστοι, οι χρήστες που έχουν τα ίδια δικαιώματα πρόσβασης συχνά τοποθετούνται σε ομάδες ασφάλειας. Ο πίνακας 2.2.4 δείχνει ένα παράδειγμα ACL για μια υπαλληλική καταχώρηση καταλόγου.

ΧΡΗΣΤΗΣ ή ΟΜΑΔΑ	ΔΙΚΑΙΩΜΑΤΑ ΠΡΟΣΒΑΣΗΣ
Δικαιούχος	Ανάγνωση, μετατροπή (όχι όμως διαγραφή)
Διαχειριστές	Όλα
Προσωπικό	Ανάγνωση όλων των πεδίων
Οι υπόλοιποι	Περιορισμένη ανάγνωση

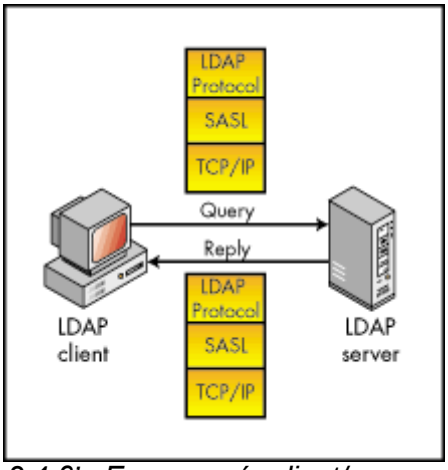
Πίνακας 2.2.4 Παράδειγμα ACL για μια υπαλληλική καταχώρηση καταλόγου

Όπως περιγράφηκε παραπάνω, το μοντέλο ασφάλειας είναι βασισμένο στην λειτουργία **bind**. Υπάρχουν πολλές διαφορετικές λειτουργίες bind και επομένως ο μηχανισμός ασφάλειας που εφαρμόζεται είναι επίσης διαφορετικός. Μια περίπτωση είναι όταν ο client ζητεί πρόσβαση δίνοντας το δικό του αναγνωριστικό DN και έναν κωδικό. Εάν δεν δηλωθούν ο κωδικός και το DN τότε θεωρείται ως ανώνυμη σύνοδος από την πλευρά του LDAP server.



Σχήμα 2.2.4.α' Επικοινωνία client/server χωρίς SASL

Η χρήση των κωδικών αποθαρρύνεται δυναμικά όταν η βαθύτερη υπηρεσία μεταφοράς δεν μπορεί να εγγυηθεί εμπιστευτικότητα και επομένως τα αποτελέσματα των αναζητήσεων κοινοποιούνται σε μη επικυρωμένα μέλη. Επιπροσθέτως, είναι πιθανή η εντολή bind του Kerberos στην έκδοση 2 του LDAP, αλλά αυτό αποδοκιμάζεται στην έκδοση 3. Αντί για αυτό, το LDAPv3 συνοδεύεται με την εντολή bind του SASL (Simple Authentication and Security Layer) μηχανισμού.



Σχήμα 2.2.4.β' Επικοινωνία client/server με SASL

Αυτό είναι ένα γενικό πλαίσιο επικύρωσης, όπου πολλές μέθοδοι επικύρωσης είναι διαθέσιμες για την επικύρωση του client στον server. Μια από αυτές είναι και το Kerberos.

Επιπλέον, οι εκτενείς λειτουργίες πρωτοκόλλου διατίθενται στο LDAPv3. Μια επέκταση σχετική με την ασφάλεια είναι η ‘’Επέκταση για την Ασφάλεια του Επιπέδου Μεταφοράς’’ (TLS) για το LDAPv3. Αυτή καθορίζει λειτουργίες που χρησιμοποιούν την TLS ως μέσο για την κρυπτογράφηση της συνόδου LDAP και προστατεύοντάς την από την παραποίηση δεδομένων. Βασίζεται στο Secure Socket Layer (SSL) Protocol 3.0, το οποίο έχει επινοηθεί από την Netscape Communications Corporation. Η TLS έχει έναν μηχανισμό που της επιτρέπει να επικοινωνεί με έναν SSL server. Μερικοί προμηθευτές όπως ο Netscape και η IBM έχουν ήδη επεκτείνει το πρωτόκολλο LDAP και έχουν προσθέσει κάποιες συγκεκριμένες SSL εντολές έτσι ώστε να είναι δυνατή η κρυπτογράφηση μιας TCP/IP σύνδεσης και κατά συνέπεια παρέχονται μέσα που περιορίζουν την ανάγκη αποστολής απροστάτευτων DN και κωδικών πρόσβασης σε όλο το δίκτυο. Από την στιγμή που ένας client θα αναγνωριστεί, οι πληροφορίες ελέγχου πρόσβασης θα ενημερώσουν για το αν ο client έχει επαρκείς άδειες προσπέλασης για να πραγματοποιήσει αυτό που ζητεί.

CONCLUSION

Το LDAP (Lightweight Directory Access Protocol) είναι ένα προτεινόμενο ανοιχτό πρότυπο για τοπικές ή παγκόσμιες υπηρεσίες καταλόγου που βρίσκονται σε ένα τοπικό δίκτυο ή στο Internet. Με αυτήν την έννοια ένας κατάλογος μοιάζει κατά πολύ με έναν τηλεφωνικό κατάλογο. Το LDAP μπορεί να χειρισθεί πολλών ειδών πληροφορίες αλλά προς το παρόν χρησιμοποιείται για τον συσχετισμό ονομάτων με τηλεφωνικούς αριθμούς και διευθύνσεις ηλεκτρονικού ταχυδρομείου. Οι κατάλογοι είναι σχεδιασμένοι να υποστηρίζουν μεγάλο όγκο ερωτημάτων, αλλά τα δεδομένα που είναι αποθηκευμένα στους καταλόγους δεν αλλάζουν συχνά.

Το LDAP είναι περισσότερο χρήσιμο σε αντίθεση με έναν τηλεφωνικό κατάλογο επειδή ο σχεδιασμός του είναι προορισμένος να υποστηρίζει την μετάδοση διαμέσου των LDAP servers σε όλο το Internet, όπως γίνεται περίπου με το DNS (Domain Name Service). Το σύστημα DNS συμπεριφέρεται σαν το βιβλίο διευθύνσεων του Internet με την παρακολούθηση των domain names και των IP διευθύνσεων. Στο μέλλον, το LDAP θα παρέχει τον ίδιο τύπο πρόσβασης για πολλά είδη πληροφοριών καταλόγου. Προς το παρόν, το LDAP χρησιμοποιείται για μικρούς οργανισμούς π.χ για ένα πανεπιστήμιο ή σε μεγάλες εταιρείες, για λογαριασμό των υπηρεσιών καταλόγου.

Το LDAP είναι ένα σύστημα client-server. Ένας LDAP client συνδέεται με έναν LDAP server και είτε ζητάει πληροφορίες είτε παρέχει πληροφορία που χρειάζεται να ενσωματωθεί στον server. Με την σειρά του ο server απαντάει στο ερώτημα παραπέμποντας το ερώτημα σε άλλον LDAP server, ή δέχεται τις πληροφορίες προς ενσωμάτωση στον κατάλογο.

Μερικές φορές το LDAP αναφέρεται ως X.500 Lite. Το X.500 είναι ένα διεθνές πρότυπο για καταλόγους. Διαθέτει πλήρη χαρακτηριστικά γνωρίσματα, αλλά είναι πολύπλοκο και απαιτεί πολλούς πόρους και ολόκληρη την στοίβα OSI. Από την άλλη μεριά, το LDAP μπορεί να τρέχει σε ένα PC εύκολα και χρησιμοποιεί το TCP/IP. Το LDAP προσπελάζει τους καταλόγους X.500 αλλά δεν υποστηρίζει κάθε δυνατότητα του X.500.

Το κύριο πλεονέκτημα του LDAP είναι η ενοποίηση ορισμένων τύπων πληροφορίας σε έναν οργανισμό. Για παράδειγμα, όλοι οι χρήστες μιας επιχείρησης συγχωνεύονται σε έναν κατάλογο LDAP. Ο κατάλογος αυτός μπορεί να ερωτηθεί από οποιαδήποτε εφαρμογή LDAP που χρειάζεται πληροφορίες καταλόγου.

Άλλα πλεονεκτήματα του LDAP περιλαμβάνουν την ευκολία εφαρμογής του (σε σύγκριση με το X.500) και το –καλά ορισμένο- API (Application Programming Interface), που σημαίνει ότι θα αυξηθεί στο μέλλον ο αριθμός των εφαρμογών που σχετίζονται με το LDAP καθώς και οι LDAP gateways.

Μειονέκτημα είναι το γεγονός ότι η χρήση του LDAP απαιτεί εφαρμογές που υποστηρίζουν το πρωτόκολλο. Επιπλέον το LDAP δεν υποστηρίζει μεγάλο αριθμό εσωτερικών γνωρισμάτων ασφάλειας όπως συμβαίνει με το X.500.

Το LDAP συνήθως χρησιμοποιείται ως ο κεντρικός εξυπηρετητής επικύρωσης έτσι ώστε οι χρήστες να έχουν μια ενοποιημένη σύνδεση που καλύπτει τους POP servers, IMAP servers, τα μηχανήματα που συνδέονται στο δίκτυο χρησιμοποιώντας το Samba και ακόμα τα Windows NT. Όλες

αυτές οι καταστάσεις σύνδεσης μπορούν να βασίζονται στον ίδιο κωδικό χρήστη χρησιμοποιώντας το LDAP.

Glossary

A

ACL (Access Control List): Κατάλογος Ελέγχου Προσπέλασης. Κατάλογος χρηστών ή προγραμμάτων που περιέχει και τις αντίστοιχες κατηγορίες εξουσιοδότησης για την προσπέλαση στο σύστημα ή σε συγκεκριμένους πόρους του.
alias: Ένα όνομα, συνήθως μικρό και εύκολο στην απομνημόνευση, το οποίο μεταφράζεται σε ένα άλλο όνομα ή στοιχειοσειρά, συνήθως μεγάλου μήκους με πολύπλοκο τύπο.
API (Application Programming Interface) : Είναι η διεπαφή με την οποία ένα πρόγραμμα εφαρμογής προσπελαύνει ένα λειτουργικό σύστημα και άλλες υπηρεσίες. Ένα API καθορίζεται στο επίπεδο πηγαίου κώδικα και παρέχει ένα επίπεδο αφαίρεσης ανάμεσα στην εφαρμογή και στον πυρήνα για να εξασφαλισθεί η μεταφορά του κώδικα.
ASCII (American Standard Code for Information Interchange): Κώδικας ASCII. Κώδικας αναπαράστασης πληροφοριών που χρησιμοποιεί 7 δυαδικά ψηφία για την αναπαράσταση 128 αλφαβητικών και αριθμητικών χαρακτήρων, ειδικών συμβόλων και χαρακτήρων ελέγχου.
ASN.1 (Abstract Syntax Notation.1): Ένα πρότυπο του ISO/ITU-T για την μετάδοση δεδομένων στα δίκτυα. Αρχικά καθορίσθηκε το 1984 ως μέρος του CCITT X.409 ’84. Η ASN.1 καθορίζει έναν αφηρημένο συντακτικό τύπο των πληροφοριών που όμως δεν περιορίζει τον τρόπο με τον οποίο οι πληροφορίες κωδικοποιούνται.
attribute (χαρακτηριστικό) : Οτιδήποτε χαρακτηρίζει ή προσδιορίζει μια οντότητα.

B

BOOLEAN: Άλγεβρα Boole. Αλγεβρικό σύστημα που διατυπώθηκε από τον Άγγλο μαθηματικό George Boole το 1847 και έπαιξε καθοριστικό ρόλο στην εξέλιξη της επιστήμης των υπολογιστών. Οι μεταβλητές της άλγεβρας Boole μπορούν να πάρουν δύο μόνον τιμές – ‘’αληθές’’ και ‘’ψευδές’’- και να συνδυαστούν μέσω των τελεστών ‘’ΚΑΙ’’ (AND) και ‘’ Η ‘’ (OR) ή να αν αναιρεθούν λογικά με τον τελεστή ‘’ΟΧΙ’’ (NOT), για να δώσουν ένα λογικό αποτέλεσμα η τιμή του οποίου (‘’αληθές’’ ή ‘’ψευδές’’) καθορίζεται από μια σειρά κανόνων.

C

C: Γλώσσα προγραμματισμού C. Υψηλού επιπέδου γλώσσα προγραμματισμού που αναπτύχθηκε το 1972 από τον Dennis Ritchie στα εργαστήρια της Bell (AT&T), αρχικά για την υλοποίηση του λειτουργικού συστήματος UNIX. Είναι μια από τις πιο διαδεδομένες σύγχρονες γλώσσες προγραμματισμού. Συνδυάζει τις δομές ελέγχου και δεδομένων και τις ευκολίες χρήσης των μοντέρνων, υψηλού επιπέδου γλωσσών προγραμματισμού, με τις δυνατότητες των χαμηλού επιπέδου συμβολικών γλωσσών. Τα χαρακτηριστικά αυτά την καθιστούν εξαιρετικά αποδοτική για τον προγραμματισμό συστημάτων.

C++: Γλώσσα προγραμματισμού C++. Υψηλού επιπέδου γλώσσα προγραμματισμού που αναπτύχθηκε από την AT&T και χαρακτηρίζεται ως υπερσύνολο της γλώσσας C. Συνδυάζει τα πλεονεκτήματα της γλώσσας C με βασικά στοιχεία των γλωσσών LISP και Algol 68. Χαρακτηρίζεται επίσης ως ‘’γλώσσα προσανατολισμένη στο αντικείμενο’’. Από τα πιο δυναμικά χαρακτηριστικά της είναι η ευκολία στη χρήση αφηρημένων τύπων δεδομένων, που έχει ως αποτέλεσμα την αποδοτική χρήση της στην ανάπτυξη δομημένων προγραμμάτων.
class (κατηγορία): Μια κατηγορία, η οποία συχνά συγχέεται με το αντικείμενο, καθορίζει μια οντότητα εφαρμογής που προγραμματίζεται με βάση την δομή των δεδομένων και των ενεργειών που μπορούν να γίνουν πάνω στα δεδομένα.

D

DARPA (Defense Advanced Research Projects Agency): Αντιπροσωπεία της αμερικανικής κυβέρνησης που παρείχε την αρχική υποστήριξη για την ανάπτυξη των διασυνδεδεμένων δικτύων που αργότερα οδήγησε στην δημιουργία του Internet.
de facto πρότυπο: Ένα σχέδιο, πρόγραμμα ή γλώσσα, που έχει γίνει ευρέως αποδεκτό αλλά ακόμα δεν έχει αναγνωρισθεί ως πρότυπο από τον ANSI (American National Standards Institute) ή από τον OSI (International Organizational for Standardization).
DIT (Directory Information Tree): Δένδρο Πληροφοριών Καταλόγου. Μια γραφική απεικόνιση των καταλόγων και υποκαταλόγων σε δενδρική μορφή, με τους υποκαταλόγους να εμφανίζονται ως «κλαδιά» του κυρίως καταλόγου.
DNS (Domain Name System) : Το σύστημα το οποίο χρησιμοποιείται από τα μηχανήματα ενός δικτύου για την σύνδεση των IP διευθύνσεων (όπως η 195.130.72.52) με τα hostnames (όπως το www.teiep.gr). Οι υπολογιστές λαμβάνουν την IP διεύθυνση για ένα hostname από τον DNS server, ή ψάχνουν στους πίνακες που διατηρούνται στα συστηματά τους.
DSA (Directory System Agent): Το λογισμικό που παρέχει την υπηρεσία καταλόγου X.500 για ένα τμήμα της πληροφοριακής βάσης καταλόγου. Γενικά, ένας DSA είναι υπεύθυνος για τις πληροφορίες καταλόγου ενός οργανισμού.

E

EDI (Electronic Data Interchange): Ηλεκτρονική Ανταλλαγή Δεδομένων. Μετάδοση δομημένης (προκαθορισμένης/ τυποποιημένης) πληροφορίας, με ηλεκτρονικό τρόπο, ανάμεσα σε συστήματα υπολογιστών που ανήκουν σε διαφορετικές εταιρίες ή οργανισμούς. Ο όρος δεν περιλαμβάνει το ηλεκτρονικό ταχυδρομείο ή την τηλεομοιοτυπία, όπου η πληροφορία δεν είναι δομημένη. Ο ηλεκτρονικός τρόπος μετάδοσης συμπεριλαμβάνει την απευθείας σύνδεση ή την ανταλλαγή με τη βοήθεια π.χ μαγνητικών ταινιών.
EDUCAUSE: Η EDUCAUSE είναι μια μη κερδοσκοπική οργάνωση η οποία έχει ως σκοπό την προαγωγή της τριτοβάθμιας εκπαίδευσης με την προώθηση χρήσης ευφυούς τεχνολογίας πληροφοριών.
Entry (καταχώρηση): Κάθε στοιχείο μιας λίστας, ενός πίνακα κ.λ.π. Στην περίπτωση των Βάσεων Δεδομένων ο όρος είναι γνωστός ως εγγραφή.

F

FAX (facsimile): Τηλεομοιοτυπία. Η μετάδοση εγγράφων, σχεδίων, φωτογραφιών κ.λ.π. μέσω τηλεπικοινωνιακών δικτύων. Η πρώτη προσπάθεια αποδίδεται στο Σκοτσέζο Alexander Bayn, ο οποίος ανέπτυξε γύρω στο 1842 μια τεχνική για τη μετάδοση εικόνων και εντύπων μέσω τηλεγραφικών γραμμών. Η εξέλιξη της τηλεομοιοτυπίας υπήρξε ραγδαία και σήμερα τείνει να αντικαταστήσει τις υπηρεσίες τέλεξ καθώς και τις ειδικές ταχυδρομικές υπηρεσίες (επείγουσες αποστολές).
filter (φίλτρο): Δυαδική παράσταση που χρησιμοποιείται για την τροποποίηση ή την αναγνώριση ενός τμήματος μιας άλλης δυαδικής παράστασης. Η αναγνώριση ή τροποποίηση γίνεται με την εκτέλεση της κατάλληλης λογικής πράξης (AND, OR, XOR) ανάμεσα στις δύο παραστάσεις.
firewall: 1. Μηχάνημα το οποίο διαθέτει ειδικές προφυλάξεις για τις συνδέσεις εκτός τοπικού δικτύου π.χ στο Internet. Η ιδέα βρίσκεται στο να προστατευθούν οι υπολογιστές που βρίσκονται πίσω του από τους crackers. Ένα τυπικό firewall είναι ένας προσιτός μικροεπεξεργαστής που βασίζεται σε ένα μηχάνημα UNIX. 2. Ο κώδικας που τοποθετούμε σε ένα σύστημα για να εξασφαλισθεί η αποφυγή βλάβης από τους χρήστες.
format (μορφή, τροποποίηση): Η διάταξη των πεδίων μιας εγγραφής ή των δεδομένων σε ένα έντυπο, οθόνη ή μέσο αποθήκευσης – ορισμός της μορφής αποθήκευσης των πληροφοριών σ' ένα φορέα δεδομένων, π.χ σ' ένα μαγνητικό δίσκο ή σε μια συσκευή εξόδου.

G

gateway (δικτυακή πύλη): Ο κοινός κόμβος δύο ή περισσότερων δικτύων υπολογιστών. Η λειτουργία της πύλης δικτύων είναι ανάλογη με εκείνη ενός διερμηνέα που μεταφράζει π.χ από τα αγγλικά στα κινέζικα (ή και σε περισσότερες ταυτόχρονα γλώσσες) και αντίστροφα, προσαρμόζει δηλαδή τα πακέτα μηνυμάτων στα διαφορετικά πρωτόκολλα που χρησιμοποιούν τα συνδεδεμένα δίκτυα. Η λειτουργία της πύλης είναι άμεσα αντιληπτή από τους χρήστες των δικτύων.

H

host name(ή ''site name'') : Το μοναδικό όνομα με το οποίο ένας υπολογιστής γνωστοποιείται σε ένα δίκτυο και χρησιμεύει στην αναγνώριση του κατά την διάρκεια οποιασδήποτε ηλεκτρονικής ανταλλαγής πληροφοριών.
HTTP(Hypertext Transfer Protocol) : Πρωτόκολλο που βασίζεται στο μοντέλο client/server και χρησιμοποιείται για την ανταλλαγή κειμένων HTML στο Internet. Χρησιμοποιεί το port 80.

I

IP address (Internet Protocol address): Ένας 32- bit (4- byte) δυαδικός αριθμός που προσδιορίζει μοναδικά έναν host ο οποίος είναι συνδεδεμένος στο Internet με άλλους Internet hosts με σκοπό την επικοινωνία μέσω της μετάδοσης πακέτων.

J

JAVA: Δημοφιλής αντικειμενοστραφής γλώσσα προγραμματισμού που αναπτύχθηκε στις αρχές του 1990 από την Sun Microsystems και κυκλοφόρησε προς εφαρμογή το 1995.

K

Kerberos: Σύστημα επικύρωσης που αναπτύχθηκε στο MIT (Massachusetts Institute of Technology). Το Kerberos σχεδιάσθηκε για να κάνει δυνατή την ανταλλαγή ιδιωτικής πληροφορίας μεταξύ δυο μερών σε ένα διαφορετικό ανοιχτό δίκτυο. Λειτουργεί με την ανάθεση ενός μοναδικού κλειδιού, το οποίο ονομάζεται ticket, σε κάθε χρήστη που συνδέεται στο δίκτυο. Έπειτα το ticket ενσωματώνεται σε μηνύματα για να αναγνωρισθεί ο αποστολέας του μηνύματος.

M

modules (ενότητες): Κάθε αυτοτελές τμήμα ενός συστήματος ή προγράμματος που λειτουργεί ανεξάρτητα ή/και συνεργάζεται με άλλα τμήματα.

N

NOS (Network Operating System): Το πρόγραμμα λογισμικού που παρέχει την διεπαφή χρήστη στο LAN και ελέγχει την λειτουργία του δικτύου. Ένα NOS επικοινωνεί με το υλικό του LAN και δίνει την δυνατότητα στους χρήστες να επικοινωνούν μεταξύ τους και να μοιράζονται αρχεία και περιφερειακά.

P

PDU (Protocol Data Unit): Ένα πακέτο δεδομένων που βγαίνει εκτός δικτύου. Ο όρος αφορά ένα συγκεκριμένο στρώμα του OSI και ένα συγκεκριμένο πρωτόκολλο.
port number: Ένα από τα input/output κανάλια δικτύου ενός Η/Υ ο οποίος τρέχει το TCP/IP. Στο World Wide Web, με τον όρο port εννοούμε συνήθως τον αριθμό στον οποίο τρέχει ένας server. Σε έναν Η/Υ μπορεί να τρέχουν πολλοί Web servers, αλλά μόνο ένας server μπορεί να τρέχει σε κάθε port. Ο προεπιλεγμένος port number για τους Web servers είναι 80.
POSTSCRIPT (γλώσσα προγραμματισμού POSTSCRIPT): Γλώσσα προγραμματισμού που χρησιμοποιείται στα επιτραπέζια εκδοτικά συστήματα και τους εκτυπωτές λέιζερ.
Proxy(server): Ένα στοιχείο firewall που διαχειρίζεται την κυκλοφορία Internet από και προς ένα τοπικό δίκτυο όπου παρέχει διάφορα features όπως εναποθήκευση εγγράφων και έλεγχο πρόσβασης. Ένας proxy server μπορεί να βελτιώσει την απόδοση με την παροχή των συχνά ζητούμενων δεδομένων, όπως μια Web σελίδα, και μπορεί να φιλτράρει και να απορρίπτει αιτήματα τα οποία ο δικαιούχος δεν θεωρεί κατάλληλα όπως μια ανεπικύρωτη πρόσβαση σε κάποια ιδιόκτητα αρχεία.

R

RFC (Request For Comment): Τα πρότυπα του δικτύου Internet ονομάζονται RFCs. Ένα προτεινόμενο Internet standard αρχικά θέτεται ως πρόταση και του δίνεται ένας

αριθμός RFC. Κατόπιν, όταν γίνεται αποδεκτό, προστίθεται στα Official Internet Protocols, αλλά ακόμα αναφέρεται σύμφωνα με τον RFC αριθμό.
RPC (Remote Procedure Call): Πρόκειται για την βάση του κατανεμημένου υπολογιστικού συστήματος. Η Απομακρυσμένη Διαδικασία Κλήσης αποφασίζεται από τους clients και εκτελείται στους servers, με τα αποτελέσματα να επιστρέφουν μέσω του δικτύου στους clients.

S

SASL (Simple Authentication and Security Layer): Μέθοδος που προσθέτει υποστήριξη επικύρωσης στα πρωτόκολλα διασύνδεσης. Για την χρησιμοποίηση του SASL, ένα πρωτόκολλο συμπεριλαμβάνει μια εντολή αναγνώρισης και επικύρωσης για τον χρήστη που πρόκειται να συνδεθεί με έναν server.
Schema (σχήμα): Η περιγραφή της δομής μιας Βάσης Δεδομένων.
SQL (Structured Query Language): Η SQL είναι μια σχεσιακή γλώσσα δεδομένων που παρέχει ένα σύνολο ευκολιών για ερωτήματα, ορισμούς δεδομένων καθώς και για τον χειρισμό και έλεγχο δεδομένων. Αποτελεί το API για τα RDBMS.
SSL (Secure Sockets Layer): Πρωτόκολλο το οποίο σχεδιάσθηκε από την Netscape Communications Corporation ώστε να παρέχει κρυπτογραφημένη επικοινωνία στο Διαδίκτυο. Το SSL βρίσκεται κάτω από τα πρωτόκολλα εφαρμογής όπως το HTTP, SMTP, Telnet, FTP και ένα στρώμα πάνω από το πρωτόκολλο διασύνδεσης TCP/IP. Χρησιμοποιείται με την μέθοδο πρόσβασης HTTPS.

T

TLS (Transport Layer Security protocol): Ένα πρωτόκολλο σχεδιασμένο ώστε να επιτρέπει την επικοινωνία των εφαρμογών client/server διαμέσου του Internet χωρίς να κινδυνεύουν από ανεπικύρωτες παρεμβάσεις και παραβιάσεις μηνυμάτων.

U

UNIX (λειτουργικό σύστημα UNIX): Λειτουργικό σύστημα που σχεδίασε και υλοποίησε το 1996 Ken Thomson στα εργαστήρια της BELL (AT&T). Θεωρείται από τα πλέον δημοφιλή και ‘’φιλικά στο χρήστη’’ λειτουργικά συστήματα και υιοθετήθηκε από πολλούς κατασκευαστές μεσαίων και μεγάλων υπολογιστών ως σύστημα πολυεπεξεργασίας και διαμερισμού χρόνου, με δυνατότητα υποστήριξης ενός σχετικά μικρού αριθμού χρηστών.
URL (Uniform Resource Locator): Τα URL χρησιμοποιούνται από τους Web Browsers για να εντοπίσουν πόρους στο Internet. Ένα URL προσδιορίζει το πρωτόκολλο που θα χρησιμοποιηθεί για την αναζήτηση (όπως το HTTP για μια World Wide Web σελίδα ή το FTP για ένα FTP site), το όνομα του server στον οποίο βρίσκεται ο πόρος (όπως ο //www.ldapman.org) και προαιρετικά, το path της πηγής (π.χ ένα έγγραφο HTML ή ένα αρχείο).

Πηγές

1. http://www.openldap.org/
2. http://www.openldap.org/lists/openldap-announce/200506/msg00002.html
3. http://www.openldap.org/project/
4. http://www.citi.umich.edu/techreports/reports/
5. http://www.umich.edu/~urecord/9596/Apr30_96/artcl04.html
6. http://www.umich.edu/%7Edirsvcs/ldap/doc/guides/slapd/1.html#RTFToC1
7. http://www.isode.com/products/cldap-server.html
8. http://www-306.ibm.com/software/tivoli/products/directory-server/
9. http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/ldap.asp
10. http://www.novell.com/documentation/edir873/index.html
11. http://www.novell.com/products/edirectory/features.html
12. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/diren.htm
13. http://www.ietf.org/html.charters/ldapis-charter.html
14. http://www.europe.redhat.com/documentation/rh16.2/ref-guide-en/ch-ldap.php3
15. http://www.sun.com/software/products/directory
16. http://www.uic.edu/depts/accc/inform/v106o.html
17. http://www.pcwebopedia.com/TERM/K/Kerberos.html
18. http://www.kingsmountain.com/ldapRoadmap.shtml
19 .http://www.zytrax.com/books/ldap/ch3/
20. http://searchdomino.techtarget.com/search/1,293876,sid4,00.html?query=ldap+protocol
21. http://support.microsoft.com/default.aspx?scid=%2Fsupport%2Fglossary%2Fd.asp
22. http://docs.sun.com/source/816-6733-10/index.html
23. http://support.sas.com/rnd/itech/doc/admin/ldap/ldinst.html
24. http://developer.novell.com
25. http://athina.noc.sch.gr:6080/sch-portlets/static/manual/catalog/index.php?
26. http://vnoc.grnet.gr/content/downloads/GRNET%20LDAP%20schema.pdf
27. http://quark.humbug.org.au/publications/ldap/ldap-apps-v2.pdf
28. “The Lightweight Directory Access Protocol: X.500 Lite” Timothy A. Howes CITI Technical Report 95-8 July 1995
29. “An X.500 and LDAP Database: Design and Implementation” Timothy A. Howes
30. “Directory Server Security” by Tom Bialaski- Enterprise Engineering SunBluePrints Online- December 2000
31. “Understanding LDAP” Heinz Johner, Larry Brown, Franz- Stefan Hinner, Wolfgang Reis, Johan Westman IBM, JUNE 1998
32. ‘’LDAP Servers and Applications’’ Brad Marshall
33. “Σύγχρονο Λεξικό Πληροφορικής” Π. Κ. ΓΑΡΙΔΗΣ, ΕΜ. Ν. ΔΕΛΗΓΙΑΝΝΑΚΗΣ ΔΙΑΥΛΟΣ 1989

ΠΑΡΑΡΤΗΜΑ

RFC 2251

Network Working Group M. Wahl
Request for Comments: 2251 Critical Angle Inc.
Category: Standards Track T. Howes
Netscape Communications Corp.
S. Kille
Isode Limited
December 1997

Lightweight Directory Access Protocol (v3)

1. Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

IESG Note

This document describes a directory access protocol that provides both read and update access. Update access requires secure authentication, but this document does not mandate implementation of any satisfactory authentication mechanisms.

In accordance with RFC 2026, section 4.4.1, this specification is being approved by IESG as a Proposed Standard despite this limitation, for the following reasons:

a. to encourage implementation and interoperability testing of these protocols (with or without update access) before they are deployed, and

b. to encourage deployment and use of these protocols in read-only applications. (e.g. applications where LDAPv3 is used as a query language for directories which are updated by some secure mechanism other than LDAP), and

c. to avoid delaying the advancement and deployment of other Internet standards-track protocols which require the ability to query, but not update, LDAPv3 directory servers.

Readers are hereby warned that until mandatory authentication mechanisms are standardized, clients and servers written according to this specification which make use of update functionality are UNLIKELY TO INTEROPERATE, or MAY INTEROPERATE ONLY IF AUTHENTICATION IS REDUCED TO AN UNACCEPTABLY WEAK LEVEL.

Implementors are hereby discouraged from deploying LDAPv3 clients or servers which implement the update functionality, until a Proposed Standard for mandatory authentication in LDAPv3 has been approved and published as an RFC.

Table of Contents

2. Abstract

The protocol described in this document is designed to provide access to directories supporting the X.500 models, while not incurring the resource requirements of the X.500 Directory Access Protocol (DAP). This protocol is specifically targeted at management applications and browser applications that provide read/write interactive access to directories. When used with a directory supporting the X.500 protocols, it is intended to be a complement to the X.500 DAP.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [10].

Key aspects of this version of LDAP are:

- All protocol elements of LDAPv2 (RFC 1777) are supported. The protocol is carried directly over TCP or other transport, bypassing much of the session/presentation overhead of X.500 DAP.

- Most protocol data elements can be encoded as ordinary strings (e.g., Distinguished Names).

- Referrals to other servers may be returned.

- SASL mechanisms may be used with LDAP to provide association security services.

- Attribute values and Distinguished Names have been internationalized through the use of the ISO 10646 character set.

- The protocol can be extended to support new operations, and controls may be used to extend existing operations.

- Schema is published in the directory for use by clients.

3. Models

Interest in X.500 [1] directory technologies in the Internet has led to efforts to reduce the high cost of entry associated with use of

these technologies. This document continues the efforts to define directory protocol alternatives, updating the LDAP [2] protocol specification.

3.1. Protocol Model

The general model adopted by this protocol is one of clients performing protocol operations against servers. In this model, a client transmits a protocol request describing the operation to be performed to a server. The server is then responsible for performing the necessary operation(s) in the directory. Upon completion of the operation(s), the server returns a response containing any results or errors to the requesting client.

In keeping with the goal of easing the costs associated with use of the directory, it is an objective of this protocol to minimize the complexity of clients so as to facilitate widespread deployment of applications capable of using the directory.

Note that although servers are required to return responses whenever such responses are defined in the protocol, there is no requirement for synchronous behavior on the part of either clients or servers. Requests and responses for multiple operations may be exchanged between a client and server in any order, provided the client eventually receives a response for every request that requires one.

In LDAP versions 1 and 2, no provision was made for protocol servers returning referrals to clients. However, for improved performance and distribution this version of the protocol permits servers to return to clients referrals to other servers. This allows servers to offload the work of contacting other servers to progress operations.

Note that the core protocol operations defined in this document can be mapped to a strict subset of the X.500(1997) directory abstract service, so it can be cleanly provided by the DAP. However there is not a one-to-one mapping between LDAP protocol operations and DAP operations: server implementations acting as a gateway to X.500 directories may need to make multiple DAP requests.

3.2. Data Model

This section provides a brief introduction to the X.500 data model, as used by LDAP.

The LDAP protocol assumes there are one or more servers which jointly provide access to a Directory Information Tree (DIT). The tree is made up of entries. Entries have names: one or more attribute values from the entry form its relative distinguished name (RDN), which MUST be unique among all its siblings. The concatenation of the relative distinguished names of the sequence of entries from a particular entry to an immediate subordinate of the root of the tree forms that entry's Distinguished Name (DN), which is unique in the tree. An example of a Distinguished Name is

CN=Steve Kille, O=Isode Limited, C=GB

Some servers may hold cache or shadow copies of entries, which can be used to answer search and comparison queries, but will return referrals or contact other servers if modification operations are requested.

Servers which perform caching or shadowing MUST ensure that they do not violate any access control constraints placed on the data by the originating server.

The largest collection of entries, starting at an entry that is mastered by a particular server, and including all its subordinates and their subordinates, down to the entries which are mastered by different servers, is termed a naming context. The root of the DIT is a DSA-specific Entry (DSE) and not part of any naming context: each server has different attribute values in the root DSE. (DSA is an X.500 term for the directory server).

3.2.1. Attributes of Entries

Entries consist of a set of attributes. An attribute is a type with one or more associated values. The attribute type is identified by a short descriptive name and an OID (object identifier). The attribute

type governs whether there can be more than one value of an attribute of that type in an entry, the syntax to which the values must conform, the kinds of matching which can be performed on values of that attribute, and other functions.

An example of an attribute is "mail". There may be one or more values of this attribute, they must be IA5 (ASCII) strings, and they are case insensitive (e.g. "foo@bar.com" will match "FOO@BAR.COM").

Schema is the collection of attribute type definitions, object class definitions and other information which a server uses to determine how to match a filter or attribute value assertion (in a compare operation) against the attributes of an entry, and whether to permit add and modify operations. The definition of schema for use with LDAP is given in [5] and [6]. Additional schema elements may be defined in other documents.

Each entry MUST have an objectClass attribute. The objectClass attribute specifies the object classes of an entry, which along with the system and user schema determine the permitted attributes of an entry. Values of this attribute may be modified by clients, but the objectClass attribute cannot be removed. Servers may restrict the modifications of this attribute to prevent the basic structural class of the entry from being changed (e.g. one cannot change a person into a country). When creating an entry or adding an objectClass value to an entry, all superclasses of the named classes are implicitly added as well if not already present, and the client must supply values for any mandatory attributes of new superclasses.

Some attributes, termed operational attributes, are used by servers for administering the directory system itself. They are not returned in search results unless explicitly requested by name. Attributes which are not operational, such as "mail", will have their schema and syntax constraints enforced by servers, but servers will generally not make use of their values.

Servers MUST NOT permit clients to add attributes to an entry unless those attributes are permitted by the object class definitions, the schema controlling that entry (specified in the subschema - see below), or are operational attributes known to that server and used for administrative purposes. Note that there is a particular

objectClass 'extensibleObject' defined in [5] which permits all user attributes to be present in an entry.

Entries MAY contain, among others, the following operational attributes, defined in [5]. These attributes are maintained automatically by the server and are not modifiable by clients:

- creatorsName: the Distinguished Name of the user who added this entry to the directory.

- createTimestamp: the time this entry was added to the directory.

- modifiersName: the Distinguished Name of the user who last modified this entry.

- modifyTimestamp: the time this entry was last modified.

- subschemaSubentry: the Distinguished Name of the subschema entry (or subentry) which controls the schema for this entry.

3.2.2. Subschema Entries and Subentries

Subschema entries are used for administering information about the directory schema, in particular the object classes and attribute types supported by directory servers. A single subschema entry contains all schema definitions used by entries in a particular part of the directory tree.

Servers which follow X.500(93) models SHOULD implement subschema using the X.500 subschema mechanisms, and so these subschemas are not ordinary entries. LDAP clients SHOULD NOT assume that servers implement any of the other aspects of X.500 subschema. A server which masters entries and permits clients to modify these entries MUST implement and provide access to these subschema entries, so that its clients may discover the attributes and object classes which are permitted to be present. It is strongly recommended that all other servers implement this as well.

The following four attributes MUST be present in all subschema entries:

- cn: this attribute MUST be used to form the RDN of the subschema entry.

- objectClass: the attribute MUST have at least the values "top" and "subschema".

- objectClasses: each value of this attribute specifies an object class known to the server.

- attributeTypes: each value of this attribute specifies an attribute type known to the server.

These are defined in [5]. Other attributes MAY be present in subschema entries, to reflect additional supported capabilities.

These include matchingRules, matchingRuleUse, dITStructureRules, dITContentRules, nameForms and ldapSyntaxes.

Servers SHOULD provide the attributes createTimestamp and modifyTimestamp in subschema entries, in order to allow clients to maintain their caches of schema information.

Clients MUST only retrieve attributes from a subschema entry by requesting a base object search of the entry, where the search filter is "(objectClass=subschema)". (This will allow LDAPv3 servers which gateway to X.500(93) to detect that subentry information is being requested.)

3.3. Relationship to X.500

This document defines LDAP in terms of X.500 as an X.500 access mechanism. An LDAP server MUST act in accordance with the X.500(1993) series of ITU recommendations when providing the service. However, it is not required that an LDAP server make use of any X.500 protocols in providing this service, e.g. LDAP can be mapped onto any other directory system so long as the X.500 data and service model as used in LDAP is not violated in the LDAP interface.

3.4. Server-specific Data Requirements

An LDAP server MUST provide information about itself and other information that is specific to each server. This is represented as a group of attributes located in the root DSE (DSA-Specific Entry), which is named with the zero-length LDAPDN. These attributes are retrievable if a client performs a base object search of the root with filter "(objectClass=*)", however they are subject to access control restrictions. The root DSE MUST NOT be included if the client performs a subtree search starting from the root.

Servers may allow clients to modify these attributes.

The following attributes of the root DSE are defined in section 5 of [5]. Additional attributes may be defined in other documents.

- namingContexts: naming contexts held in the server. Naming contexts are defined in section 17 of X.501 [6].

- subschemaSubentry: subschema entries (or subentries) known by this server.

- altServer: alternative servers in case this one is later unavailable.

- supportedExtension: list of supported extended operations.

- supportedControl: list of supported controls.

- supportedSASLMechanisms: list of supported SASL security features.

- supportedLDAPVersion: LDAP versions implemented by the server.

If the server does not master entries and does not know the locations

of schema information, the subschemaSubentry attribute is not present in the root DSE. If the server masters directory entries under one or more schema rules, there may be any number of values of the subschemaSubentry attribute in the root DSE.

4. Elements of Protocol

The LDAP protocol is described using Abstract Syntax Notation 1 (ASN.1) [3], and is typically transferred using a subset of ASN.1 Basic Encoding Rules [11]. In order to support future extensions to this protocol, clients and servers MUST ignore elements of SEQUENCE encodings whose tags they do not recognize.

Note that unlike X.500, each change to the LDAP protocol other than through the extension mechanisms will have a different version number. A client will indicate the version it supports as part of the bind request, described in section 4.2. If a client has not sent a bind, the server MUST assume that version 3 is supported in the client (since version 2 required that the client bind first).

Clients may determine the protocol version a server supports by reading the supportedLDAPVersion attribute from the root DSE. Servers which implement version 3 or later versions MUST provide this attribute. Servers which only implement version 2 may not provide this attribute.

4.1. Common Elements

This section describes the LDAPMessage envelope PDU (Protocol Data Unit) format, as well as data type definitions which are used in the protocol operations.

4.1.1. Message Envelope

For the purposes of protocol exchanges, all protocol operations are encapsulated in a common envelope, the LDAPMessage, which is defined as follows:

```
     LDAPMessage ::= SEQUENCE {
```



```
                messageID       MessageID,
                protocolOp      CHOICE {
                        bindRequest     BindRequest,
                        bindResponse    BindResponse,
                        unbindRequest   UnbindRequest,
                        searchRequest   SearchRequest,
                        searchResEntry  SearchResultEntry,
                        searchResDone   SearchResultDone,
                        searchResRef    SearchResultReference,
                        modifyRequest   ModifyRequest,
                        modifyResponse  ModifyResponse,
                        addRequest      AddRequest,
                        addResponse     AddResponse,
                        delRequest      DelRequest,
                        delResponse     DelResponse,
                        modDNRequest    ModifyDNRequest,
                        modDNResponse   ModifyDNResponse,
                        compareRequest  CompareRequest,
                        compareResponse CompareResponse,
                        abandonRequest  AbandonRequest,
                        extendedReq     ExtendedRequest,
```

```
                    extendedResp    ExtendedResponse },
              controls       [0] Controls OPTIONAL }

     MessageID ::= INTEGER (0 .. maxInt)

     maxInt INTEGER ::= 2147483647 -- (2^^31 - 1) --
```

The function of the LDAPMessage is to provide an envelope containing common fields required in all protocol exchanges. At this time the only common fields are the message ID and the controls.

If the server receives a PDU from the client in which the LDAPMessage SEQUENCE tag cannot be recognized, the messageID cannot be parsed, the tag of the protocolOp is not recognized as a request, or the encoding structures or lengths of data fields are found to be incorrect, then the server MUST return the notice of disconnection described in section 4.4.1, with resultCode protocolError, and immediately close the connection. In other cases that the server cannot parse the request received by the client, the server MUST return an appropriate response to the request, with the resultCode set to protocolError.

If the client receives a PDU from the server which cannot be parsed, the client may discard the PDU, or may abruptly close the connection.

The ASN.1 type Controls is defined in section 4.1.12.

4.1.1.1. Message ID

All LDAPMessage envelopes encapsulating responses contain the messageID value of the corresponding request LDAPMessage.

The message ID of a request MUST have a value different from the values of any other requests outstanding in the LDAP session of which this message is a part.

A client MUST NOT send a second request with the same message ID as an earlier request on the same connection if the client has not received the final response from the earlier request. Otherwise the behavior is undefined. Typical clients increment a counter for each request.

A client MUST NOT reuse the message id of an abandonRequest or of the abandoned operation until it has received a response from the server for another request invoked subsequent to the abandonRequest, as the abandonRequest itself does not have a response.

4.1.2. String Types

The LDAPString is a notational convenience to indicate that, although strings of LDAPString type encode as OCTET STRING types, the ISO 10646 [13] character set (a superset of Unicode) is used, encoded following the UTF-8 algorithm [14]. Note that in the UTF-8 algorithm characters which are the same as ASCII (0x0000 through 0x007F) are represented as that same ASCII character in a single byte. The other byte values are used to form a variable-length encoding of an arbitrary character.

```
     LDAPString ::= OCTET STRING
```

The LDAPOID is a notational convenience to indicate that the permitted value of this string is a (UTF-8 encoded) dotted-decimal representation of an OBJECT IDENTIFIER.

```
LDAPOID ::= OCTET STRING
```

For example,

```
1.3.6.1.4.1.1466.1.2.3
```

4.1.3. Distinguished Name and Relative Distinguished Name

An LDAPDN and a RelativeLDAPDN are respectively defined to be the representation of a Distinguished Name and a Relative Distinguished Name after encoding according to the specification in [4], such that


```
<distinguished-name> ::= <name>

<relative-distinguished-name> ::= <name-component>
```

where <name> and <name-component> are as defined in [4].

```
LDAPDN ::= LDAPString

RelativeLDAPDN ::= LDAPString
```

Only Attribute Types can be present in a relative distinguished name component; the options of Attribute Descriptions (next section) MUST NOT be used in specifying distinguished names.

4.1.4. Attribute Type

An AttributeType takes on as its value the textual string associated with that AttributeType in its specification.

```
AttributeType ::= LDAPString
```

Each attribute type has a unique OBJECT IDENTIFIER which has been assigned to it. This identifier may be written as decimal digits with components separated by periods, e.g. "2.5.4.10".

A specification may also assign one or more textual names for an attribute type. These names MUST begin with a letter, and only contain ASCII letters, digit characters and hyphens. They are case insensitive. (These ASCII characters are identical to ISO 10646 characters whose UTF-8 encoding is a single byte between 0x00 and 0x7F.)

If the server has a textual name for an attribute type, it MUST use a textual name for attributes returned in search results. The dotted-decimal OBJECT IDENTIFIER is only used if there is no textual name for an attribute type.

Attribute type textual names are non-unique, as two different specifications (neither in standards track RFCs) may choose the same name.

A server which masters or shadows entries SHOULD list all the attribute types it supports in the subschema entries, using the attributeTypes attribute. Servers which support an open-ended set of attributes SHOULD include at least the attributeTypes value for the

'objectClass' attribute. Clients MAY retrieve the attributeTypes value from subschema entries in order to obtain the OBJECT IDENTIFIER and other information associated with attribute types.

Some attribute type names which are used in this version of LDAP are described in [5]. Servers may implement additional attribute types.

4.1.5. Attribute Description

An AttributeDescription is a superset of the definition of the AttributeType. It has the same ASN.1 definition, but allows additional options to be specified. They are also case insensitive.

```
AttributeDescription ::= LDAPString
```

A value of AttributeDescription is based on the following BNF:

```
<AttributeDescription> ::= <AttributeType> [ ";" <options> ]

<options>  ::= <option> | <option> ";" <options>

<option>   ::= <opt-char> <opt-char>*

<opt-char> ::=  ASCII-equivalent letters, numbers and hyphen
```

Examples of valid AttributeDescription:

```
cn
userCertificate;binary
```

One option, "binary", is defined in this document. Additional options may be defined in IETF standards-track and experimental RFCs. Options beginning with "x-" are reserved for private experiments. Any option could be associated with any AttributeType, although not all combinations may be supported by a server.

An AttributeDescription with one or more options is treated as a subtype of the attribute type without any options. Options present in an AttributeDescription are never mutually exclusive. Implementations MUST generate the <options> list sorted in ascending order, and servers MUST treat any two AttributeDescription with the same AttributeType and options as equivalent. A server will treat an AttributeDescription with any options it does not implement as an unrecognized attribute type.

The data type "AttributeDescriptionList" describes a list of 0 or more attribute types. (A list of zero elements has special significance in the Search request.)

```
AttributeDescriptionList ::= SEQUENCE OF
        AttributeDescription
```


4.1.5.1. Binary Option

If the "binary" option is present in an AttributeDescription, it overrides any string-based encoding representation defined for that attribute in [5]. Instead the attribute is to be transferred as a binary value encoded using the Basic Encoding Rules [11]. The syntax of the binary value is an ASN.1 data type definition which is referenced by the "SYNTAX" part of the attribute type definition.

The presence or absence of the "binary" option only affects the transfer of attribute values in protocol; servers store any particular attribute in a single format. If a client requests that a server return an attribute in the binary format, but the server cannot generate that format, the server MUST treat this attribute type as an unrecognized attribute type. Similarly, clients MUST NOT expect servers to return an attribute in binary format if the client requested that attribute by name without the binary option.

This option is intended to be used with attributes whose syntax is a complex ASN.1 data type, and the structure of values of that type is needed by clients. Examples of this kind of syntax are "Certificate" and "CertificateList".

4.1.6. Attribute Value

A field of type AttributeValue takes on as its value either a string encoding of a AttributeValue data type, or an OCTET STRING containing an encoded binary value, depending on whether the "binary" option is present in the companion AttributeDescription to this AttributeValue.

The definition of string encodings for different syntaxes and types may be found in other documents, and in particular [5].

```
AttributeValue ::= OCTET STRING
```

Note that there is no defined limit on the size of this encoding; thus protocol values may include multi-megabyte attributes (e.g. photographs).

Attributes may be defined which have arbitrary and non-printable syntax. Implementations MUST NEITHER simply display nor attempt to decode as ASN.1 a value if its syntax is not known. The implementation may attempt to discover the subschema of the source entry, and retrieve the values of attributeTypes from it.

Clients MUST NOT send attribute values in a request which are not valid according to the syntax defined for the attributes.

4.1.7. Attribute Value Assertion

The AttributeValueAssertion type definition is similar to the one in the X.500 directory standards. It contains an attribute description and a matching rule assertion value suitable for that type.

```
AttributeValueAssertion ::= SEQUENCE {
        attributeDesc   AttributeDescription,
        assertionValue  AssertionValue }

AssertionValue ::= OCTET STRING
```

If the "binary" option is present in attributeDesc, this signals to

the server that the assertionValue is a binary encoding of the assertion value.

For all the string-valued user attributes described in [5], the assertion value syntax is the same as the value syntax. Clients may use attribute values as assertion values in compare requests and search filters.

Note however that the assertion syntax may be different from the value syntax for other attributes or for non-equality matching rules. These may have an assertion syntax which contains only part of the value. See section 20.2.1.8 of X.501 [6] for examples.

4.1.8. Attribute

An attribute consists of a type and one or more values of that type. (Though attributes MUST have at least one value when stored, due to access control restrictions the set may be empty when transferred in protocol. This is described in section 4.5.2, concerning the PartialAttributeList type.)

```
Attribute ::= SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }
```

Each attribute value is distinct in the set (no duplicates). The order of attribute values within the vals set is undefined and implementation-dependent, and MUST NOT be relied upon.

4.1.9. Matching Rule Identifier

A matching rule is a means of expressing how a server should compare an AssertionValue received in a search filter with an abstract data value. The matching rule defines the syntax of the assertion value and the process to be performed in the server.

An X.501(1993) Matching Rule is identified in the LDAP protocol by the printable representation of its OBJECT IDENTIFIER, either as one of the strings given in [5], or as decimal digits with components separated by periods, e.g. "caseIgnoreIA5Match" or "1.3.6.1.4.1.453.33.33".

```
MatchingRuleId ::= LDAPString
```

Servers which support matching rules for use in the extensibleMatch search filter MUST list the matching rules they implement in subschema entries, using the matchingRules attributes. The server SHOULD also list there, using the matchingRuleUse attribute, the attribute types with which each matching rule can be used. More information is given in section 4.4 of [5].

4.1.10. Result Message

The LDAPResult is the construct used in this protocol to return success or failure indications from servers to clients. In response to various requests servers will return responses containing fields of type LDAPResult to indicate the final status of a protocol operation request.

```
LDAPResult ::= SEQUENCE {
        resultCode      ENUMERATED {
```

```
                        success                      (0),
                        operationsError              (1),
                        protocolError                (2),
                        timeLimitExceeded            (3),
                        sizeLimitExceeded            (4),
                        compareFalse                 (5),
                        compareTrue                  (6),

                        authMethodNotSupported       (7),
                        strongAuthRequired           (8),
                                -- 9 reserved --
                        referral                     (10),  -- new
                        adminLimitExceeded           (11),  -- new
                        unavailableCriticalExtension (12),  -- new
                        confidentialityRequired      (13),  -- new
                        saslBindInProgress           (14),  -- new
                        noSuchAttribute              (16),
                        undefinedAttributeType       (17),
                        inappropriateMatching        (18),
                        constraintViolation          (19),
                        attributeOrValueExists       (20),
                        invalidAttributeSyntax       (21),
                                -- 22-31 unused --
```



```
                        noSuchObject                 (32),
                        aliasProblem                 (33),
                        invalidDNSyntax              (34),
                        -- 35 reserved for undefined isLeaf --
                        aliasDereferencingProblem    (36),
                                -- 37-47 unused --
                        inappropriateAuthentication  (48),
                        invalidCredentials           (49),
                        insufficientAccessRights     (50),
                        busy                         (51),
                        unavailable                  (52),
                        unwillingToPerform           (53),
                        loopDetect                   (54),
                                -- 55-63 unused --
                        namingViolation              (64),
                        objectClassViolation         (65),
                        notAllowedOnNonLeaf          (66),
                        notAllowedOnRDN              (67),
                        entryAlreadyExists           (68),
                        objectClassModsProhibited    (69),
                                -- 70 reserved for CLDAP --
                        affectsMultipleDSAs          (71), -- new
                                -- 72-79 unused --
                        other                        (80) },
                        -- 81-90 reserved for APIs --
                matchedDN       LDAPDN,
                errorMessage    LDAPString,
                referral        [3] Referral OPTIONAL }
```

All the result codes with the exception of success, compareFalse and compareTrue are to be treated as meaning the operation could not be completed in its entirety.

Most of the result codes are based on problem indications from X.511 error data types. Result codes from 16 to 21 indicate an AttributeProblem, codes 32, 33, 34 and 36 indicate a NameProblem, codes 48, 49 and 50 indicate a SecurityProblem, codes 51 to 54

indicate a ServiceProblem, and codes 64 to 69 and 71 indicates an UpdateProblem.

If a client receives a result code which is not listed above, it is to be treated as an unknown error condition.

The errorMessage field of this construct may, at the server's option, be used to return a string containing a textual, human-readable (terminal control and page formatting characters should be avoided) error diagnostic. As this error diagnostic is not standardized,

implementations MUST NOT rely on the values returned. If the server chooses not to return a textual diagnostic, the errorMessage field of the LDAPResult type MUST contain a zero length string.

For result codes of noSuchObject, aliasProblem, invalidDNSyntax and aliasDereferencingProblem, the matchedDN field is set to the name of the lowest entry (object or alias) in the directory that was matched. If no aliases were dereferenced while attempting to locate the entry, this will be a truncated form of the name provided, or if aliases were dereferenced, of the resulting name, as defined in section 12.5 of X.511 [8]. The matchedDN field is to be set to a zero length string with all other result codes.

4.1.11. Referral

The referral error indicates that the contacted server does not hold the target entry of the request. The referral field is present in an LDAPResult if the LDAPResult.resultCode field value is referral, and absent with all other result codes. It contains a reference to another server (or set of servers) which may be accessed via LDAP or other protocols. Referrals can be returned in response to any operation request (except unbind and abandon which do not have responses). At least one URL MUST be present in the Referral.

The referral is not returned for a singleLevel or wholeSubtree search in which the search scope spans multiple naming contexts, and several different servers would need to be contacted to complete the operation. Instead, continuation references, described in section 4.5.3, are returned.

```
Referral ::= SEQUENCE OF LDAPURL  -- one or more

LDAPURL ::= LDAPString -- limited to characters permitted in URLs
```

If the client wishes to progress the operation, it MUST follow the referral by contacting any one of servers. All the URLs MUST be equally capable of being used to progress the operation. (The mechanisms for how this is achieved by multiple servers are outside the scope of this document.)

URLs for servers implementing the LDAP protocol are written according to [9]. If an alias was dereferenced, the <dn> part of the URL MUST be present, with the new target object name. If the <dn> part is present, the client MUST use this name in its next request to progress the operation, and if it is not present the client will use the same name as in the original request. Some servers (e.g. participating in distributed indexing) may provide a different filter in a referral for a search operation. If the filter part of the URL

is present in an LDAPURL, the client MUST use this filter in its next request to progress this search, and if it is not present the client MUST use the same filter as it used for that search. Other aspects of the new request may be the same or different as the request which generated the referral.

Note that UTF-8 characters appearing in a DN or search filter may not be legal for URLs (e.g. spaces) and MUST be escaped using the % method in RFC 1738 [7].

Other kinds of URLs may be returned, so long as the operation could be performed using that protocol.

4.1.12. Controls

A control is a way to specify extension information. Controls which are sent as part of a request apply only to that request and are not saved.

```
Controls ::= SEQUENCE OF Control

Control ::= SEQUENCE {
        controlType             LDAPOID,
        criticality             BOOLEAN DEFAULT FALSE,
        controlValue            OCTET STRING OPTIONAL }
```

The controlType field MUST be a UTF-8 encoded dotted-decimal representation of an OBJECT IDENTIFIER which uniquely identifies the control. This prevents conflicts between control names.

The criticality field is either TRUE or FALSE.

If the server recognizes the control type and it is appropriate for the operation, the server will make use of the control when performing the operation.

If the server does not recognize the control type and the criticality field is TRUE, the server MUST NOT perform the operation, and MUST instead return the resultCode unsupportedCriticalExtension.

If the control is not appropriate for the operation and criticality field is TRUE, the server MUST NOT perform the operation, and MUST instead return the resultCode unsupportedCriticalExtension.

If the control is unrecognized or inappropriate but the criticality field is FALSE, the server MUST ignore the control.

The controlValue contains any information associated with the control, and its format is defined for the control. The server MUST be prepared to handle arbitrary contents of the controlValue octet string, including zero bytes. It is absent only if there is no value information which is associated with a control of its type.

This document does not define any controls. Controls may be defined in other documents. The definition of a control consists of:

- the OBJECT IDENTIFIER assigned to the control,

- whether the control is always noncritical, always critical, or critical at the client's option,

- the format of the controlValue contents of the control.

Servers list the controls which they recognize in the supportedControl attribute in the root DSE.

4.2. Bind Operation

The function of the Bind Operation is to allow authentication information to be exchanged between the client and server.

The Bind Request is defined as follows:

```
BindRequest ::= [APPLICATION 0] SEQUENCE {
        version                 INTEGER (1 .. 127),
        name                    LDAPDN,
        authentication          AuthenticationChoice }

AuthenticationChoice ::= CHOICE {
        simple                  [0] OCTET STRING,
                                 -- 1 and 2 reserved
        sasl                    [3] SaslCredentials }

SaslCredentials ::= SEQUENCE {
        mechanism               LDAPString,
        credentials             OCTET STRING OPTIONAL }
```

Parameters of the Bind Request are:

- version: A version number indicating the version of the protocol to be used in this protocol session. This document describes version 3 of the LDAP protocol. Note that there is no version negotiation, and the client just sets this parameter to the version it desires. If the client requests protocol version 2, a server that supports the version 2 protocol as described in [2] will not return any v3-

specific protocol fields. (Note that not all LDAP servers will support protocol version 2, since they may be unable to generate the attribute syntaxes associated with version 2.)

- name: The name of the directory object that the client wishes to bind as. This field may take on a null value (a zero length string) for the purposes of anonymous binds, when authentication has been performed at a lower layer, or when using SASL credentials with a mechanism that includes the LDAPDN in the credentials.

- authentication: information used to authenticate the name, if any, provided in the Bind Request.

Upon receipt of a Bind Request, a protocol server will authenticate the requesting client, if necessary. The server will then return a Bind Response to the client indicating the status of the authentication.

Authorization is the use of this authentication information when performing operations. Authorization MAY be affected by factors outside of the LDAP Bind request, such as lower layer security services.

4.2.1. Sequencing of the Bind Request

For some SASL authentication mechanisms, it may be necessary for the client to invoke the BindRequest multiple times. If at any stage the client wishes to abort the bind process it MAY unbind and then drop the underlying connection. Clients MUST NOT invoke operations between two Bind requests made as part of a multi-stage bind.

A client may abort a SASL bind negotiation by sending a BindRequest with a different value in the mechanism field of SaslCredentials, or an AuthenticationChoice other than sasl.

If the client sends a BindRequest with the sasl mechanism field as an empty string, the server MUST return a BindResponse with authMethodNotSupported as the resultCode. This will allow clients to abort a negotiation if it wishes to try again with the same SASL mechanism.

Unlike LDAP v2, the client need not send a Bind Request in the first PDU of the connection. The client may request any operations and the server MUST treat these as unauthenticated. If the server requires that the client bind before browsing or modifying the directory, the server MAY reject a request other than binding, unbinding or an extended request with the "operationsError" result.

If the client did not bind before sending a request and receives an operationsError, it may then send a Bind Request. If this also fails or the client chooses not to bind on the existing connection, it will close the connection, reopen it and begin again by first sending a PDU with a Bind Request. This will aid in interoperating with servers implementing other versions of LDAP.

Clients MAY send multiple bind requests on a connection to change their credentials. A subsequent bind process has the effect of abandoning all operations outstanding on the connection. (This simplifies server implementation.) Authentication from earlier binds are subsequently ignored, and so if the bind fails, the connection will be treated as anonymous. If a SASL transfer encryption or integrity mechanism has been negotiated, and that mechanism does not support the changing of credentials from one identity to another, then the client MUST instead establish a new connection.

4.2.2. Authentication and Other Security Services

The simple authentication option provides minimal authentication facilities, with the contents of the authentication field consisting only of a cleartext password. Note that the use of cleartext passwords is not recommended over open networks when there is no authentication or encryption being performed by a lower layer; see the "Security Considerations" section.

If no authentication is to be performed, then the simple authentication option MUST be chosen, and the password be of zero length. (This is often done by LDAPv2 clients.) Typically the DN is

also of zero length.

The sasl choice allows for any mechanism defined for use with SASL [12]. The mechanism field contains the name of the mechanism. The credentials field contains the arbitrary data used for authentication, inside an OCTET STRING wrapper. Note that unlike some Internet application protocols where SASL is used, LDAP is not text-based, thus no base64 transformations are performed on the credentials.

If any SASL-based integrity or confidentiality services are enabled, they take effect following the transmission by the server and reception by the client of the final BindResponse with resultCode success.

The client can request that the server use authentication information from a lower layer protocol by using the SASL EXTERNAL mechanism.

4.2.3. Bind Response

The Bind Response is defined as follows.

```
BindResponse ::= [APPLICATION 1] SEQUENCE {
     COMPONENTS OF LDAPResult,
     serverSaslCreds    [7] OCTET STRING OPTIONAL }
```

BindResponse consists simply of an indication from the server of he status of the client's request for authentication.

f the bind was successful, the resultCode will be success, therwise it will be one of:

- operationsError: server encountered an internal error,
- protocolError: unrecognized version number or incorrect PDU structure,
- authMethodNotSupported: unrecognized SASL mechanism name,
- strongAuthRequired: the server requires authentication be performed with a SASL mechanism,
- referral: this server cannot accept this bind and the client should try another,
- saslBindInProgress: the server requires the client to send a new bind request, with the same sasl mechanism, to continue the authentication process,
- inappropriateAuthentication: the server requires the client which had attempted to bind anonymously or without supplying credentials to provide some form of credentials,
- invalidCredentials: the wrong password was supplied or the SASL credentials could not be processed,
- unavailable: the server is shutting down.

If the server does not support the client's requested protocol

version, it MUST set the resultCode to protocolError.

If the client receives a BindResponse response where the resultCode was protocolError, it MUST close the connection as the server will be unwilling to accept further operations. (This is for compatibility with earlier versions of LDAP, in which the bind was always the first operation, and there was no negotiation.)

The serverSaslCreds are used as part of a SASL-defined bind mechanism to allow the client to authenticate the server to which it is communicating, or to perform "challenge-response" authentication. If the client bound with the password choice, or the SASL mechanism does not require the server to return information to the client, then this field is not to be included in the result.

4.3. Unbind Operation

The function of the Unbind Operation is to terminate a protocol session. The Unbind Operation is defined as follows:

```
UnbindRequest ::= [APPLICATION 2] NULL
```

The Unbind Operation has no response defined. Upon transmission of an UnbindRequest, a protocol client may assume that the protocol session is terminated. Upon receipt of an UnbindRequest, a protocol server may assume that the requesting client has terminated the session and that all outstanding requests may be discarded, and may close the connection.

4.4. Unsolicited Notification

An unsolicited notification is an LDAPMessage sent from the server to the client which is not in response to any LDAPMessage received by the server. It is used to signal an extraordinary condition in the server or in the connection between the client and the server. The notification is of an advisory nature, and the server will not expect any response to be returned from the client.

The unsolicited notification is structured as an LDAPMessage in which the messageID is 0 and protocolOp is of the extendedResp form. The responseName field of the ExtendedResponse is present. The LDAPOID value MUST be unique for this notification, and not be used in any other situation.

One unsolicited notification is defined in this document.

4.4.1. Notice of Disconnection

This notification may be used by the server to advise the client that the server is about to close the connection due to an error condition. Note that this notification is NOT a response to an unbind requested by the client: the server MUST follow the procedures of section 4.3. This notification is intended to assist clients in distinguishing between an error condition and a transient network

failure. As with a connection close due to network failure, the client MUST NOT assume that any outstanding requests which modified the directory have succeeded or failed.

The responseName is 1.3.6.1.4.1.1466.20036, the response field is absent, and the resultCode is used to indicate the reason for the disconnection.

The following resultCode values are to be used in this notification:

- protocolError: The server has received data from the client in which the LDAPMessage structure could not be parsed.

- strongAuthRequired: The server has detected that an established underlying security association protecting communication between the client and server has unexpectedly failed or been compromised.

- unavailable: This server will stop accepting new connections and operations on all existing connections, and be unavailable for an extended period of time. The client may make use of an alternative server.

After sending this notice, the server MUST close the connection. After receiving this notice, the client MUST NOT transmit any further on the connection, and may abruptly close the connection.

4.5. Search Operation

The Search Operation allows a client to request that a search be performed on its behalf by a server. This can be used to read attributes from a single entry, from entries immediately below a particular entry, or a whole subtree of entries.

4.5.1. Search Request

The Search Request is defined as follows:

```
SearchRequest ::= [APPLICATION 3] SEQUENCE {
        baseObject      LDAPDN,
        scope           ENUMERATED {
                baseObject              (0),
                singleLevel             (1),
                wholeSubtree            (2) },
        derefAliases    ENUMERATED {
                neverDerefAliases       (0),
                derefInSearching        (1),
                derefFindingBaseObj     (2),
```



```
                derefAlways             (3) },
        sizeLimit       INTEGER (0 .. maxInt),
        timeLimit       INTEGER (0 .. maxInt),
        typesOnly       BOOLEAN,
        filter          Filter,
        attributes      AttributeDescriptionList }

Filter ::= CHOICE {
        and             [0] SET OF Filter,
```

```
                or              [1] SET OF Filter,
                not             [2] Filter,
                equalityMatch   [3] AttributeValueAssertion,
                substrings      [4] SubstringFilter,
                greaterOrEqual  [5] AttributeValueAssertion,
                lessOrEqual     [6] AttributeValueAssertion,
                present         [7] AttributeDescription,
                approxMatch     [8] AttributeValueAssertion,
                extensibleMatch [9] MatchingRuleAssertion }

        SubstringFilter ::= SEQUENCE {
                type            AttributeDescription,
                -- at least one must be present
                substrings      SEQUENCE OF CHOICE {
                        initial [0] LDAPString,
                        any     [1] LDAPString,
                        final   [2] LDAPString } }

        MatchingRuleAssertion ::= SEQUENCE {
                matchingRule    [1] MatchingRuleId OPTIONAL,
                type            [2] AttributeDescription OPTIONAL,
                matchValue      [3] AssertionValue,
                dnAttributes    [4] BOOLEAN DEFAULT FALSE }
```

Parameters of the Search Request are:

- baseObject: An LDAPDN that is the base object entry relative to which the search is to be performed.

- scope: An indicator of the scope of the search to be performed. The semantics of the possible values of this field are identical to the semantics of the scope field in the X.511 Search Operation.

- derefAliases: An indicator as to how alias objects (as defined in X.501) are to be handled in searching. The semantics of the possible values of this field are:

  neverDerefAliases: do not dereference aliases in searching or in locating the base object of the search;

  derefInSearching: dereference aliases in subordinates of the base object in searching, but not in locating the base object of the search;

  derefFindingBaseObj: dereference aliases in locating the base object of the search, but not when searching subordinates of the base object;

  derefAlways: dereference aliases both in searching and in locating the base object of the search.

- sizelimit: A sizelimit that restricts the maximum number of entries to be returned as a result of the search. A value of 0 in this field indicates that no client-requested sizelimit restrictions are in effect for the search. Servers may enforce a maximum number of entries to return.

- timelimit: A timelimit that restricts the maximum time (in seconds) allowed for a search. A value of 0 in this field indicates that no client-requested timelimit restrictions are in effect for the search.

- typesOnly: An indicator as to whether search results will contain both attribute types and values, or just attribute types. Setting this field to TRUE causes only attribute types (no values) to be returned. Setting this field to FALSE causes both attribute types and values to be returned.

- filter: A filter that defines the conditions that must be fulfilled in order for the search to match a given entry.

  The 'and', 'or' and 'not' choices can be used to form combinations of filters. At least one filter element MUST be present in an 'and' or 'or' choice. The others match against individual attribute values of entries in the scope of the search. (Implementor's note: the 'not' filter is an example of a tagged choice in an implicitly-tagged module. In BER this is treated as if the tag was explicit.)

  A server MUST evaluate filters according to the three-valued logic of X.511(93) section 7.8.1. In summary, a filter is evaluated to either "TRUE", "FALSE" or "Undefined". If the filter evaluates to TRUE for a particular entry, then the attributes of that entry are returned as part of the search result (subject to any applicable access control restrictions). If the filter evaluates to FALSE or Undefined, then the entry is ignored for the search.

A filter of the "and" choice is TRUE if all the filters in the SET OF evaluate to TRUE, FALSE if at least one filter is FALSE, and otherwise Undefined. A filter of the "or" choice is FALSE if all of the filters in the SET OF evaluate to FALSE, TRUE if at least one filter is TRUE, and Undefined otherwise. A filter of the "not" choice is TRUE if the filter being negated is FALSE, FALSE if it is TRUE, and Undefined if it is Undefined.

The present match evaluates to TRUE where there is an attribute or subtype of the specified attribute description present in an entry, and FALSE otherwise (including a presence test with an unrecognized attribute description.)

The extensibleMatch is new in this version of LDAP. If the matchingRule field is absent, the type field MUST be present, and the equality match is performed for that type. If the type field is absent and matchingRule is present, the matchValue is compared against all attributes in an entry which support that matchingRule, and the matchingRule determines the syntax for the assertion value (the filter item evaluates to TRUE if it matches with at least one attribute in the entry, FALSE if it does not match any attribute in the entry, and Undefined if the matchingRule is not recognized or the assertionValue cannot be parsed.) If the type field is present and matchingRule is present, the matchingRule MUST be one permitted for use with that type, otherwise the filter item is undefined. If the dnAttributes field is set to TRUE, the match is applied against all the attributes in an entry's distinguished name as well, and also evaluates to TRUE if there is at least one attribute in the distinguished name for which the filter item evaluates to TRUE. (Editors note: The dnAttributes field is present so that there does not need to be multiple versions of generic matching rules such as for word matching, one to apply to entries and another to apply to entries and dn attributes as well).

A filter item evaluates to Undefined when the server would not be able to determine whether the assertion value matches an entry. If an attribute description in an equalityMatch, substrings, greaterOrEqual, lessOrEqual, approxMatch or extensibleMatch filter is not recognized by the server, a matching rule id in the extensibleMatch is not recognized by the server, the assertion value cannot be parsed, or the type of filtering requested is not implemented, then the filter is Undefined. Thus for example if a server did not recognize the attribute type shoeSize, a filter of (shoeSize=*) would evaluate to FALSE, and the filters (shoeSize=12), (shoeSize>=12) and (shoeSize<=12) would evaluate to Undefined.

Servers MUST NOT return errors if attribute descriptions or matching rule ids are not recognized, or assertion values cannot be parsed. More details of filter processing are given in section 7.8 of X.511 [8].

- attributes: A list of the attributes to be returned from each entry which matches the search filter. There are two special values which may be used: an empty list with no attributes, and the attribute description string "*". Both of these signify that all user attributes are to be returned. (The "*" allows the client to request all user attributes in addition to specific operational attributes).

  Attributes MUST be named at most once in the list, and are returned at most once in an entry. If there are attribute descriptions in the list which are not recognized, they are ignored by the server.

  If the client does not want any attributes returned, it can specify a list containing only the attribute with OID "1.1". This OID was chosen arbitrarily and does not correspond to any attribute in use.

  Client implementors should note that even if all user attributes are requested, some attributes of the entry may not be included in search results due to access control or other restrictions. Furthermore, servers will not return operational attributes, such as objectClasses or attributeTypes, unless they are listed by name, since there may be extremely large number of values for certain operational attributes. (A list of operational attributes for use in LDAP is given in [5].)

Note that an X.500 "list"-like operation can be emulated by the client requesting a one-level LDAP search operation with a filter checking for the existence of the objectClass attribute, and that an X.500 "read"-like operation can be emulated by a base object LDAP search operation with the same filter. A server which provides a gateway to X.500 is not required to use the Read or List operations, although it may choose to do so, and if it does must provide the same semantics as the X.500 search operation.

4.5.2. Search Result

The results of the search attempted by the server upon receipt of a Search Request are returned in Search Responses, which are LDAP messages containing either SearchResultEntry, SearchResultReference, ExtendedResponse or SearchResultDone data types.

```
        SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
                objectName      LDAPDN,
```



```
                attributes      PartialAttributeList }

        PartialAttributeList ::= SEQUENCE OF SEQUENCE {
                type    AttributeDescription,
                vals    SET OF AttributeValue }
        -- implementors should note that the PartialAttributeList may
        -- have zero elements (if none of the attributes of that entry
        -- were requested, or could be returned), and that the vals set
        -- may also have zero elements (if types only was requested, or
        -- all values were excluded from the result.)

        SearchResultReference ::= [APPLICATION 19] SEQUENCE OF LDAPURL
        -- at least one LDAPURL element must be present

        SearchResultDone ::= [APPLICATION 5] LDAPResult
```

Upon receipt of a Search Request, a server will perform the necessary search of the DIT.

If the LDAP session is operating over a connection-oriented transport such as TCP, the server will return to the client a sequence of responses in separate LDAP messages. There may be zero or more responses containing SearchResultEntry, one for each entry found during the search. There may also be zero or more responses containing SearchResultReference, one for each area not explored by this server during the search. The SearchResultEntry and SearchResultReference PDUs may come in any order. Following all the SearchResultReference responses and all SearchResultEntry responses to be returned by the server, the server will return a response containing the SearchResultDone, which contains an indication of success, or detailing any errors that have occurred.

Each entry returned in a SearchResultEntry will contain all attributes, complete with associated values if necessary, as specified in the attributes field of the Search Request. Return of attributes is subject to access control and other administrative policy. Some attributes may be returned in binary format (indicated by the AttributeDescription in the response having the binary option present).

Some attributes may be constructed by the server and appear in a SearchResultEntry attribute list, although they are not stored attributes of an entry. Clients MUST NOT assume that all attributes can be modified, even if permitted by access control.

LDAPMessage responses of the ExtendedResponse form are reserved for returning information associated with a control requested by the client. These may be defined in future versions of this document.

4.5.3. Continuation References in the Search Result

If the server was able to locate the entry referred to by the baseObject but was unable to search all the entries in the scope at and under the baseObject, the server may return one or more SearchResultReference, each containing a reference to another set of servers for continuing the operation. A server MUST NOT return any SearchResultReference if it has not located the baseObject and thus has not searched any entries; in this case it would return a SearchResultDone containing a referral resultCode.

In the absence of indexing information provided to a server from servers holding subordinate naming contexts, SearchResultReference responses are not affected by search filters and are always returned when in scope.

The SearchResultReference is of the same data type as the Referral. URLs for servers implementing the LDAP protocol are written according to [9]. The <dn> part MUST be present in the URL, with the new target object name. The client MUST use this name in its next request. Some servers (e.g. part of a distributed index exchange system) may provide a different filter in the URLs of the SearchResultReference. If the filter part of the URL is present in an LDAP URL, the client MUST use the new filter in its next request to progress the search, and if the filter part is absent the client will use again the same filter. Other aspects of the new search request may be the same or different as the search which generated the continuation references.

Other kinds of URLs may be returned so long as the operation could be performed using that protocol.

The name of an unexplored subtree in a SearchResultReference need not be subordinate to the base object.

In order to complete the search, the client MUST issue a new search operation for each SearchResultReference that is returned. Note that the abandon operation described in section 4.11 applies only to a particular operation sent on a connection between a client and server, and if the client has multiple outstanding search operations to different servers, it MUST abandon each operation individually.

4.5.3.1. Example

For example, suppose the contacted server (hosta) holds the entry "O=MNN,C=WW" and the entry "CN=Manager,O=MNN,C=WW". It knows that either LDAP-capable servers (hostb) or (hostc) hold "OU=People,O=MNN,C=WW" (one is the master and the other server a

shadow), and that LDAP-capable server (hostd) holds the subtree "OU=Roles,O=MNN,C=WW". If a subtree search of "O=MNN,C=WW" is requested to the contacted server, it may return the following:

```
SearchResultEntry for O=MNN,C=WW
SearchResultEntry for CN=Manager,O=MNN,C=WW
SearchResultReference {
  ldap://hostb/OU=People,O=MNN,C=WW
  ldap://hostc/OU=People,O=MNN,C=WW
}
SearchResultReference {
  ldap://hostd/OU=Roles,O=MNN,C=WW
}
```

```
  SearchResultDone (success)
```

Client implementors should note that when following a SearchResultReference, additional SearchResultReference may be generated. Continuing the example, if the client contacted the server (hostb) and issued the search for the subtree "OU=People,O=MNN,C=WW", the server might respond as follows:

```
  SearchResultEntry for OU=People,O=MNN,C=WW
  SearchResultReference {
   ldap://hoste/OU=Managers,OU=People,O=MNN,C=WW
  }
  SearchResultReference {
   ldap://hostf/OU=Consultants,OU=People,O=MNN,C=WW
  }
  SearchResultDone (success)
```

If the contacted server does not hold the base object for the search, then it will return a referral to the client. For example, if the client requests a subtree search of "O=XYZ,C=US" to hosta, the server may return only a SearchResultDone containing a referral.

```
  SearchResultDone (referral) {
    ldap://hostg/
  }
```

4.6. Modify Operation

The Modify Operation allows a client to request that a modification of an entry be performed on its behalf by a server. The Modify Request is defined as follows:

```
     ModifyRequest ::= [APPLICATION 6] SEQUENCE {
             object          LDAPDN,
             modification    SEQUENCE OF SEQUENCE {
```



```
                     operation       ENUMERATED {
                                             add     (0),
                                             delete  (1),
                                             replace (2) },
                     modification    AttributeTypeAndValues } }

     AttributeTypeAndValues ::= SEQUENCE {
             type    AttributeDescription,
             vals    SET OF AttributeValue }
```

Parameters of the Modify Request are:

- object: The object to be modified. The value of this field contains the DN of the entry to be modified. The server will not perform any alias dereferencing in determining the object to be modified.

- modification: A list of modifications to be performed on the entry. The entire list of entry modifications MUST be performed in the order they are listed, as a single atomic operation. While individual modifications may violate the directory schema, the resulting entry after the entire list of modifications is performed MUST conform to the requirements of the directory schema. The values that may be taken on by the 'operation' field in each modification construct have the following semantics respectively:

add: add values listed to the given attribute, creating the attribute if necessary;

delete: delete values listed from the given attribute, removing the entire attribute if no values are listed, or if all current values of the attribute are listed for deletion;

replace: replace all existing values of the given attribute with the new values listed, creating the attribute if it did not already exist. A replace with no value will delete the entire attribute if it exists, and is ignored if the attribute does not exist.

The result of the modify attempted by the server upon receipt of a Modify Request is returned in a Modify Response, defined as follows:

```
ModifyResponse ::= [APPLICATION 7] LDAPResult
```

Upon receipt of a Modify Request, a server will perform the necessary modifications to the DIT.

The server will return to the client a single Modify Response indicating either the successful completion of the DIT modification, or the reason that the modification failed. Note that due to the requirement for atomicity in applying the list of modifications in the Modify Request, the client may expect that no modifications of the DIT have been performed if the Modify Response received indicates any sort of error, and that all requested modifications have been performed if the Modify Response indicates successful completion of the Modify Operation. If the connection fails, whether the modification occurred or not is indeterminate.

The Modify Operation cannot be used to remove from an entry any of its distinguished values, those values which form the entry's relative distinguished name. An attempt to do so will result in the server returning the error notAllowedOnRDN. The Modify DN Operation described in section 4.9 is used to rename an entry.

If an equality match filter has not been defined for an attribute type, clients MUST NOT attempt to delete individual values of that attribute from an entry using the "delete" form of a modification, and MUST instead use the "replace" form.

Note that due to the simplifications made in LDAP, there is not a direct mapping of the modifications in an LDAP ModifyRequest onto the EntryModifications of a DAP ModifyEntry operation, and different implementations of LDAP-DAP gateways may use different means of representing the change. If successful, the final effect of the operations on the entry MUST be identical.

4.7. Add Operation

The Add Operation allows a client to request the addition of an entry into the directory. The Add Request is defined as follows:

```
AddRequest ::= [APPLICATION 8] SEQUENCE {
        entry           LDAPDN,
        attributes      AttributeList }
```

```
AttributeList ::= SEQUENCE OF SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }
```

Parameters of the Add Request are:

- entry: the Distinguished Name of the entry to be added. Note that the server will not dereference any aliases in locating the entry to be added.

- attributes: the list of attributes that make up the content of the entry being added. Clients MUST include distinguished values (those forming the entry's own RDN) in this list, the objectClass attribute, and values of any mandatory attributes of the listed object classes. Clients MUST NOT supply the createTimestamp or creatorsName attributes, since these will be generated automatically by the server.

The entry named in the entry field of the AddRequest MUST NOT exist for the AddRequest to succeed. The parent of the entry to be added MUST exist. For example, if the client attempted to add "CN=JS,O=Foo,C=US", the "O=Foo,C=US" entry did not exist, and the "C=US" entry did exist, then the server would return the error noSuchObject with the matchedDN field containing "C=US". If the parent entry exists but is not in a naming context held by the server, the server SHOULD return a referral to the server holding the parent entry.

Servers implementations SHOULD NOT restrict where entries can be located in the directory. Some servers MAY allow the administrator to restrict the classes of entries which can be added to the directory.

Upon receipt of an Add Request, a server will attempt to perform the add requested. The result of the add attempt will be returned to the client in the Add Response, defined as follows:

```
AddResponse ::= [APPLICATION 9] LDAPResult
```

A response of success indicates that the new entry is present in the directory.

4.8. Delete Operation

The Delete Operation allows a client to request the removal of an entry from the directory. The Delete Request is defined as follows:

```
DelRequest ::= [APPLICATION 10] LDAPDN
```

The Delete Request consists of the Distinguished Name of the entry to be deleted. Note that the server will not dereference aliases while resolving the name of the target entry to be removed, and that only leaf entries (those with no subordinate entries) can be deleted with this operation.

The result of the delete attempted by the server upon receipt of a Delete Request is returned in the Delete Response, defined as follows:

```
DelResponse ::= [APPLICATION 11] LDAPResult
```

Upon receipt of a Delete Request, a server will attempt to perform the entry removal requested. The result of the delete attempt will be returned to the client in the Delete Response.

4.9. Modify DN Operation

The Modify DN Operation allows a client to change the leftmost (least significant) component of the name of an entry in the directory, or to move a subtree of entries to a new location in the directory. The Modify DN Request is defined as follows:

```
ModifyDNRequest ::= [APPLICATION 12] SEQUENCE {
        entry           LDAPDN,
        newrdn          RelativeLDAPDN,
        deleteoldrdn    BOOLEAN,
        newSuperior     [0] LDAPDN OPTIONAL }
```

Parameters of the Modify DN Request are:

- entry: the Distinguished Name of the entry to be changed. This entry may or may not have subordinate entries.

- newrdn: the RDN that will form the leftmost component of the new name of the entry.

- deleteoldrdn: a boolean parameter that controls whether the old RDN attribute values are to be retained as attributes of the entry, or deleted from the entry.

- newSuperior: if present, this is the Distinguished Name of the entry which becomes the immediate superior of the existing entry.

The result of the name change attempted by the server upon receipt of a Modify DN Request is returned in the Modify DN Response, defined as follows:

```
ModifyDNResponse ::= [APPLICATION 13] LDAPResult
```

Upon receipt of a ModifyDNRequest, a server will attempt to perform the name change. The result of the name change attempt will be returned to the client in the Modify DN Response.

For example, if the entry named in the "entry" parameter was "cn=John Smith,c=US", the newrdn parameter was "cn=John Cougar Smith", and the newSuperior parameter was absent, then this operation would

attempt to rename the entry to be "cn=John Cougar Smith,c=US". If there was already an entry with that name, the operation would fail with error code entryAlreadyExists.

If the deleteoldrdn parameter is TRUE, the values forming the old

RDN are deleted from the entry. If the deleteoldrdn parameter is FALSE, the values forming the old RDN will be retained as non-distinguished attribute values of the entry. The server may not perform the operation and return an error code if the setting of the deleteoldrdn parameter would cause a schema inconsistency in the entry.

Note that X.500 restricts the ModifyDN operation to only affect entries that are contained within a single server. If the LDAP server is mapped onto DAP, then this restriction will apply, and the resultCode affectsMultipleDSAs will be returned if this error occurred. In general clients MUST NOT expect to be able to perform arbitrary movements of entries and subtrees between servers.

4.10. Compare Operation

The Compare Operation allows a client to compare an assertion provided with an entry in the directory. The Compare Request is defined as follows:

```
CompareRequest ::= [APPLICATION 14] SEQUENCE {
        entry           LDAPDN,
        ava             AttributeValueAssertion }
```

Parameters of the Compare Request are:

- entry: the name of the entry to be compared with.

- ava: the assertion with which an attribute in the entry is to be compared.

The result of the compare attempted by the server upon receipt of a Compare Request is returned in the Compare Response, defined as follows:

```
CompareResponse ::= [APPLICATION 15] LDAPResult
```

Upon receipt of a Compare Request, a server will attempt to perform the requested comparison. The result of the comparison will be returned to the client in the Compare Response. Note that errors and the result of comparison are all returned in the same construct.

Note that some directory systems may establish access controls which permit the values of certain attributes (such as userPassword) to be compared but not read. In a search result, it may be that an attribute of that type would be returned, but with an empty set of values.

4.11. Abandon Operation

The function of the Abandon Operation is to allow a client to request that the server abandon an outstanding operation. The Abandon Request is defined as follows:

```
AbandonRequest ::= [APPLICATION 16] MessageID
```

The MessageID MUST be that of a an operation which was requested earlier in this connection.

(The abandon request itself has its own message id. This is distinct from the id of the earlier operation being abandoned.)

There is no response defined in the Abandon Operation. Upon transmission of an Abandon Operation, a client may expect that the operation identified by the Message ID in the Abandon Request has been abandoned. In the event that a server receives an Abandon Request on a Search Operation in the midst of transmitting responses to the search, that server MUST cease transmitting entry responses to the abandoned request immediately, and MUST NOT send the SearchResponseDone. Of course, the server MUST ensure that only properly encoded LDAPMessage PDUs are transmitted.

Clients MUST NOT send abandon requests for the same operation multiple times, and MUST also be prepared to receive results from operations it has abandoned (since these may have been in transit when the abandon was requested).

Servers MUST discard abandon requests for message IDs they do not recognize, for operations which cannot be abandoned, and for operations which have already been abandoned.

4.12. Extended Operation

An extension mechanism has been added in this version of LDAP, in order to allow additional operations to be defined for services not available elsewhere in this protocol, for instance digitally signed operations and results.

The extended operation allows clients to make requests and receive responses with predefined syntaxes and semantics. These may be defined in RFCs or be private to particular implementations. Each request MUST have a unique OBJECT IDENTIFIER assigned to it.

```
ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
        requestName      [0] LDAPOID,
        requestValue     [1] OCTET STRING OPTIONAL }
```

The requestName is a dotted-decimal representation of the OBJECT IDENTIFIER corresponding to the request. The requestValue is information in a form defined by that request, encapsulated inside an OCTET STRING.

The server will respond to this with an LDAPMessage containing the ExtendedResponse.

```
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
        COMPONENTS OF LDAPResult,
        responseName     [10] LDAPOID OPTIONAL,
        response         [11] OCTET STRING OPTIONAL }
```

If the server does not recognize the request name, it MUST return only the response fields from LDAPResult, containing the protocolError result code.

5. Protocol Element Encodings and Transfer

One underlying service is defined here. Clients and servers SHOULD

implement the mapping of LDAP over TCP described in 5.2.1.

5.1. Mapping Onto BER-based Transport Services

The protocol elements of LDAP are encoded for exchange using the Basic Encoding Rules (BER) [11] of ASN.1 [3]. However, due to the high overhead involved in using certain elements of the BER, the following additional restrictions are placed on BER-encodings of LDAP protocol elements:

(1) Only the definite form of length encoding will be used.

(2) OCTET STRING values will be encoded in the primitive form only.

(3) If the value of a BOOLEAN type is true, the encoding MUST have its contents octets set to hex "FF".

(4) If a value of a type is its default value, it MUST be absent. Only some BOOLEAN and INTEGER types have default values in this protocol definition.

These restrictions do not apply to ASN.1 types encapsulated inside of OCTET STRING values, such as attribute values, unless otherwise noted.

5.2. Transfer Protocols

This protocol is designed to run over connection-oriented, reliable transports, with all 8 bits in an octet being significant in the data stream.

5.2.1. Transmission Control Protocol (TCP)

The LDAPMessage PDUs are mapped directly onto the TCP bytestream. It is recommended that server implementations running over the TCP MAY provide a protocol listener on the assigned port, 389. Servers may instead provide a listener on a different port number. Clients MUST support contacting servers on any valid TCP port.

6. Implementation Guidelines

This document describes an Internet protocol.

6.1. Server Implementations

The server MUST be capable of recognizing all the mandatory attribute type names and implement the syntaxes specified in [5]. Servers MAY also recognize additional attribute type names.

6.2. Client Implementations

Clients which request referrals MUST ensure that they do not loop between servers. They MUST NOT repeatedly contact the same server for the same request with the same target entry name, scope and filter. Some clients may be using a counter that is incremented each time referral handling occurs for an operation, and these kinds of clients MUST be able to handle a DIT with at least ten layers of naming contexts between the root and a leaf entry.

In the absence of prior agreements with servers, clients SHOULD NOT assume that servers support any particular schemas beyond those referenced in section 6.1. Different schemas can have different attribute types with the same names. The client can retrieve the subschema entries referenced by the subschemaSubentry attribute in the server's root DSE or in entries held by the server.

7. Security Considerations

When used with a connection-oriented transport, this version of the protocol provides facilities for the LDAP v2 authentication mechanism, simple authentication using a cleartext password, as well as any SASL mechanism [12]. SASL allows for integrity and privacy services to be negotiated.

It is also permitted that the server can return its credentials to the client, if it chooses to do so.

Use of cleartext password is strongly discouraged where the underlying transport service cannot guarantee confidentiality and may result in disclosure of the password to unauthorized parties.

When used with SASL, it should be noted that the name field of the BindRequest is not protected against modification. Thus if the distinguished name of the client (an LDAPDN) is agreed through the negotiation of the credentials, it takes precedence over any value in the unprotected name field.

Implementations which cache attributes and entries obtained via LDAP MUST ensure that access controls are maintained if that information is to be provided to multiple clients, since servers may have access control policies which prevent the return of entries or attributes in search results except to particular authenticated clients. For example, caches could serve result information only to the client whose request caused it to be cache.

8. Acknowledgements

This document is an update to RFC 1777, by Wengyik Yeong, Tim Howes, and Steve Kille. Design ideas included in this document are based on those discussed in ASID and other IETF Working Groups. The contributions of individuals in these working groups is gratefully acknowledged.

9. Bibliography

[1] ITU-T Rec. X.500, "The Directory: Overview of Concepts, Models and Service", 1993.

[2] Yeong, W., Howes, T., and S. Kille, "Lightweight Directory Access Protocol", RFC 1777, March 1995.

[3] ITU-T Rec. X.680, "Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation", 1994.

[4] Kille, S., Wahl, M., and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[5] Wahl, M., Coulbeck, A., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[6] ITU-T Rec. X.501, "The Directory: Models", 1993.

[7] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[8] ITU-T Rec. X.511, "The Directory: Abstract Service Definition", 1993.

[9] Howes, T., and M. Smith, "The LDAP URL Format", RFC 2255, December 1997.

[10] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[11] ITU-T Rec. X.690, "Specification of ASN.1 encoding rules: Basic, Canonical, and Distinguished Encoding Rules", 1994.

[12] Meyers, J., "Simple Authentication and Security Layer", RFC 2222, October 1997.

[13] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1 : 1993.

[14] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, October 1996.

10. Authors' Addresses

Mark Wahl
Critical Angle Inc.
4815 W Braker Lane #502-385
Austin, TX 78759
USA

Phone: +1 512 372-3160
EMail: M.Wahl@critical-angle.com

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Rd., MS MV068
Mountain View, CA 94043
USA

Phone: +1 650 937-3419
EMail: howes@netscape.com

Steve Kille
Isode Limited
The Dome, The Square
Richmond
TW9 1DT
UK

Phone: +44-181-332-9091
EMail: S.Kille@isode.com

Appendix A - Complete ASN.1 Definition

```
        Lightweight-Directory-Access-Protocol-V3 DEFINITIONS
        IMPLICIT TAGS ::=

        BEGIN

        LDAPMessage ::= SEQUENCE {
                messageID       MessageID,
                protocolOp      CHOICE {
                        bindRequest     BindRequest,
                        bindResponse    BindResponse,
                        unbindRequest   UnbindRequest,
                        searchRequest   SearchRequest,
                        searchResEntry  SearchResultEntry,
                        searchResDone   SearchResultDone,
                        searchResRef    SearchResultReference,
                        modifyRequest   ModifyRequest,
                        modifyResponse  ModifyResponse,
                        addRequest      AddRequest,
                        addResponse     AddResponse,
                        delRequest      DelRequest,
                        delResponse     DelResponse,
                        modDNRequest    ModifyDNRequest,
                        modDNResponse   ModifyDNResponse,
                        compareRequest  CompareRequest,
                        compareResponse CompareResponse,
                        abandonRequest  AbandonRequest,
                        extendedReq     ExtendedRequest,
                        extendedResp    ExtendedResponse },
                 controls       [0] Controls OPTIONAL }

        MessageID ::= INTEGER (0 .. maxInt)

        maxInt INTEGER ::= 2147483647 -- (2^^31 - 1) --

        LDAPString ::= OCTET STRING

        LDAPOID ::= OCTET STRING

        LDAPDN ::= LDAPString

        RelativeLDAPDN ::= LDAPString

        AttributeType ::= LDAPString

        AttributeDescription ::= LDAPString
```

```
        AttributeDescriptionList ::= SEQUENCE OF
                AttributeDescription

        AttributeValue ::= OCTET STRING

        AttributeValueAssertion ::= SEQUENCE {
                attributeDesc   AttributeDescription,
                assertionValue  AssertionValue }

        AssertionValue ::= OCTET STRING

        Attribute ::= SEQUENCE {
                type    AttributeDescription,
                vals    SET OF AttributeValue }

        MatchingRuleId ::= LDAPString

        LDAPResult ::= SEQUENCE {
                resultCode      ENUMERATED {
                                success                      (0),
                                operationsError              (1),
                                protocolError                (2),
                                timeLimitExceeded            (3),
                                sizeLimitExceeded            (4),
                                compareFalse                 (5),
                                compareTrue                  (6),
                                authMethodNotSupported       (7),
                                strongAuthRequired           (8),
                                        -- 9 reserved --
                                referral                     (10),  -- new
                                adminLimitExceeded           (11),  -- new
                                unavailableCriticalExtension (12),  -- new
                                confidentialityRequired      (13),  -- new
                                saslBindInProgress           (14),  -- new
                                noSuchAttribute              (16),
                                undefinedAttributeType       (17),
                                inappropriateMatching        (18),
                                constraintViolation          (19),
                                attributeOrValueExists       (20),
                                invalidAttributeSyntax       (21),
                                        -- 22-31 unused --
                                noSuchObject                 (32),
                                aliasProblem                 (33),
                                invalidDNSyntax              (34),
                                -- 35 reserved for undefined isLeaf --
                                aliasDereferencingProblem    (36),
                                        -- 37-47 unused --
                                inappropriateAuthentication  (48),
```



```
                                invalidCredentials           (49),
                                insufficientAccessRights     (50),
                                busy                         (51),
                                unavailable                  (52),
                                unwillingToPerform           (53),
                                loopDetect                   (54),
                                        -- 55-63 unused --
                                namingViolation              (64),
```

```
                        objectClassViolation        (65),
                        notAllowedOnNonLeaf         (66),
                        notAllowedOnRDN             (67),
                        entryAlreadyExists          (68),
                        objectClassModsProhibited   (69),
                                -- 70 reserved for CLDAP --
                        affectsMultipleDSAs         (71), -- new
                                -- 72-79 unused --
                        other                       (80) },
                        -- 81-90 reserved for APIs --
                matchedDN       LDAPDN,
                errorMessage    LDAPString,
                referral        [3] Referral OPTIONAL }

        Referral ::= SEQUENCE OF LDAPURL

        LDAPURL ::= LDAPString -- limited to characters permitted in URLs

        Controls ::= SEQUENCE OF Control

        Control ::= SEQUENCE {
                controlType             LDAPOID,
                criticality             BOOLEAN DEFAULT FALSE,
                controlValue            OCTET STRING OPTIONAL }

        BindRequest ::= [APPLICATION 0] SEQUENCE {
                version                 INTEGER (1 .. 127),
                name                    LDAPDN,
                authentication          AuthenticationChoice }

        AuthenticationChoice ::= CHOICE {
                simple                  [0] OCTET STRING,
                                         -- 1 and 2 reserved
                sasl                    [3] SaslCredentials }

        SaslCredentials ::= SEQUENCE {
                mechanism               LDAPString,
                credentials             OCTET STRING OPTIONAL }

        BindResponse ::= [APPLICATION 1] SEQUENCE {
```



```
             COMPONENTS OF LDAPResult,
             serverSaslCreds    [7] OCTET STRING OPTIONAL }

        UnbindRequest ::= [APPLICATION 2] NULL

        SearchRequest ::= [APPLICATION 3] SEQUENCE {
                baseObject      LDAPDN,
                scope           ENUMERATED {
                        baseObject              (0),
                        singleLevel             (1),
                        wholeSubtree            (2) },
                derefAliases    ENUMERATED {
                        neverDerefAliases       (0),
                        derefInSearching        (1),
                        derefFindingBaseObj     (2),
                        derefAlways             (3) },
                sizeLimit       INTEGER (0 .. maxInt),
                timeLimit       INTEGER (0 .. maxInt),
                typesOnly       BOOLEAN,
                filter          Filter,
```

```
        attributes      AttributeDescriptionList }

Filter ::= CHOICE {
        and             [0] SET OF Filter,
        or              [1] SET OF Filter,
        not             [2] Filter,
        equalityMatch   [3] AttributeValueAssertion,
        substrings      [4] SubstringFilter,
        greaterOrEqual  [5] AttributeValueAssertion,
        lessOrEqual     [6] AttributeValueAssertion,
        present         [7] AttributeDescription,
        approxMatch     [8] AttributeValueAssertion,
        extensibleMatch [9] MatchingRuleAssertion }

SubstringFilter ::= SEQUENCE {
        type            AttributeDescription,
        -- at least one must be present
        substrings      SEQUENCE OF CHOICE {
                initial [0] LDAPString,
                any     [1] LDAPString,
                final   [2] LDAPString } }

MatchingRuleAssertion ::= SEQUENCE {
        matchingRule    [1] MatchingRuleId OPTIONAL,
        type            [2] AttributeDescription OPTIONAL,
        matchValue      [3] AssertionValue,
        dnAttributes    [4] BOOLEAN DEFAULT FALSE }
```



```
SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
        objectName      LDAPDN,
        attributes      PartialAttributeList }

PartialAttributeList ::= SEQUENCE OF SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }

SearchResultReference ::= [APPLICATION 19] SEQUENCE OF LDAPURL

SearchResultDone ::= [APPLICATION 5] LDAPResult

ModifyRequest ::= [APPLICATION 6] SEQUENCE {
        object          LDAPDN,
        modification    SEQUENCE OF SEQUENCE {
                operation       ENUMERATED {
                                        add     (0),
                                        delete  (1),
                                        replace (2) },
                modification    AttributeTypeAndValues } }

AttributeTypeAndValues ::= SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }

ModifyResponse ::= [APPLICATION 7] LDAPResult

AddRequest ::= [APPLICATION 8] SEQUENCE {
        entry           LDAPDN,
        attributes      AttributeList }

AttributeList ::= SEQUENCE OF SEQUENCE {
```

```
                type    AttributeDescription,
                vals    SET OF AttributeValue }

        AddResponse ::= [APPLICATION 9] LDAPResult

        DelRequest ::= [APPLICATION 10] LDAPDN

        DelResponse ::= [APPLICATION 11] LDAPResult

        ModifyDNRequest ::= [APPLICATION 12] SEQUENCE {
                entry           LDAPDN,
                newrdn          RelativeLDAPDN,
                deleteoldrdn    BOOLEAN,
                newSuperior     [0] LDAPDN OPTIONAL }

        ModifyDNResponse ::= [APPLICATION 13] LDAPResult
```



```
        CompareRequest ::= [APPLICATION 14] SEQUENCE {
                entry           LDAPDN,
                ava             AttributeValueAssertion }

        CompareResponse ::= [APPLICATION 15] LDAPResult

        AbandonRequest ::= [APPLICATION 16] MessageID

        ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
                requestName      [0] LDAPOID,
                requestValue     [1] OCTET STRING OPTIONAL }

        ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
                COMPONENTS OF LDAPResult,
                responseName     [10] LDAPOID OPTIONAL,
                response         [11] OCTET STRING OPTIONAL }

        END
```


Full Copyright Statement