ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Χρήση της τεχνολογίας των Web Services για την υποστήριξη επεξεργασίας εγγραφών σε υπηρεσίες καταλόγου (LDAP)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος, Βλ. Καλευράς

Επιβλέπων : Βασίλης Μάγκλαρης, Δημήτριος, Βλ. Καλογεράς
Καθηγητής, Ερευνητής Β

Αθήνα, Σεπτέμβριος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Χρήση της τεχνολογίας των Web Services για την υποστήριξη επεξεργασίας εγγραφών σε υπηρεσίες καταλόγου (LDAP)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Βλ. Καλευράς

Επιβλέπων : Βασίλης Μάγκλαρης, Δημήτριος Βλ. Καλογεράς
Καθηγητής, Ερευνητής Β

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ……..

.....	.....	.....
Βασίλης Μάγκλαρης	Συμεών Παπαβασιλείου	Δημήτρης Καλογεράς
Καθηγητής	Αναπληρωτής Καθηγητής	Ερευνητής Β

Αθήνα, Σεπτέμβριος 2013

.....
Κωνσταντίνος Βλ. Καλευράς

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της διπλωματικής εργασίας είναι η μελέτη, ανάπτυξη και λειτουργία framework διαστρωματωμένης πραγματοποίησης αλλαγών σε δεδομένα εξυπηρετητή υπηρεσιών καταλόγου (directory server) με τη χρήση τεχνολογιών Web Services. Η υπηρεσία αυτή έχει σα βασικό ρόλο τη δημιουργία, διατήρηση, ενημέρωση και διαγραφή πληροφοριών (διευθύνσεις ηλεκτρονικού ταχυδρομείου, τηλέφωνα επικοινωνίας, κωδικοί πρόσβασης υπηρεσιών, κ.λπ.) για τους πολίτες που σχετίζονται άμεσα ή έμμεσα με τα Ακαδημαϊκά Ιδρύματα της χώρας. Αυτές θα μπορούν να χρησιμοποιηθούν για την ταυτοποίηση και την εξουσιοδότηση των χρηστών σε κεντρικά υπολογιστικά συστήματα, αλλά και στη διάθεση σε αυτούς συγκεκριμένων δικτυακών υπηρεσιών.

Στόχος της εργασίας είναι η παρουσίαση τρόπου δημιουργίας μιας δικτυακής εφαρμογής γραμμένης σε PHP η οποία μπορεί να επικοινωνεί με την υπηρεσία καταλόγου μέσω Web Services (πρωτόκολλο SOAP over HTTP). Η εφαρμογή περιλαμβάνει πολλαπλά στρώματα (layers) επικοινωνίας και λειτουργίας, υιοθετώντας τη λογική πολλών πρωτοκόλλων επικοινωνιών (με πρώτο το πρωτόκολλο TCP/IP και γενικότερα τη διαστρωμάτωση Layer-1 ως Layer-7 των δικτύων Η/Υ).

Για το σκοπό αυτό χρησιμοποιήθηκε πληθώρα προγραμματιστικών εργαλείων τα περισσότερα των οποίων προσφέρονται δωρεάν από διάφορους φορείς του Διαδικτύου και αποτελούν καθιερωμένα εργαλεία τα οποία χρησιμοποιούνται από πολλές επιχειρήσεις που δραστηριοποιούνται στον τομέα των δικτυακών εφαρμογών. Επιπλέον, η πλειοψηφία των προγραμμάτων αυτών είναι ανοικτού λογισμικού και δίνεται ως συνέπεια η ευκαιρία στο χρήστη να προσθέσει λειτουργίες σε αυτά και να τα ταιριάξει στην εφαρμογή του. Παράλληλα, για τις ανάγκες της διπλωματικής αναπτύχθηκε ένα ισχυρό, παραμετροποιήσιμο middleware σε PHP, με πολλές δυνατότητες και εύκολο στη χρήση.

Λέξεις κλειδιά: *web services, δικτυακή υπηρεσία, δικτυακή εφαρμογή, πρωτόκολλο SOAP, PHP, nuSOAP, LDAP, εξυπηρετητής εφαρμογών, δικτυακός εξυπηρετητής, εξυπηρετητής βάσεων δεδομένων.*

Abstract

The target of the current essay is to study, develop and operate a multi-layered framework used to perform changes on data held into directory servers. The framework makes heavy use of Web Services technology. The service has a major role in helping create, update and remove personal and authorization information (e-mail addresses, telephone numbers, passwords) for users using services provided by Greek universities. This information will be used to authenticate and authorize users to be able to use various Web-based services as well as other core user services (such as e-mail).

In order to achieve the above, the document will provide, in a detailed manner, a description of how to develop a web-based application, written in PHP, which can communicate and perform write functions on a directory service (LDAP) using Web Services (SOAP protocol over HTTP). The application will include multiple layers in its communication and processing functions, using the approach pioneered by several communication protocols (among others, the TCP/IP protocol).

To this end, the author has used freely available, open source program tools. Additionally, a feature rich, configurable middleware was developed (written in PHP), to provide the main API used by the web services for communicated with the directory service (and abstracting most of the needed, yet reusable functionality.

Key Words: *web services, SOAP protocol, PHP, nuSOAP, LDAP.*

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Στόχος της παρούσας εργασίας είναι ο σχεδιασμός ενός κατανεμημένου συστήματος παροχής υπηρεσιών και ανταλλαγής δεδομένων βασισμένου στο πρωτόκολλο SOAP πάνω από το πρωτόκολλο HTTP, παρέχοντας έτσι συμβατότητα με την πλειοψηφία των αντίστοιχων πακέτων λογισμικού (.Net FrameWork, Java κτλ).Ο ορισμός της διεπαφής έγινε με χρήση γλώσσας προδιαγραφής Web Services WSDL (Web Service Description Language) την XML γλώσσα δηλαδή που έχει προδιαγραφεί από το W3 Consortium για την υλοποίηση και περιγραφή web services.
. Οι διαδικτυακές εφαρμογές σήμερα χρησιμοποιούνται κυρίως στον εμπορικό τομέα ενώ στην παρούσα εργασία θα παρουσιαστεί η εφαρμογή τους στις υπηρεσίες εκπαίδευσης.

1.2 Η εξέλιξη των πρωτοκόλλων καταλόγων

1.2.1 DAP (Directory Access Protocol) –X.500

Το Directory Access Protocol (DAP) είναι ένα πρότυπο δικτύωσης υπολογιστών που εκδόθηκε από την ITU-T και το πρότυπο ISO το 1988 για την πρόσβαση σε ένα X.500 directory service. Το πρωτόκολλο DAP επρόκειτο να χρησιμοποιηθεί από τα συστήματα υπολογιστή-πελάτη, αλλά δεν έγινε ποτέ δημοφιλές αφού υπήρχαν λίγες εφαρμογές που χρησιμοποιούσαν την πλήρη στοίβα πρωτοκόλλου του OSI για επιτραπέζιους υπολογιστές και συνάμα δεν διέθεταν το κατάλληλο υλικό και τα λειτουργικά συστήματα για να λειτουργήσουν με αυτό. Οι βασικές λειτουργίες του πρωτοκόλλου DAP: Read, List, Search, Compare, Modify, Add, Delete και ModifyRDN, έπειτα προσαρμόστηκαν στο Novell Directory Services (NDS) και στο Lightweight Directory Access Protocol (LDAP).

Το πρότυπο X.500 που προτάθηκε από τον οργανισμό ISO (International Standards Organisation) ήταν αυτό που προτάθηκε αρχικά και χρησιμοποιήθηκε για την καθιέρωση μιας παγκόσμιας ονοματολογίας καταλόγου. Καθιέρωσε ένα ανοιχτό πρωτόκολλο επικοινωνίας, το Directory Access Protocol (DAP) το οποίο μπορούσε να χρησιμοποιηθεί από οποιαδήποτε εφαρμογή για την πρόσβαση στον κατάλογο. Το X.500 είχε αρκετά πλεονεκτήματα, όπως για παράδειγμα ότι υποστήριζε τη δημιουργία ενός επεκτάσιμου σχήματος στον κατάλογο ανάλογα με τις ανάγκες, και στο οποίο μπορούσε να αποθηκευτεί οποιαδήποτε μορφή πληροφορίας. Το κύριο μειονέκτημά του όμως, ήταν ότι προσέφερε ένα επικοινωνιακό πρωτόκολλο το οποίο δεν βασιζόταν στο πρωτόκολλο TCP/IP, ενώ επιπλέον παρουσίαζε σημαντική πολυπλοκότητα στη διαχείριση της ονοματολογίας του καταλόγου.
Αναλυτικά το X.500 είναι μια σειρά από πρότυπα δικτύωσης ηλεκτρονικών υπολογιστών που καλύπτουν υπηρεσίες καταλόγου. Η σειρά X.500 αναπτύχθηκε από την ITU-T, παλαιότερα γνωστή ως CCITT. Οι υπηρεσίες καταλόγου αναπτύχθηκαν με σκοπό να υποστηρίξουν τις απαιτήσεις του X.400 (την ηλεκτρονική ανταλλαγή αλληλογραφίας).
Τα πρωτόκολλα που ορίζονται από το X.500 περιλαμβάνουν

- DAP (Directory Access Protocol)
- DSP (Directory System Protocol)
- DISP (Directory Information Shadowing Protocol)
- DOP (Directory Operational Bindings Management Protocol)

Επειδή αυτά τα πρωτόκολλα χρησιμοποιούσαν το πρωτόκολλο δικτύωσης της OSI στοίβας, αναπτύχθηκαν μια σειρά εναλλακτικών λύσεων για το DAP με σκοπό να επιτρέψουν στους πελάτες να έχουν πρόσβαση στο X.500 κατάλογο, χρησιμοποιώντας το πρωτόκολλο TCP / IP. Η πιο γνωστή εναλλακτική λύση για το DAP είναι το Lightweight Directory Access Protocol (LDAP). Τώρα το DAP και τα υπόλοιπα πρωτόκολλα X.500 μπορούν πλέον να χρησιμοποιούν το πρωτόκολλο TCP / IP αλλά το LDAP παραμένει το κύριο και δημοφιλές πρωτόκολλο πρόσβασης καταλόγου.

1.2.2 LDAP (Lightweight Directory Access Protocol)

Το πρωτόκολλο LDAP (Lightweight Directory Access Protocol) που καθιερώθηκε στη συνέχεια, διατήρησε τα θετικά στοιχεία του X.500, ενώ ταυτόχρονα παρείχε ένα επικοινωνιακό πρωτόκολλο βασισμένο στο TCP/IP. Προτάθηκε αρχικά το 1996 από την Netscape Communications Corp., το Πανεπιστήμιο του Michigan και περισσότερες από 40 εταιρίες, ως ένα ανοιχτό πρότυπο για τις Υπηρεσίες Καταλόγου στο Internet. Ο όρος Lightweight στην ονομασία του, σημαίνει ότι συγκρινόμενο με το DAP περιέχει λιγότερο κώδικα και πιο απλές συναρτήσεις, και συνεπώς είναι για τους κατασκευαστές πιο εύκολα υλοποιήσιμο και μικρότερο σε κόστος.
Πιο συγκεκριμένα το Lightweight Directory Access Protocol (LDAP) είναι ένα πρωτόκολλο της υπηρεσίας καταλόγου που τρέχει πάνω ακριβώς από την στοίβα TCP / IP. Το μοντέλο των πληροφοριών (τόσο για τα δεδομένα όσο και για τα σύνολα των χαρακτηριστικών, namespaces) του LDAP είναι παρόμοια με εκείνη της υπηρεσίας X.500 του OSI καταλόγου, αλλά με λιγότερες δυνατότητες και χαμηλότερες απαιτήσεις πόρων από το X.500. Το LDAP API χρησιμοποιείται για την διαχείριση του καταλόγου.

2 Επισκόπηση της τεχνολογίας LDAP

2.1 Κατάλογοι

Οι κατάλογοι είναι παρόμοιοι με τις βάσεις δεδομένων, μόνο που περιέχουν περισσότερο περιγραφική πληροφορία. Τα περιεχόμενα τους στηρίζονται στην έννοια της κλάσης (class), της εγγραφής (entry) και των χαρακτηριστικών (attributes). Η οργάνωση των πληροφοριών ακολουθεί το ιεραρχικό (σε αντίθεση με το σχεσιακό) μοντέλο. Η πληροφορία είναι οργανωμένη σε δεντρική δομή με τα δεδομένα να περιέχονται σε εγγραφές στα φύλλα του δέντρου πληροφοριών. Κάθε εγγραφή είναι πλήρης και περιέχει το σύνολο των δεδομένων που αντιστοιχούν σε αυτή (σε αντίθεση με τις σχεσιακές βάσεις δεδομένων όπου τα στοιχεία είναι διασκορπισμένα ανάμεσα σε πολλαπλούς πίνακες δεδομένων). Το ‘όνομα’ της κάθε εγγραφής λέγεται Distinguished Name (DN) και αποτυπώνει τη θέση της εγγραφής στο δέντρο

πληροφοριών (πχ uid=<username>,ou=people,dc=ntua,dc=gr) Οι πληροφορίες που περιέχει ο κατάλογος τείνουν να γίνουν συνεχώς περισσότερες. Αρχικά περιείχαν τις εγγραφές φυσικών προσώπων με τα χαρακτηριστικά (στοιχεία) τους, χαρακτήρας που προφανώς κληρονομήθηκε από προγενέστερες μορφές και διαδικασίες αποθήκευσης (π.χ. τηλεφωνικοί κατάλογοι). Η τάση όμως είναι να ενσωματώνουν όσο το δυνατό περισσότερες πληροφορίες, τόσο για φυσικά πρόσωπα (με ολοκληρωμένα προφίλ και πληροφορία προσαρμοσμένη σε ειδικές ανάγκες) περιλαμβανομένων πληροφοριών πιστοποίησης και στοιχείων δικτυακών υπηρεσιών, όσο και για οτιδήποτε αφορά ένα υπολογιστικό σύστημα ή ένα δίκτυο (συσκευές, εφαρμογές). Κατ' αυτόν τον τρόπο αφενός είναι προσπελάσιμες οι πληροφορίες από όσους ενδιαφέρονται, αφετέρου μπορεί εύκολα να γίνεται κεντρικά η διαχείριση των πληροφοριών αυτών και των δικαιωμάτων πρόσβασης.

Είναι προφανές ότι όσο μεγαλύτερο είναι το δίκτυο τόσο μεγαλύτερη γίνεται και η ανάγκη για τη χρήση καταλόγων. Και αυτό γιατί είναι περισσότεροι οι πόροι και τα άτομα που χρειάζεται κανείς να προσπελάσει και να εντοπίσει. Επίσης είναι μεγαλύτερη η ανάγκη για προτυποποιημένες (standardized) μεθόδους προσπέλασης και επικοινωνίας των καταλόγων αυτών.

Έχοντας υπόψη την ανάγκη για ένα παγκόσμιο κατάλογο όπου κάθε στοιχείο του θα έχει και ένα μοναδικό όνομα (όπως κάθε υπολογιστής έχει μια μοναδική διεύθυνση) έπρεπε να βρεθεί μια ονοματολογία που θα ικανοποιούσε το σκοπό αυτό. Με βάση παρόμοιες περιπτώσεις σχεδιάστηκε μια ιεραρχική ονοματολογία που θα αντανακλούσε την ιεραρχική οργάνωση των ευρετηρίων. Η ονοματολογία αυτή προτυποποιήθηκε μέσω του πρωτοκόλλου X.500-DAP (Directory Access Protocol) που σχεδιάστηκε από τον ISO (International Standards Organisation) και ενσωματώθηκε στο πιο πρόσφατο LDAP (Lightweight Directory Access Protocol) που σχεδιάστηκε από την Internet κοινότητα και όπως γίνεται σε ανάλογες περιπτώσεις προτάθηκε μέσω των RFCs (Request for Comments).

Το πρωτόκολλο LDAP ορίζεται από τα παρακάτω αρχεία Request For Comments [9]:

- RFC 4510 - Lighweight Directory Access Protocol (LDAP) Technical Specification Roadmap (replaced the previous LDAP Technical specification, RFC 3377, in its entirety)

- RFC 4511 - Lightweight Directory Access Protocol (LDAP): The Protocol
- RFC 4512 - Lightweight Directory Access Protocol (LDAP): Directory Information Models
- RFC 4513 - Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms
- RFC 4514 - Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names
- RFC 4515 - Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters
- RFC 4516 - Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator RFC 4517 - Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules
- RFC 4518 - Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation
- RFC 4519 - Lightweight Directory Access Protocol (LDAP): Schema for User Applications

Οι LDAPv3 extensions ορίζονται από τα παρακάτω αρχεία Request For Comments:

- RFC 2247 - Use of DNS domains in distinguished names
- RFC 2307 - Using LDAP as a Network Information Service
- RFC 2589 - LDAPv3: Dynamic Directory Services Extensions
- RFC 2649 - LDAPv3 Operational Signatures
- RFC 2696 - LDAP Simple Paged Result Control
- RFC 2798 - inetOrgPerson LDAP Object Class
- RFC 2849 - The LDAP Data Interchange Format (LDIF)
- RFC 2891 - Server Side Sorting of Search Results
- RFC 3045 - Storing Vendor Information in the LDAP root DSE
- RFC 3062 - LDAP Password Modify Extended Operation
- RFC 3296 - Named Subordinate References in LDAP Directories
- RFC 3671 - Collective Attributes in LDAP
- RFC 3672 - Subentries in LDAP
- RFC 3673 - LDAPv3: All Operational Attributes
- RFC 3687 - LDAP Component Matching Rules
- RFC 3698 - LDAP: Additional Matching Rules

- RFC 3829 - LDAP Authorization Identity Controls
- RFC 3866 - Language Tags and Ranges in LDAP
- RFC 3909 - LDAP Cancel Operation
- RFC 3928 - LDAP Client Update Protocol
- RFC 4370 - LDAP Proxied Authorization Control
- RFC 4373 – LBURP
- RFC 4403 - LDAP Schema for UDDI
- RFC 4522 - LDAP: Binary Encoding Option
- RFC 4523 - LDAP: X.509 Certificate Schema
- RFC 4524 - LDAP: COSINE Schema (replaces RFC 1274)
- RFC 4525 - LDAP: Modify-Increment Extension
- RFC 4526 - LDAP: Absolute True and False Filters
- RFC 4527 - LDAP: Read Entry Controls
- RFC 4528 - LDAP: Assertion Control
- RFC 4529 - LDAP: Requesting Attributes by Object Class
- RFC 4530 - LDAP: entryUUID
- RFC 4531 - LDAP Turn Operation
- RFC 4532 - LDAP Who am I? Operation
- RFC 4533 - LDAP Content Sync OperationLDAPv2 was specified in the following RFCs:
    - RFC 1777 - Lightweight Directory Access Protocol (replaced RFC 1487)
    - RFC 1778 - The String Representation of Standard Attribute Syntaxes (replaced RFC 1488)
    - RFC 1779 - A String Representation of Distinguished Names (replaced RFC 1485)

Σχετικά RFCs:

- RFC 2254 - The String Representation of LDAP Search Filters
- RFC 4520 (also BCP 64) - Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP) (replaced RFC 3383)
- RFC 4521 (also BCP 118) - Considerations for Lightweight Directory Access Protocol (LDAP) Extensions

Το πρωτόκολλο LDAP είναι ένα ασφαλές πρωτόκολλο, το οποίο χρησιμοποιεί τον έλεγχο ταυτότητας για να διασφαλίσει ότι η μεταφορά των δεδομένων είναι ασφαλής. Ο έλεγχος αυτός χρησιμοποιείται από τον διακομιστή ο οποίος ελέγχει την ταυτότητα του πελάτη. Η τρέχουσα έκδοση 3 του LDAP χρησιμοποιεί το SASL (Simple Authentication and Security Layer -SASL), χαρακτηριστικό το οποίο παρέχει μια ευελιξία στην επιλογή του μηχανισμού ελέγχου ταυτότητας. Παράλληλα, το πρωτόκολλο Secure Socket Layer (SSL), είναι το πιο δημοφιλές για το σκοπό της προστασίας του καναλιού επικοινωνίας με την υπηρεσία καταλόγου, παρέχοντας ένα υψηλό επίπεδο ασφάλειας.

2.2 Επισκόπηση της λειτουργίας του Πρωτοκόλλου LDAP

Ένας πελάτης αρχίζει μια συνεδρία LDAP όταν συνδέεται στον εξυπηρετητή στην προεπιλεγμένη πόρτα TCP 389. Ο πελάτης στέλνει ένα αίτημα λειτουργίας στον εξυπηρετητή, και με την σειρά του ο εξυπηρετητής απαντάει. Αυτή η διαδικασία δεν είναι απαραίτητο να είναι σειριακή καθώς πέραν μερικών εξαιρέσεων ο πελάτης δεν χρειάζεται να περιμένει για μια απάντηση συγκεκριμένη από τον εξυπηρετητή αλλά και οι απαντήσεις του εξυπηρετητή μπορούν να στέλνονται με οποιαδήποτε σειρά.

Το πρωτόκολλο LDAP συγκεκριμένα υποστηρίζει τις παρακάτω διεργασίες:

- Start TLS — χρησιμοποιεί την επέκταση του LDAPv3 Transport Layer Security (TLS) για να δημιουργήσει μια ασφαλή σύνδεση.
- Bind — Πιστοποιεί τη σύνδεση.
- Search — ψάχνει και ανακτά τις εγγραφές του καταλόγου.
- Compare — Ελέγχει εάν μια εγγραφή περιέχει μία δοσμένη τιμής σε συγκεκριμένο attribute.
- Add - Προσθέτει μια νέα εγγραφή
- Delete - Διαγράφει μια εγγραφή
- Modify - Μετατρέπει μια εγγραφή
- Modify Distinguished Name (DN) — μεταφέρει και μετατρέπει τις εγγραφές

- Abandon — εγκαταλείπει μια προηγούμενη αίτηση
- Extended Operation — μια γενική λειτουργία που χρησιμοποιείται για να καθορίσει άλλες λειτουργίες του συστήματος (επεκτασιμότητα)
- Unbind — κλείνει την σύνδεση του πελάτη με τον διακομιστή / εξυπηρετητή

Επιπρόσθετες λειτουργίες μπορούν να χαρακτηριστούν οι "Unsolicited Notifications" που στέλνει ο εξυπηρετητής προς τους πελάτες που δεν καταγράφονται σαν απαντήσεις σε κανένα αίτημα παραδείγματος χάρη πριν κλείσει μια σύνδεση με τον πελάτη λόγω λήξης του χρονικού ορίου.

Μια κοινή μέθοδος για να ασφαλίσουμε την υπηρεσία επικοινωνίας με τον LDAP είναι να χρησιμοποιήσουμε SSL tunnel από τον εξυπηρετητή μέχρι τον πελάτη. Το παραπάνω δηλώνεται στα LDAP URLs και η προεπιλεγμένη πόρτα που χρησιμοποιείται στο SSL είναι η 636. Η χρήση του LDAP πάνω από SSL είναι κοινή στην έκδοση 2 (LDAPv2) αλλά δεν ήταν ποτέ τυποποιημένη σε κάθε επίσημη προδιαγραφή. Αυτή η χρήση υποβιβάστηκε συνολικά μαζί με το LDAPv2, το οποίο επίσημα αποσύρθηκε το 2003.

Γενικά το LDAP καθορίζεται με όρους ASN.1 και τα μηνύματα του πρωτοκόλλου κωδικοποιούνται σε δυαδική μορφή BER. Επίσης χρησιμοποιεί αναπαράσταση βασισμένη σε κείμενο για ένα αριθμό από ASN.1 πεδία και τύπους.

Στις τηλεπικοινωνίες και στα δίκτυα υπολογιστών το Abstract Syntax Notation One (ASN.1) είναι ένα προτυποποιημένο και ευέλικτο σύστημα χαρακτήρων που περιγράφει δομές δεδομένων για αναπαράσταση, κωδικοποίηση, αναμετάδοση και αποκωδικοποίηση των δεδομένων. Αυτό παρέχει ένα σύνολο από τυπικούς κανόνες που περιγράφουν την δομή των αντικειμένων. Παράλληλα οι Basic Encoding Rules (BER) είναι ένα από τα format κωδικοποίησης που καθορίζονται σαν ένα μέρος του ASN.1 στάνταρ όπως καθορίζεται από την ITU στο X.690.

2.3 Οι λειτουργίες του πρωτοκόλλου LDAP

Ο πελάτης δίνει σε κάθε αίτηση ένα Message ID και ο εξυπηρετητής το χρησιμοποιεί στην απάντηση του. Η απάντηση συμπεριλαμβάνει ένα αριθμητικό καταληκτικό κώδικα που δηλώνει επιτυχία, κάποια κατάσταση λάθους και ή κάποια άλλη ιδιαίτερη περίπτωση. Πριν την απάντηση ο εξυπηρετητής μπορεί να στέλνει άλλα μηνύματα με άλλα αποτελέσματα, για παράδειγμα εάν ψάχνει για μια κάποια εγγραφή και βρεθεί η εγγραφή αυτή από την διαδικασία της Εύρεσης (Search) επιστρέφεται από ένα τέτοιο μήνυμα.

2.3.1 StartTLS

Η διαδικασία StartTLS εγκαθιδρύει την «Transport Layer Security» τον απόγονο του SSL στη νέα σύνδεση που δημιουργείται. Κατ' αυτόν τρόπο παρέχεται εμπιστευτικότητα και ακεραιότητα στα δεδομένα που ανταλλάσσονται με τον εξυπηρετητή. Κατά την διάρκεια της διαπραγμάτευσης του TLS ο εξυπηρετητής στέλνει το πιστοποιητικό X.509 για να αποδείξει την ταυτότητά του. Ο πελάτης μπορεί επίσης να αποστείλει και αυτός το πιστοποιητικό της ταυτότητάς του (Client Certificate Authentication).

Οι εξυπηρετητές επίσης συνήθως υποστηρίζουν το πρωτόκολλο «LDAPS» (Secure LDAP, που είναι γνωστό και ως LDAP πάνω από SSL) σε μια ξεχωριστή πόρτα που όπως προαναφέραμε είναι η 636. Το LDAPS διαφέρει από το απλό LDAP σε δύο στοιχεία: 1) κατά την σύνδεση, ο πελάτης και ο εξυπηρετητής εγκαθιδρύουν TLS πριν οποιοδήποτε μήνυμα LDAP μεταφερθεί από το μέσο (στο LDAPS δεν υπάρχει η ενέργεια StartTLS) , και 2) η σύνδεση του LDAPS πρέπει να κλείσει όταν τερματιστεί η TLS.

Το LDAPS χρησιμοποιήθηκε αρχικά με το LDAPv2 γιατί η λειτουργία StartTLS δεν είχε ακόμα καθοριστεί. Τώρα η χρήση του LDAPS έχει προ πολλού εγκαταλειφτεί και όλα τα νέα λογισμικά χρησιμοποιούν την StartTLS.

2.3.2 Bind (authenticate)

Η διαδικασία αυτή πιστοποιεί τον πελάτη στον εξυπηρετητή.

“Simple Bind” Η απλή αυτή διαδικασία μπορεί να στέλνει απλά το DN (Distinguished Name) και τον κωδικό (password) του πελάτη ως απλό κείμενο. Ο εξυπηρετητής απλά ελέγχει τον αποσταλμένο κωδικό έναντι του χαρακτηριστικού userPassword στην συγκεκριμένη καταχώρηση.

“Anonymous Bind” Η ανώνυμη διαδικασία (με κενό DN και password) τοποθετεί την σύνδεση σε ανώνυμη κατάσταση.

“SASL Bind” (Simple Authentication and Security Layer) προσφέρει υπηρεσίες αυθεντικοποίησης με πολλούς διαφορετικούς τρόπους, όπως π.χ. το Kerberos ή η αποστολή του πιστοποιητικού του πελάτη (μέσω TLS – Client Certificate Authentication).

Κατά την διαδικασία Bind επίσης στέλνεται η έκδοση του LDAP, κανονικά οι πελάτες χρησιμοποιούν την έκδοση του LDAPv3. Στο LDAPv3 δεν είναι απαραίτητη η διαδικασία του Bind ενώ στο LDAPv2 είναι προαπαιτούμενη.

2.3.3 Search and Compare

Η διαδικασία αυτή χρησιμοποιείται για την αναζήτηση και ανάγνωση των εγγραφών σε μια βάση LDAP. Οι παράμετροι που χρησιμοποιούνται είναι οι παρακάτω:

baseObject

Σε αυτήν την παράμετρο καθορίζεται το DN (Distinguished Name) της εγγραφής από το οποίο δύναται να ξεκινήσει η αναζήτηση.

scope

Το εύρος (βάθος) της αναζήτησης. Μπορεί να είναι το BaseObject δηλαδή αναζήτηση μόνο στο επίπεδο του baseObject, το singleLevel για την πραγματοποίηση αναζήτησης ένα επίπεδο κάτω του baseObject ή wholeSubtree για την αναζήτηση σε όλο το υποδέντρο κάτω του baseObject.

filter

Το κριτήριο αναζήτησης. Για παράδειγμα θεωρούμε το παρακάτω filter:

(&(objectClass=person)(|(givenName=Mike)(mail=mike*)))

Αυτό θα επιλέξει τα "persons" ως στοιχεία του objectClass person ,
που είτε έχουν το όνομα " Mike " ή ένα e-mail που αρχίζει με την συμβολοσειρά "mike ". Το φίλτρο αναζήτησης με άλλα λόγια επιτρέπει τη χρήση πολλαπλών κριτηρίων αναζήτησης και τη σύνδεση τους με βασικές λογικές πύλες (and, or, not). Το ίδιο το κριτήριο μπορεί να είναι τύπου ισότητας ή μερικής αναζήτησης (χρήση συμβόλου *).

derefAliases

Χρησιμοποιείται όταν απαιτείται να οδηγηθεί σε μια άλλη εγγραφή. (alias) όπου μια εγγραφή δείχνει μια άλλη.

attributes

Ποιες ιδιότητες χρειάζονται να επιστραφούν στις εγγραφές των αποτελεσμάτων.

sizeLimit, timeLimit

Ο μέγιστος αριθμός των εγγραφών που χρειάζεται να επιστραφούν από μια αναζήτηση και ο μέγιστος χρόνος που επιτρέπεται να εκτελεστεί η αναζήτηση στην βάση.

typesOnly

Αυτή η παράμετρος ορίζει να επιστρέφονται μόνο τα ονόματα των ιδιοτήτων (attributes) και όχι οι τιμές τους.

Ο εξυπηρετητής επιστρέφει τις εγγραφές που ταιριάζουν στα κριτήρια αναζήτησης. Αυτές μπορούν να παρουσιαστούν με κάθε σειρά και το τελικό αποτέλεσμα θα περιλαμβάνει τον κώδικα των αποτελεσμάτων.

Η διαδικασία Compare παίρνει ένα DN, ένα όνομα ιδιότητας και μια τιμή ιδιότητας και ελέγχει εάν η ονοματισμένη εγγραφή περιέχει αυτή την ιδιότητα με την συγκεκριμένη τιμή.

2.3.4 Update Data

Add, Delete, Modify και Modify DN

Το Modify παίρνει μια λίστα από χαρακτηριστικά και τα διαγράφει, προσθέτει νέες τιμές ή ακόμα και αντικαθιστά τις τωρινές τιμές με νέες. Παραδείγματος χάριν η λειτουργία Modify DN (move/rename entry) παίρνει το νέο RDN (Relative Distinguished Name) λαμβάνει υπόψη ένα flag που καθορίζει εάν θα διαγράψει ή όχι τις τιμές του παλιού RDN και αναλόγως το αντικαθιστά. Ο εξυπηρετητής μπορεί να υποστηρίζει μετονομασία ολόκληρου υποδέντρου του καταλόγου.

Η Add λειτουργία προσθέτει νέα εγγραφή στον κατάλογο ενώ η Delete αφαιρεί.

2.3.5 Extended operations

Η διαδικασία Extended Operations είναι μια γενική λειτουργία του LDAP που μπορεί να χρησιμοποιηθεί για να καθορίσει νέες διαδικασίες όπως Cancel, το Password Modify και Start TLS.

2.3.6 Abandon

Η διαδικασία Abandon χρησιμοποιείται όταν ο εξυπηρετητής εγκαταλείπει μια διαδικασία που του δίνεται από ένα message ID. Ο εξυπηρετητής δεν χρειάζεται να απαντήσει σε αυτό το μήνυμα και τελικά ούτε η διαδικασία Abandon ούτε η εγκαταλελειμμένη διαδικασία στέλνει μια απάντηση. Αυτό δημιουργεί πρόβλημα και για να καλυφθεί και αυτή η αδυναμία του συστήματος υπάρχει η παρόμοια Extended Operation Cancel η οποία στέλνει απάντηση την οποία όμως δεν υποστηρίζουν όλες οι εκδόσεις των εξυπηρετητών LDAP.

2.3.7 Unbind

Η λειτουργία Unbind εγκαταλείπει όλες τις διαδικασίες που εκτελούνται και έπειτα κλείνει την σύνδεση με τον εξυπηρετητή. Αυτή δεν στέλνει καμία απάντηση και το

όνομα που χρησιμοποιήθηκε εδώ (Unbind) δεν σημαίνει πως είναι η αντίστροφη διαδικασία από το Bind. Απλά οι πελάτες θα μπορούσαν να κλείσουν την σύνδεση τους με τον εξυπηρετητή, αλλά υπό κανονικές συνθήκες χρησιμοποιούν συνολικά την λειτουργία Unbind, η οποία επιτρέπει στον εξυπηρετητή να κλείσει ομαλά την σύνδεση με τον πελάτη και να απελευθερώσει τους πόρους που χρησιμοποιούσε. Αν δεν γίνει αυτό τότε ο εξυπηρετητής κρατάει τους πόρους αυτούς για ένα εύλογο χρονικό διάστημα που καθορίζεται από τον χρόνο που κάνει ο πελάτης να φανεί πως έχει εγκαταλείψει την σύνδεση. Επίσης είναι σημαντική λειτουργία καθώς με το Unbind στέλνονται οι λειτουργίες cancel σε όποιες διαδικασίες μπορούν να εγκαταλειφθούν και σταματά τον εξυπηρετητή από το να στέλνει απαντήσεις για τις διαδικασίες που δεν μπορούν να διακοπούν.

2.3.8 LDAP Αντιγραφή (replication)

Ένας άλλος παράγοντας που εξασφαλίζει την καλή απόδοση και την αξιοπιστία ενός συστήματος υπηρεσίας καταλόγου είναι η δυνατότητα χρήσης μηχανισμών που βελτιώνουν συνολικά την επίδοση του συστήματος και εξασφαλίζουν την άμεση και αποτελεσματική αντιμετώπιση τυχόν προβλημάτων. Στους διακομιστές LDAP η δυνατότητα αυτή παρέχεται μέσω της δημιουργίας και χρήσης μηχανισμών αντιγραφής (replication). Το κύριο πλεονέκτημα της λειτουργίας αυτής είναι ότι μπορούμε να δημιουργούμε αντίγραφα ενός διακομιστή υπηρεσίας καταλόγου για να εξασφαλίσουμε την ανθεκτικότητα σε σφάλματα, το διαμοιρασμό του φόρτου μέσω της κατανομής των λειτουργιών σε διαφορετικά μηχανήματα και τη μεγαλύτερη ασφάλεια των δεδομένων.

Η πιο απλή μορφή αντιγραφής είναι η single-master αντιγραφή όπου ένας κύριος διακομιστής αποτελεί τον προμηθευτή (supplier) που προωθεί τις αλλαγές σε διακομιστές-αντίγραφα του, τους καταναλωτές (consumers). Κάποια από τα προσφερόμενα προϊόντα, υποστηρίζουν multi-master αντιγραφή όπου ένας διακομιστής μπορεί να είναι ταυτόχρονα προμηθευτής και καταναλωτής και κατά συνέπεια το ίδιο υποδένδρο (subtree) υπάρχει και ενημερώνεται από περισσότερους από έναν διακομιστές. Συνήθως όμως οι αλλαγές πραγματοποιούνται πάντα σε μία replica προκειμένου να μην υπάρξουν περιπτώσεις replication conflicts – πχ προσθήκη εγγραφής σε ένα υποδέντρο σε μία replica ενώ το ίδιο υποδέντρο

διαγράφεται σε μία άλλη replica. Το υποδένδρο αποτελεί ξεχωριστό αντίγραφο ανάγνωσης/γραφής σε καθέναν από τους κύριους διακομιστές. Ο κάθε καταναλωτής διατηρεί το δικό του read-only replica. Οι καταναλωτές δέχονται αλλαγές από τους προμηθευτές και μπορούν να διαθέτουν ορισμένα referrals σχετικά με αυτούς ώστε να ανακατευθύνουν τα αιτήματα που προέρχονται από αυτούς.

Τα σημαντικότερα πλεονεκτήματα που παρουσιάζονται με την υλοποίηση της multi-master αντιγραφής είναι η υποστήριξη (fault tolerance) που μπορεί να παρέχει ο καθένας από τους κύριους διακομιστές στην περίπτωση που υπάρχει κάποιο πρόβλημα με τον άλλο, καθώς επίσης και το γεγονός ότι οι αλλαγές θα γίνονται στον τοπικό διακομιστή του κατανεμημένου συστήματος.

Η υλοποίηση του μηχανισμού αντιγραφής πραγματοποιείται στα περισσότερα συστήματα με τη χρήση χρονοσφραγίδων. Υποστηρίζονται δύο πρωτόκολλα, αντιγραφής το Change Log πρωτόκολλο και το LCUP (LDAP Client Update Protocol). Με τη χρήση του LCUP οι κύριοι διακομιστές ενημερώνουν τους καταναλωτές, με τη χρήση cookies, χωρίς να μεταφέρουν ολόκληρη τη πληροφορία. Με τη χρήση του Change Log πρωτόκολλου κάθε κύριος διακομιστής διατηρεί το δικό του αρχείο καταγραφής αλλαγών για κάθε αντίγραφο. Το αρχείο αυτό περιγράφει τις αλλαγές που πρέπει να προωθηθούν στους καταναλωτές αλλά και στους υπόλοιπους κύριους διακομιστές στην περίπτωση της multi-master αντιγραφής.

2.4 OpenLDAP

Ο OpenLDAP [1] συνιστά λογισμικό ανοικτού κώδικα Υπηρεσίας Καταλόγου. Η ανάπτυξη του υποστηρίζεται από μία παγκόσμια κοινότητα εθελοντών που επικοινωνούν μεταξύ τους με σκοπό την υλοποίηση του λογισμικού και του συνοδευτικού υλικού. Συνεπώς αποτελεί προϊόν μιας συλλογικής προσπάθειας με στόχο τη δημιουργία ενός πλήρους λειτουργικού προϊόντος λογισμικού. Η χρήση του λογισμικού OpenLDAP είναι ιδιαίτερα διαδεδομένη και πλέον είναι σε εμπορικά ώριμη μορφή.

Το λογισμικό του OpenLDAP περιλαμβάνει τον μοναδικό (stand-alone) διακομιστή LDAP (slapd), , LDAP C++ σετ εργαλείων ανάπτυξης λογισμικού (Software Development Kit), βιβλιοθήκες (lldap – LDAP βιβλιοθήκη πελάτη, llber - BER/DER βιβλιοθήκη κωδικοποίησης/αποκωδικοποίησης), συμπληρωματικά εργαλεία (LDIF εργαλεία για μετατροπή δεδομένων, LDAP εργαλεία γραμμής εντολών, SNACC - ASN.1 εργαλεία ανάπτυξης, clients κλπ) και τη σχετική τεκμηρίωση.

Η έκδοση, του OpenLDAP 2.4.21 (19/2/2010) υποστηρίζει τις περισσότερες πλατφόρμες Unix ή UNIX-like (Linux, FreeBSD, NetBSD, OpenBSD, AIX, HP-UX, Solaris κλπ). Προηγούμενες εκδόσεις του λογισμικού διατίθενται για άλλες πλατφόρμες όπως Apple MacOS X, IBM zOS, και Microsoft Windows 2000 (OpenLDAP 2.2.29).

2.5 Προβλήματα στην απευθείας πραγματοποίηση αλλαγών μέσω LDAP

Στην τρέχουσα κατάσταση που επικρατεί στην ακαδημαϊκή κοινότητα, όλες οι λειτουργίες αλληλεπίδρασης των εφαρμογών/υπηρεσιών με την υπηρεσία καταλόγου πραγματοποιούνται απευθείας με τη χρήση του πρωτοκόλλου LDAP (και τη χρήση κατάλληλης πιστοποίησης και Access List ανά υπηρεσία). Η περίπτωση των λειτουργιών ανάγνωσης δε δημιουργεί πρόβλημα αλλά στην περίπτωση λειτουργιών εγγραφής (προσθήκη, διαγραφή και ανανέωση εγγραφών) παρουσιάζονται οι παρακάτω δυσκολίες:

- Αλλαγές συμβαίνουν από μία πλειάδα πελατών και όχι από ένα σημείο εισόδου-εξόδου, γεγονός το οποίο μειώνει τις δυνατότητες αυστηρού ελέγχου της διαδικασίας, αναλυτικής καταγραφής των αιτήσεων (logging) και εφαρμογής μέτρων ελέγχου της πρόσβασης (access lists).
- Κάθε υπηρεσία διαθέτει γνώση μόνο του σχήματος δεδομένων που αναφέρεται στην ίδια την υπηρεσία και όχι του συνολικού σχήματος που απαιτείται να διατηρείται σε κάθε εγγραφή. Αυτό γίνεται ιδιαίτερα σαφές στην περίπτωση της λειτουργίας δημιουργίας εγγραφής. Στην πλειοψηφία των περιπτώσεων μία εγγραφή αναφέρεται σε φυσικό πρόσωπο στο οποίο παρέχεται ένας αριθμός από δικτυακές υπηρεσίες. Οι υπηρεσίες αυτές απαιτούν την ύπαρξη συγκεκριμένων attributes στην εγγραφή του χρήστη,

attributes τα οποία είναι γνωστά (και χρησιμοποιούνται) μόνο από την εκάστοτε υπηρεσία. Κατά συνέπεια εάν η δημιουργία της εγγραφής χρήστη πραγματοποιείται από μία υπηρεσία η εγγραφή θα λαμβάνει μόνο τα attributes που αντιστοιχούν στην υπηρεσία αυτή και δε θα απολαμβάνει πρόσβαση σε άλλες υπηρεσίες.

- Ένας αριθμός από attributes (πχ username, email address κτλ) απαιτείται να είναι μοναδικά για το σύνολο των εγγραφών της υπηρεσίας καταλόγου. Κάτι τέτοιο δεν είναι ιδιαιτέρως εύκολο να το εξασφαλίσει μία υπηρεσία κατά τη δημιουργία μίας εγγραφής αλλά μόνο η ίδια η υπηρεσία καταλόγου.
- Η δημιουργία (ή αλλαγή) κάποιων attributes απαιτεί την ταυτόχρονη αλλαγή κάποιων άλλων παραγόμενων attributes (πχ η αλλαγή του ονόματος ενός χρήστη απαιτεί την ταυτόχρονη αλλαγή του παραγόμενου ονοματεπώνυμου).
- Αλλαγές σε εγγραφές (κάθε είδους) σε πολλές περιπτώσεις απαιτούν την εκτέλεση εξωτερικών εργασιών ανεξάρτητων από την ίδια την υπηρεσία καταλόγου.

Οι παραπάνω δυσκολίες δεν μπορούν να απλοποιηθούν από την τρέχουσα υλοποίηση της υπηρεσίας καταλόγου. Η υπηρεσία καταλόγου παρέχει μόνο απευθείας LDAP πρόσβαση η οποία δεν δίνει τη δυνατότητα λειτουργιών post-operation ή αυτόματης αλλαγής σε attributes πέραν αυτών για τα οποία αιτήθηκε αλλαγή. Τέτοιες λειτουργίες είναι ευθύνη ενός ενδιάμεσου στρώματος λειτουργιών (middleware) το οποίο αναλαμβάνει να αυξήσει την ‘εξυπνάδα’ στις λειτουργίες εγγραφής και να εκτελέσει τυχόν επιπλέον λειτουργίες που απαιτούνται μετά την πραγματοποίηση κάποιας αλλαγής.

Το middleware αυτό απαιτεί επιπλέον φόρτο τόσο αρχικής ανάπτυξης του όσο και δημιουργίας συγκεκριμένων interfaces για κάθε προβλεπόμενη λειτουργία. Η ανάπτυξη του όμως αυξάνει την ευελιξία της υπηρεσίας, προσφέρει ένα έτοιμο και πλήρες interface για κάθε νέα υπηρεσία που επιθυμεί να συνδεθεί στην υπηρεσία καταλόγου και επιτρέπει την εύκολη επέκταση των λειτουργιών εγγραφής όποτε απαιτηθεί.

Ο σχεδιασμός του ενδιάμεσου στρώματος αυτού θα πρέπει να περιλαμβάνει πολλαπλά στρώματα με συγκεκριμένες λειτουργίες και ‘γνώση περιβάλλοντος’ σε κάθε στρώμα ώστε να είναι εύκολη η διαμόρφωση των ιδιαίτερων χαρακτηριστικών λειτουργίας κάθε στρώματος.

3 Υπηρεσίες Ιστού (Web Services)

3.1 Επισκόπηση της τεχνολογία των Web Services

Οι υπηρεσίες ιστού (web services) είναι μια ανερχόμενη και πολλά υποσχόμενη τεχνολογία που συνεχώς εξαπλώνεται σε εφαρμοσμένα πληροφοριακά συστήματα. Τα web services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML (χρησιμοποιώντας το πρωτόκολλο HTTP για επικοινωνία). Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από web services οι οποίες αλληλεπιδρούν μεταξύ τους αποτελεί μια εφαρμογή web services.

Σύμφωνα με τον οργανισμό W3C (World Wide Web Consortium) "η υπηρεσία ιστού αποτελεί ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει τη διαλειτουργικότητα μεταξύ των μηχανημάτων πάνω από δίκτυο". Μία υπηρεσία ιστού συνεπώς αποτελεί σύστημα λογισμικού το οποίο αναγνωρίζεται από ένα URI [IETF RFC 2396], ενώ η διεπαφή καθώς και οι δράσεις της ορίζονται πλήρως και περιγράφονται σε eXtensble Markup Language (XML) μορφή.

Παρά το γεγονός ότι η χρήση των υπηρεσιών ιστού δεν περιορίζεται με συγκεκριμένα πρωτόκολλα, μετά από μία κοινή αποδοχή των μεγαλύτερων εταιρειών λογισμικού στον κόσμο της αρχιτεκτονικής αυτής, έχει πλέον καθοριστεί ένα πιο συγκεκριμένο μοντέλο σύμφωνα με το οποίο θα πρέπει οι εταιρείες να παράγουν και να χρησιμοποιούν τις συγκεκριμένες υπηρεσίες..

«Το SOAP στην έκδοση 1.2 είναι ένα ελαφρύ πρωτόκολλο προορισμένο για την ανταλλαγή δομημένων πληροφοριών σε ένα αποκεντρωμένο, διανεμημένο περιβάλλον. Χρησιμοποιεί τεχνολογίες XML για να καθορίσει ένα επεκτάσιμο πλαίσιο παρέχοντας

μια δομή μηνυμάτων η οποία μπορεί να ανταλλαχθεί πάνω από ποικίλα δικτυακά πρωτόκολλα. Το πλαίσιο έχει σχεδιαστεί να είναι ανεξάρτητο από οποιοδήποτε προγραμματιστικό μοντέλο και σημασιολογία υλοποίησης» ***UDDI.org***

Για την επικοινωνία συνεπώς με βάση μοντέλο αυτό στο επίπεδο εφαρμογής χρησιμοποιείται συνήθως το πρωτόκολλο HTTP. Πάνω από το HTTP πρωτόκολλο χρησιμοποιείται το πρωτόκολλο της W3C Simple Object Access Protocol (SOAP), πρωτόκολλο το οποίο επιτρέπει την ανταλλαγή XML μηνυμάτων (αντικειμένων) πάνω από δίκτυο. Το SOAP πρωτόκολλο μπορεί να λειτουργήσει και πάνω από άλλα πρωτόκολλα (FTP, SMTP κλπ). Με τη χρήση του υλοποιείται ο διακομιστής SOAP (αρχιτεκτονική client-server). Για κάθε λειτουργία της υπηρεσίας ιστού, ορίζεται μία λειτουργία στον SOAP διακομιστή. Έπειτα, το σύνολο των λειτουργιών του SOAP διακομιστή καθώς και τα υπόλοιπα χαρακτηριστικά του, περιγράφονται σε ένα αρχείο που ονομάζεται WSDL (Web Service Discription Language) και το οποίο δημοσιεύεται στο Internet έτσι ώστε να δίνεται η δυνατότητα σε κάθε ενδιαφερόμενο χρήστη να μπορεί άμεσα να χρησιμοποιήσει την υπηρεσία. Οι περισσότερες γλώσσες πλέον προγραμματισμού από την Delphi v7 μέχρι το Visual Studio .net, την PHP, την JSP και άλλες πολλές, υποστηρίζουν τη δημιουργία SOAP διακομιστών με πάρα πολύ απλό τρόπο.

«Η WSDL είναι ένα σχήμα XML για την περιγραφή δικτυακών υπηρεσιών σαν ένα σύνολο από τελικά σημεία που λειτουργούν σε μηνύματα τα οποία περιέχουν πληροφορία είτε προσανατολισμένη στα έγγραφα είτε προσανατολισμένη στις διαδικασίες.» ***W3C***

Η WSDL (Web Service Description Language) είναι μία γλώσσα σε XML μορφή η οποία περιγράφει απόλυτα μία web service. Με πιο απλά λόγια η WSDL μας βηθάει να περιγράψουμε ένα σύνολο από λειτουργίες, μηνύματα και το πώς αυτά τα μηνύματα ανταλλάσσονται. Έτσι για κάθε web services που δημιουργείται, αντίστοιχα πρέπει να δημιουργείται ένα αρχείο WSDL στο οποίο θα καταγράφονται όλες οι πληροφορίες για την ίδια την υπηρεσία. Πιο συγκεκριμένα εκεί καταγράφεται η IP διεύθυνση του διακομιστή, ποιες λειτουργίες υποστηρίζει καθώς και πως δέχεται και πως επιστρέφει τα δεδομένα για κάθε λειτουργία. Υπάρχουν πάρα πολλά

διαθέσιμα εργαλεία που δημιουργούν αυτόματα ένα WSDL αρχείο παράλληλα με τη δημιουργία του SOAP server.

«*Τα web services έχουν νόημα μόνο όταν δυνητικοί χρήστες μπορούν να βρουν πληροφορίες ικανές ώστε να επιτρέψουν την εκτέλεσή τους.*» ***OASIS***

Τέλος το UDDI (Universal Description, Discovery, and Integration) αποτελεί ένα πρωτόκολλο καταχώρησης για web services. Χρησιμοποιείται για να μπορούμε να παρέχουμε πληροφορίες για τις υπηρεσίες ιστού. Κάθε καταχώρηση περιέχει το WSDL αρχείο και τη διεύθυνση που λειτουργεί η υπηρεσία στο Internet. Επιπρόσθετα σε κάθε καταχώρηση υπάρχουν και διάφορες άλλες πληροφορίες για την υπηρεσία που σχετίζονται με τον ιδιοκτήτη της, την πολιτική του κλπ. Υπάρχουν διαφορετικοί τύποι καταχωρήσεων μίας υπηρεσίας. Πιο συγκεκριμένα υπάρχουν καταχωρήσεις που μπορούν να γίνουν για υπηρεσίες από όλο τον κόσμο και που απευθύνονται σε όλο τον κόσμο, αλλά και καταχωρίσεις που απευθύνονται μόνο σε εξειδικευμένες επιχειρήσεις προωθώντας έτσι και το B2B μοντέλο συνεργασίας.
Τέλος υπάρχουν και καταχωρήσεις υπηρεσιών για πιο εξειδικευμένες περιπτώσεις.

Πληροφορία	**Λειτουργίες**
White pages: Πληροφορίες όπως το όνομα, η διεύθυνση, το τηλέφωνο και άλλες πληροφορίες επικοινωνίας για μία επιχείρηση.	**Publish**: Πώς ο προμηθευτής ενός web service καταχωρεί των εαυτό του.
Yellow Pages: Πληροφορίες που κατηγοριοποιούν επιχειρήσεις. Βασίζονται σε υπάρχοντα πρότυπα κατηγοριοποίησης (μη ηλεκτρονικά).	**Find**: Πώς μία εφαρμογή βρίσκει ένα συγκεκριμένο web service.
Green Pages: Τεχνικές πληροφορίες για τα web service που παρέχονται από μία	**Bind**: Πώς μία εφαρμογή συνδέεται, και αλληλεπιδρά με ένα web service αφού

επιχείρηση.	αυτό βρεθεί.

Υπηρεσίες του UDDI

Επίπεδο	**Τεχνολογία**
Ομοιόμορφη μορφή και ανταλλαγή δεδομένων	XML
Πρότυπο κανάλι επικοινωνίας	SOAP
Πρότυπη περιγραφική γλώσσα για την περιγραφή των παρεχόμενων υπηρεσιών	WSDL
Καταχώρηση και εντοπισμός των παρεχόμενων υπηρεσιών	UDDI

Βασικές Τεχνολογίες των Web Services

3.1.1 Remote procedure call

Κλήση απομακρυσμένης διαδικασίας (RPC) είναι μια διαδικασία επικοινωνίας που επιτρέπει σε ένα πρόγραμμα να καλέσει μια υπορουτίνα ή διαδικασία για την εκτέλεση σε άλλο χώρο διευθύνσεων (συνήθως σε κάποιον άλλο υπολογιστή σε έναν κοινόχρηστο δίκτυο) χωρίς ο προγραμματιστής να καθορίζει τις λεπτομέρειες για αυτή την απομακρυσμένη αλληλεπίδραση. Δηλαδή, ο προγραμματιστής θα γράψει τον ίδιο κώδικα είτε η υπορουτίνα είναι τοπική, ή εξ αποστάσεως. Όταν το εν λόγω λογισμικό είναι γραμμένο με αντικειμενοστραφή αρχές (object-oriented), το RPC ενδέχεται να αναφέρεται ως απομακρυσμένης επίκλησης (remote invocation).

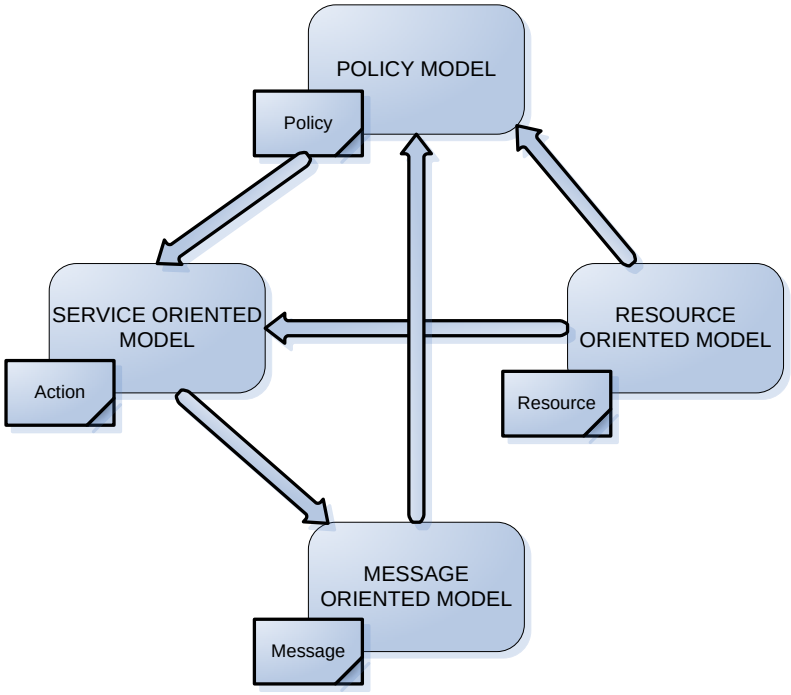
3.1.2 Message passing

Το RPC είναι ένα προφανές και δημοφιλές μοντέλο για την εφαρμογή του μοντέλου πελάτη- διακομιστή. Ένα RPC ξεκινά από τον πελάτη στέλνοντας ένα αίτημα προς ένα γνωστό, απομακρυσμένο server για να εκτελέσει μια συγκεκριμένη διαδικασία παρέχοντάς του συγκεκριμένες παραμέτρους. Έπειτα η απάντηση επιστρέφεται στον πελάτη, όπου συνεχίζει μαζί με τη διαδικασία να εκτελείται. Ενώ ο διακομιστής επεξεργάζεται την κλήση, ο πελάτης σταματάει την εκτέλεση προγραμμάτων

(περιμένει μέχρι ο διακομιστής να ολοκληρώσει την επεξεργασία πριν αρχίσει και πάλι την εκτέλεση).

Μια σημαντική διαφορά μεταξύ των κλήσεων απομακρυσμένης διαδικασίας και των τοπικών κλήσεων είναι ότι οι απομακρυσμένες κλήσεις μπορεί να αποτύχουν εξαιτίας των απρόβλεπτων προβλημάτων δικτύου. Επίσης, οι καλούντες σε γενικές γραμμές πρέπει να ανταπεξέρχονται σε τέτοιου είδους βλάβες, ότι δηλαδή η απομακρυσμένη διαδικασία έγινε στην πραγματικότητα. Όταν οι διαδικασίες αυτές είναι γραμμένες σε κώδικα που δίνει την δυνατότητα σε μια ρουτίνα να τις καλεί περισσότερες από μια φορές και να μην αλλάζουν το αποτέλεσμα, μπορούν εύκολα να διαχειριστούν, αλλά δημιουργούνται αρκετές δυσκολίες όταν είναι γραμμένες οι διαδικασίας για συστήματα χαμηλού επιπέδου, όπου τότε οποιαδήποτε αλλαγή επιφέρει προβλήματα.

3.2 Αρχιτεκτονικά μοντέλα των Web Services

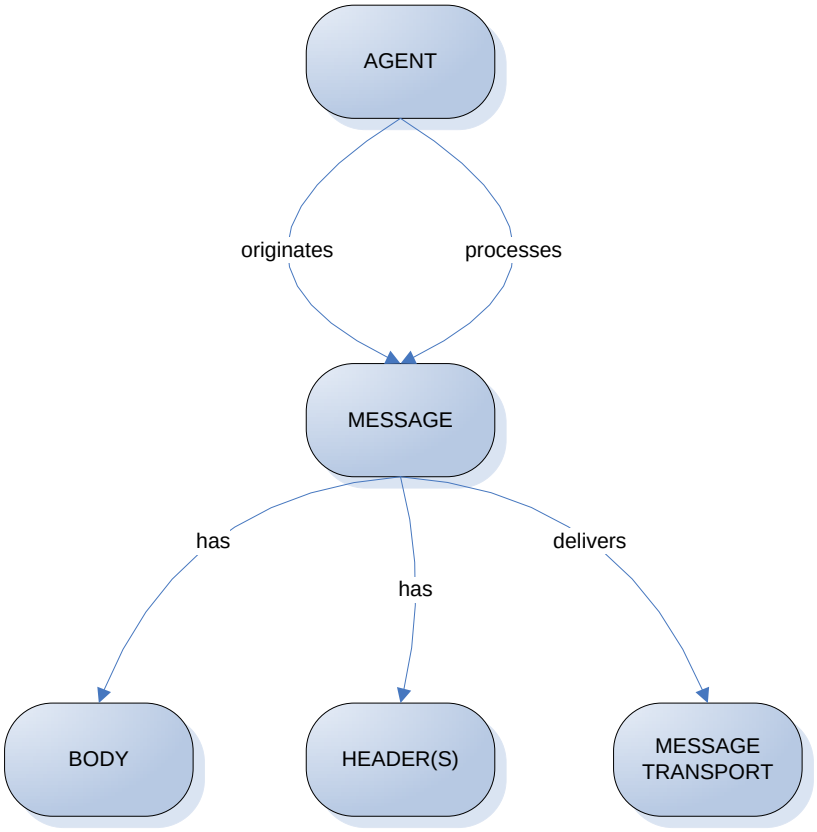


Σύνδεση των μοντέλων των Web Services

Στο παραπάνω διάγραμμα φαίνεται η ενδεχόμενη σύνδεση των αρχιτεκτονικών μοντέλων των Web Services. Θα εξηγήσουμε παρακάτω την επικοινωνία των τεσσάρων μοντέλων μεταξύ τους καθώς αναλύουμε το κάθε μοντέλο ξεχωριστά.

Τα τέσσερα μοντέλα είναι τα εξής:

1. Το Message Oriented μοντέλο επικεντρώνεται στα μηνύματα, στην δομή του μηνύματος, στην μεταφορά του μηνύματος δίχως να δίνει ιδιαίτερη αναφορά ως προς τους λόγους δημιουργίας των μηνυμάτων, ούτε για την σημασία τους.

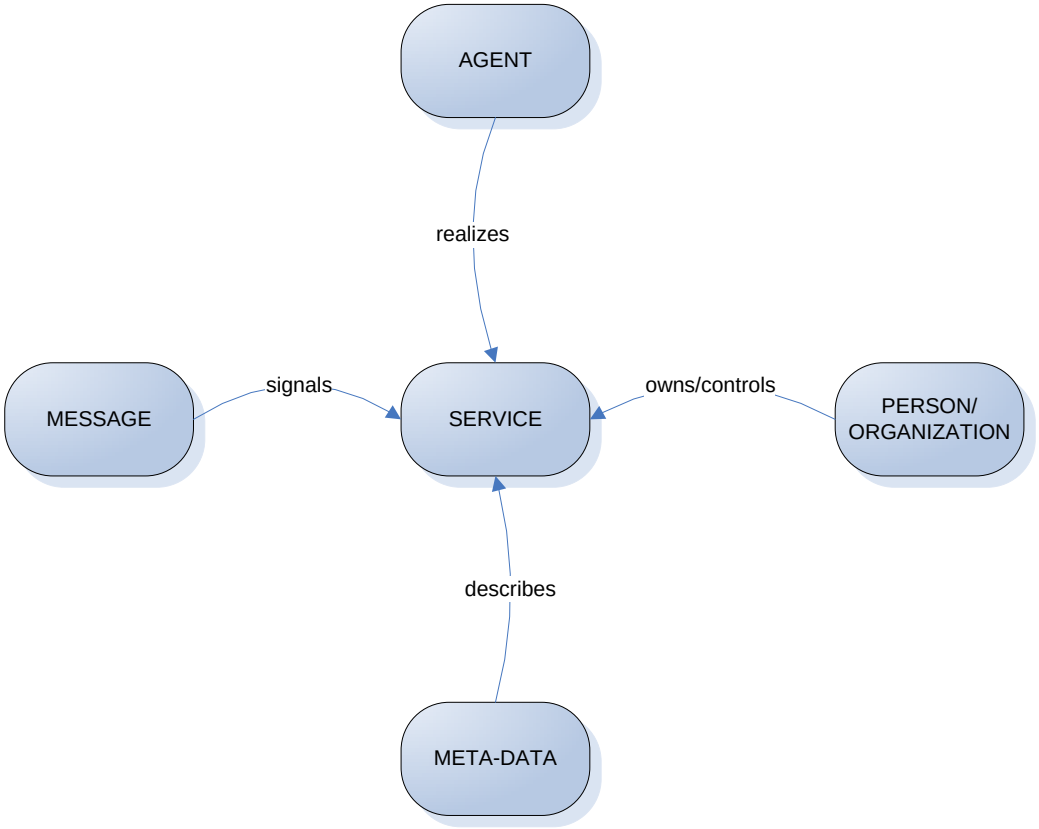


Απλοποιημένη μορφή μοντέλου Μηνύματος

Η ουσία του παραπάνω μοντέλου βρίσκεται γύρω από μερικές βασικές έννοιες: ο πράκτορας (agent) που στέλνει και λαμβάνει τα μηνύματα, η δομή του μηνύματος όσο αναφορά τις κεφαλίδες των μηνυμάτων (message headers) και το κυρίως σώμα (body) καθώς και τους μηχανισμούς που χρησιμοποιούνται για την παράδοση των μηνυμάτων. Φυσικά, υπάρχουν πρόσθετα στοιχεία για να ληφθούν υπόψη, όπως ο ρόλος των πολιτικών και το πώς αυτές διέπουν το πρότυπο μήνυμα.

2. Το Service Oriented μοντέλο επικεντρώνεται στις πτυχές της υπηρεσίας και της δράσης. Είναι σαφές ότι, σε κάθε κατανεμημένο σύστημα, οι υπηρεσίες

δεν μπορούν να υλοποιηθούν επαρκώς χωρίς την ύπαρξη των μηνυμάτων, ενώ το αντίστροφο δεν συμβαίνει: Τα μηνύματα δεν χρειάζονται να αφορούν τις υπηρεσίες.



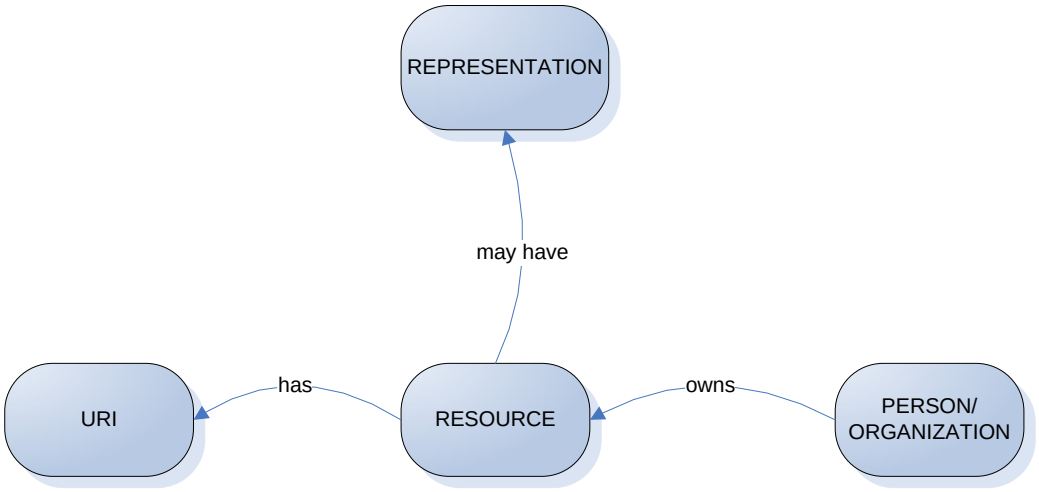
Απλοποιημένο Service Oriented Μοντέλο

Το Service Oriented μοντέλο είναι το πιο περίπλοκο από όλα τα μοντέλα της αρχιτεκτονικής. Ωστόσο, είναι στηριγμένο σε πολύ λίγες βασικές ιδέες. Μια υπηρεσία πραγματοποιείται από έναν πράκτορα (agent) και χρησιμοποιείται από άλλον πράκτορα. Οι Υπηρεσίες αυτές επάγονται μέσω των μηνυμάτων που ανταλλάσσονται μεταξύ των πρακτόρων που τις παρέχουν και των πρακτόρων που τις ζητάνε.

Μια πολύ σημαντική πτυχή των υπηρεσιών είναι η σχέση τους με τον πραγματικό κόσμο: οι υπηρεσίες έχουν αναπτυχθεί κυρίως για να προσφέρουν λειτουργικότητα στον πραγματικό κόσμο. Αυτό συνεπάγεται πως κάποιος όταν δημιουργεί μια υπηρεσία έχει ένα σκοπό να εξυπηρετήσει. Ο ιδιοκτήτης της υπηρεσίας αυτής μπορεί να είναι ένα πρόσωπο ή μια οργάνωση, η οποία έχει μια πραγματική παγκόσμια ευθύνη για την υπηρεσία.

Το Service Oriented μοντέλο κάνει χρήση των μετα-δεδομένων. Αυτά τα μετα-δεδομένα χρησιμοποιούνται για να καταγράψουν πολλές πτυχές των υπηρεσιών: από τα στοιχεία της διεπαφής και τον τρόπο μετάδοσής τους, μέχρι την σημασιολογία της υπηρεσίας και ποια πολιτική ασφαλείας έχει δοθεί στην υπηρεσία από τον ιδιοκτήτη της. Για αυτόν τον λόγω η παροχή μιας πλούσιας περιγραφής είναι το κλειδί για την επιτυχή ανάπτυξη και χρήση των υπηρεσιών μέσω του Internet.

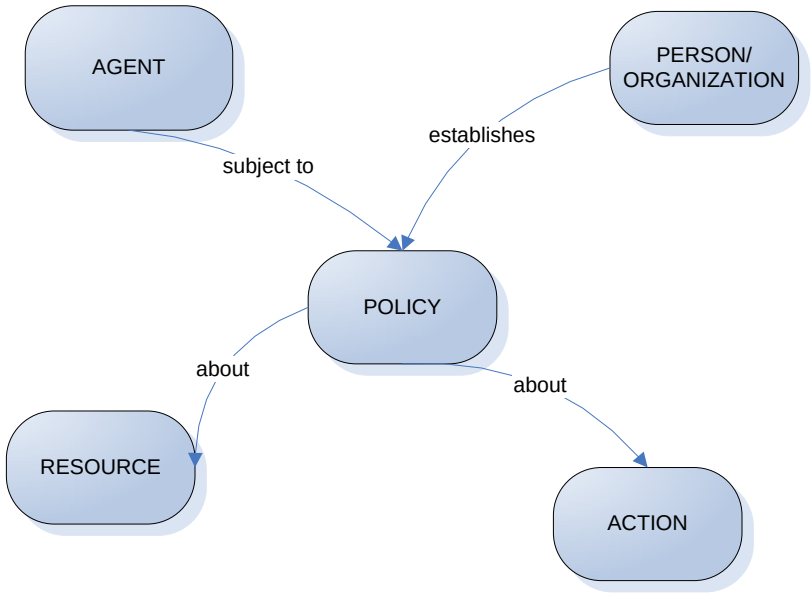
3. Το Resource Oriented μοντέλο επικεντρώνεται στους πόρους που υπάρχουν και έχουν ιδιοκτήτες.



Απλοποιημένη μορφή Resource Oriented Μοντέλου

Το μοντέλο των πόρων πηγάζει από την ίδια ιδέα της Web Αρχιτεκτονικής των πόρων. Επεκτείνοντας αυτό μπορούμε να συμπεριλάβουμε τις σχέσεις μεταξύ των πόρων και των ιδιοκτητών. Εξηγώντας θα έπρεπε να προστεθεί πως ο Uniform Resource Identifier στην πληροφορική (URI) αποτελείται από μια σειρά χαρακτήρων που χρησιμοποιούνται για τον εντοπισμό ενός πόρου στο Διαδίκτυο. Η ταυτοποίηση αυτή επιτρέπει την αλληλεπίδραση με τυχόν αναπαραστάσεις (representation) που μπορούν να έχουν οι πόροι μέσω δικτύου (συνήθως στο World Wide Web) χρησιμοποιώντας ειδικά πρωτόκολλα.

4. Τέλος το μοντέλο πολιτικής (Policy Model) επικεντρώνεται σε προβλήματα σχετικά με τη συμπεριφορά των πρακτόρων και των υπηρεσιών. Μπορούμε να γενικεύσουμε αυτό στους πόρους (όπως πριν) εφόσον οι πολιτικές μπορούν να εφαρμόζονται εξίσου στα έγγραφα (όπως οι περιγραφές των υπηρεσιών).



Απλοποιημένο Μοντέλο Policy

Οι πολιτικές δημιουργούνται για τους πόρους. Οι πολιτικές αυτές εφαρμόζονται στους πράκτορες που μπορεί να επιχειρήσουν την πρόσβαση στους πόρους αυτούς, και έχουν τεθεί σε εφαρμογή, ή είναι εγκατεστημένοι, από ανθρώπους που έχουν την ευθύνη των πόρων. Πολιτικές μπορούν να ληφθούν για να απαλείψουν τυχόν ανησυχίες για την ασφάλεια, για την ποιότητα των υπηρεσιών, για την διαχείριση τους και για ανησυχίες κατά την εφαρμογή τους.

3.3 Πλεονεκτήματα της αρχιτεκτονικής των υπηρεσιών ιστού

Η αρχιτεκτονική των υπηρεσιών ιστού παρέχει αρκετά πλεονεκτήματα όπως

- Διαλειτουργικότητα: Μία υπηρεσία ιστού παρέχει ανεξαρτησία τόσο από λειτουργικό σύστημα όσο και από το υλικό που χρησιμοποιείται. Οποιοδήποτε πρόγραμμα που υποστηρίζει αυτή τη τεχνολογία μπορεί πολύ εύκολα να προσπελάσει μία τέτοια υπηρεσία.

- Ενσωμάτωση: Η δημιουργία μίας υπηρεσία ιστού δεν απαιτεί αλλαγές στον μηχανισμό του λογισμικού συστήματος από το οποίο χρησιμοποιείται.
- Διαθεσιμότητα και δημοσίευση: Οι πληροφορίες για τις υπηρεσίες ιστού δημοσιεύονται οπότε η εύρεση και η χρήση τους μπορεί να είναι ταχύτατες.
- Επεκτασιμότητα: Η λειτουργία μίας υπηρεσίας ιστού είναι δυνατό να ανανεωθεί με εύκολο τρόπο παρέχοντας έτσι επιπρόσθετες υπηρεσίες στους χρήστες της.
- Χαμηλό κόστος δημιουργίας και χρήσης: Το κόστος ανάπτυξης – επέκτασης σε ένα υπάρχον λογισμικό σύστημα εφαρμογής σε περιβάλλον web για την δημιουργία υπηρεσίας ιστού της εφαρμογής κοστίζει ελάχιστα. Το κόστος ενσωμάτωσης μίας υπηρεσίας ιστού σε κάποιο δικτυακό τόπο ή σε εφαρμογή είναι πάρα πολύ μικρό.
- Χρήση λογισμικών συστημάτων: Όλα τα λογισμικά συστήματα και ειδικότερα οι ιστοσελίδες που χρησιμοποιούν έτοιμες υπηρεσίες γίνονται πιο λειτουργικά και πιο φιλικά αφού παρέχουν περισσότερες υπηρεσίες και σε εφαρμογές εκτός από χρήστες.

3.3.1 Πλεονεκτήματα της χρήσης web services στην τρέχουσα υλοποίηση

Το middleware που περιγράφηκε προηγουμένως μπορεί να υλοποιηθεί με τη χρήση της τεχνολογίας των web services. Πιο συγκεκριμένα, για κάθε εξωτερική υπηρεσία που απαιτείται να πραγματοποιήσει αλλαγές στην υπηρεσία καταλόγου δημιουργείται κατάλληλο προγραμματιστικό interface μέσω web services. Το interface αυτό λαμβάνει περιγραφικό όνομα και συγκεκριμένες λειτουργίες τις οποίες χρησιμοποιεί η εξωτερική υπηρεσία για την πραγματοποίηση των αλλαγών. Οι λειτουργίες αυτές αναφέρονται σε υψηλού επιπέδου εργασίες (πχ Δημιουργία Χρήστη, Ενεργοποίηση Υπηρεσίας) και όχι στις τελικές αλλαγές επί των δεδομένων της υπηρεσίας καταλόγου. Κάθε λειτουργία λαμβάνει συγκεκριμένες εισόδους (που εξαρτώνται από τις ανάγκες της λειτουργίας) και επιστρέφει το αποτέλεσμα της εργασίας (επιτυχία ή αποτυχία), τυχόν μήνυμα λάθους καθώς και άλλα στοιχεία που απαιτούνται ανά περίπτωση. Κατ’ αυτόν τον τρόπο:

- Διατηρείται η ήδη υπάρχουσα λογική ανάπτυξης εφαρμογών που προβλέπει τη δημιουργία συγκεκριμένου και αυστηρά ορισμένου πλαισίου συναρτήσεων

– λειτουργιών οι οποίες καλούνται για την υλοποίηση εργασιών (με συγκεκριμένες παραμέτρους) και οι οποίες αναλαμβάνουν την εκτέλεση τους και την επιστροφή των αποτελεσμάτων (functional API).

- Χρησιμοποιείται μία τεχνολογία (web service) που είναι ανεξάρτητη της προγραμματιστικής πλατφόρμας με συνέπεια να παρέχει τη δυνατότητα συνεργασίας μεταξύ ετερογενών συστημάτων (πχ .Net με Java), ενώ τα πρωτόκολλα επικοινωνίας (HTTP, XML, WSDL, SOAP) είναι ιδιαίτερα διαδεδομένα και πολύ καλά ορισμένα με συνέπεια να αποφεύγονται τυχόν περιπτώσεις ασυμβατότητας.
- Όλες οι αλλαγές στα δεδομένα της υπηρεσίας καταλόγου πραγματοποιούνται κεντρικά διαμέσου των web services και όχι με απευθείας πρόσβαση από τις εξωτερικές υπηρεσίες. Αυτό έχει ως συνέπεια να μπορεί να οριστεί πολύ καλύτερα η πολιτική ασφάλειας επί των δεδομένων και να υπάρχει πλήρης έλεγχος επί των λειτουργιών που πραγματοποιούνται. Αντί οι εξωτερικές υπηρεσίες να έχουν αυξημένα δικαιώματα αλλαγών στα δεδομένα, απλά καλούν συγκεκριμένες συναρτήσεις. Έτσι, οι αλλαγές που μπορεί να πραγματοποιηθούν είναι πολύ συγκεκριμένες, γίνονται όλες από το ίδιο σημείο και μπορούν εύκολα να παρακολουθηθούν και να αποσφαλματοποιηθούν.
- Η κεντρική πραγματοποίηση των λειτουργιών επιτρέπει τη διατήρηση κεντρικού ιστορικού αλλαγών. Παράλληλα, ο συγγραφέας/διαχειριστής των web services έχει τη δυνατότητα να επιλέξει το επίπεδο λεπτομέρειας εργασιών που θα διατηρούνται στο ιστορικό. κάτι που δεν είναι δυνατό με την υπηρεσία καταλόγου όπου απλά διατηρούνται γενικές πληροφορίες για τις ενέργειες που εκτελούνται (αναλυτικό ιστορικό δεν είναι δυνατό να διατηρείται λόγω του πολύ μεγάλου πλήθους λειτουργιών, ιδιαίτερα αναζητήσεων, που πραγματοποιούνται σε μία υπηρεσία καταλόγου).
- Είναι δυνατόν να οριστούν εξωτερικές λειτουργίες που εκτελούνται πριν ή μετά την κλήση/ολοκλήρωση των web services. Έτσι για παράδειγμα η μετακίνηση ενός χρήστη στο δέντρο πληροφοριών, μπορεί να οδηγεί στην εκτέλεση εξωτερικής λειτουργίας που μετακινεί το mailbox του σε άλλο εξυπηρετητή email.

- Δεν απαιτείται η παροχή όλων των τυχόν attributes που μπορεί να περιέχει μία εγγραφή αλλά μόνο των απολύτως απαραίτητων για την πραγματοποίηση της λειτουργίας. Τα υπόλοιπα attributes μπορούν να είναι παραγόμενα με βάση κατάλληλους κανόνες και συναρτήσεις. Έτσι για παράδειγμα μπορεί από το username του χρήστη να προκύπτει το email του (με βάση τον τύπο email=<username>@<domain>) ενώ τα όρια χρήσης της dialup σύνδεσης του (ημερήσιο και εβδομαδιαίο όριο) να είναι στατικά ορισμένα και να λαμβάνουν συγκεκριμένες και προκαθορισμένες τιμές τις οποίες δεν μπορεί να επηρεάσει ούτε ο χρήστης, ούτε η εξωτερική υπηρεσία που καλεί το web service.
- Ο διαχειριστής των web services μπορεί να ορίσει συγκεκριμένες και αρκετά περίπλοκες πολιτικές επί των τιμών των παρερχομένων attributes, κάτι που δεν είναι δυνατό από την υπηρεσία καταλόγου. Έτσι για παράδειγμα μπορεί να ορίσει συγκεκριμένη πολιτική επί του επιλεγόμενου username/password ή να ορίσει απαγορευμένες τιμές για άλλα attributes.
- Οι εξωτερικές υπηρεσίες δεν απαιτείται να γνωρίζουν απολύτως τίποτα για το σχήμα δεδομένων που χρησιμοποιείται από την υπηρεσία καταλόγου. Απλά παρέχουν τιμές σε συγκεκριμένα και αυστηρά ορισμένα ορίσματα των λειτουργιών που έχουν συμφωνηθεί χωρίς να έχουν γνώση πώς αυτά αντιστοιχίζονται σε attributes στις εγγραφές της υπηρεσίας καταλόγου. Ο διαχειριστής των δεδομένων, μπορεί να κάνει οποιαδήποτε μετατροπή, προσθήκη, αφαίρεση στο σχήμα δεδομένων χωρίς να απαιτείται να ενημερώσει τις εξωτερικές υπηρεσίες παρά μόνο να κάνει τις κατάλληλες αλλαγές στον κώδικα των web services (αν απαιτείται).
- Η λειτουργία διαγραφής χρήστη μπορεί να υλοποιηθεί με την μορφή λήξης εγγραφής και όχι διαγραφής της. Έτσι για παράδειγμα κάποιος χρήστης του οποίου η εγγραφή έχει λήξει μπορεί να ενημερωθεί μέσω email (χρησιμοποιώντας εξωτερικό port-operation όπως περιγράφηκε προηγουμένως) ώστε να μπορέσει να αλλάξει email διεύθυνση σε εύλογο χρονικό διάστημα. Κατ’ αυτόν τον τρόπο δίνεται η δυνατότητα για ορισμό συγκεκριμένης πολιτικής λήξης και διαγραφής εγγραφών αντί για την απλή διαγραφή τους όταν αυτό ζητείται από εξωτερική υπηρεσία. Παράλληλα, η εγγραφή ενός χρήστη στην υπηρεσία καταλόγου μπορεί να αντιστοιχεί σε πολλαπλές εγγραφές σε εξωτερικές υπηρεσίες στην οποία περίπτωση η

λειτουργίας διαγραφής θα πρέπει απλά να διαγράφει τα attributes που αντιστοιχούν στην κάθε εξωτερική υπηρεσία κάτι που μπορεί να υλοποιηθεί πολύ εύκολα μέσω των web services.

3.3.2 Πολύ-στρωματική λογική λειτουργίας του προτεινόμενου συστήματος

Πέραν των γενικών πλεονεκτημάτων που περιγράφηκαν παραπάνω, η λογική σχεδίασης και ανάπτυξης του συστήματος ακολουθεί μία αρχιτεκτονική πολλαπλών στρωμάτων (layers), τα οποία θα είναι όσο το δυνατόν ανεξάρτητα και αυτόνομα, με τη δική τους διαμόρφωση και όσο το δυνατόν περιορισμένη γνώση του συνολικού συστήματος και των αναγκών του. Κατά αυτόν τον τρόπο το σύστημα γίνεται 'pluggable', με ευκολία διαμόρφωσης, ελέγχου και αποσφαλμάτωσης κάθε επιπέδου και μειωμένων αναγκών ανάπτυξης κάθε νέας διεπαφής.

Πιο συγκεκριμένα, κάθε διεπαφή θα περιλαμβάνει τα ακόλουθα στρώματα:

1. Στρώμα Web Servies: Στο συγκεκριμένο στρώμα θα υπάρχει μία εφαρμογή 'εξυπηρετητή' web services (στην περίπτωση μας γραμμένη σε γλώσσα δικτυακού προγραμματισμού PHP) η οποία και θα εμφανίζει ένα συγκεκριμένο συναρτησιακό περιβάλλον προς χρήση από τους πελάτες. Κάθε συνάρτηση θα εκτελεί τον αρχικό έλεγχο των δεδομένων εισόδου, τυχόν περίπλοκους υπολογισμούς, καταγραφή δεδομένων και στη συνέχεια θα χρησιμοποιεί το στρώμα της βιβλιοθήκης επαφής με την υπηρεσία καταλόγου προκειμένου να πραγματοποιήσει τις αναγκαίες αλλαγές. Η βιβλιοθήκη επαφής θα παρέχει περιορισμένο εύρος συναρτήσεων (Add, Delete, Modify κτλ) και θα βασίζεται στο αρχείο διαμόρφωσης ανά τύπο αντικειμένου προκειμένου να κάνει τις απαραίτητες ενέργειες στην υπηρεσία καταλόγου.
2. Στρώμα βιβλιοθήκης Επαφής: Μία βιβλιοθήκη λειτουργιών (γραμμένη και αυτή σε PHP) θα παρέχει τις συναρτήσεις επαφής με την υπηρεσία καταλόγου. Κάθε συνάρτηση θα αφορά μία συγκεκριμένη λειτουργία και θα αποτελείται από το όνομα της, τον τύπο του αντικειμένου και κατάλληλα ανά περίπτωση δεδομένα εισόδου. Στο επίπεδο αυτό και με βάση ένα ευέλικτο και αναλυτικό αρχείο διαμόρφωσης θα εκτελούνται οι περισσότερες λειτουργίες όπως:
    i. Έλεγχος μοναδικότητας.

ii. Παραγόμενα attributes.
iii. Έλεγχος δεδομένων για συμφωνία με τον τύπο που αντιστοιχεί σε κάθε attribute και τυχόν επιπλέον έλεγχος περιορισμένου λεξικού τιμών.

3. Στρώμα επικοινωνίας με υπηρεσία καταλόγου: Στο επίπεδο αυτό, η βιβλιοθήκη λειτουργιών θα χρησιμοποιεί την ήδη διαθέσιμη βιβλιοθήκη επικοινωνίας με υπηρεσίες LDAP (που παρέχεται στη γλώσσα PHP) ώστε να εκτελέσει τις τελικές ενέργειες στην υπηρεσία καταλόγου.

4 Ανάπτυξη συστήματος

4.1 Βασικές απαιτήσεις προτεινόμενου συστήματος υπηρεσίας καταλόγου

Η υπό ανάπτυξη υπηρεσία καταλόγου θα πρέπει να χαρακτηρίζεται από τα ακόλουθα στοιχεία:

- Πλήρη υποστήριξη του πρωτοκόλλου της υπηρεσίας (LDAPv3)
- Επεκτασιμότητα σε επίπεδο σχήματος δεδομένων και λειτουργικότητας
- Ασφάλεια και εμπιστευτικότητα δεδομένων
- Ανοικτό κώδικα για την εύκολη επέκταση της υπηρεσίας και επίλυση τυχόν προβλημάτων στο λογισμικό
- Δομή υψηλής διαθεσιμότητας
- Δυνατότητα συγχρονισμού με εξωτερικά συστήματα
- Αυξημένες επιδόσεις οι οποίες να έχουν όριο μόνο το υλικό της υπηρεσίας και όχι εγγενείς περιορισμούς. Η απόδοση της υπηρεσίας θα πρέπει να μπορεί να αυξηθεί γραμμικά με βάση το διαθέσιμο υλικό καθώς και να υποστηρίζει το διαχωρισμό της υπηρεσίας σε πολλαπλούς χαμηλού κόστους εξυπηρετητές αντί για μία υψηλού κόστους μηχανή.
- Χρήση πολλαπλών εξυπηρετητών με συγχρονισμό δεδομένων μεταξύ τους. Τυχόν απώλεια ενός εξυπηρετητή δε θα πρέπει να επηρεάζει τη συνολική

υπηρεσία πέραν της απώλειας του αντίστοιχου ποσοστού συνολικής ισχύος (no single point of failure)

- Δυνατότητα απομακρυσμένης διαχείρισης της υπηρεσίας

Σε επίπεδο λογισμικού υπηρεσίας καταλόγου οι παραπάνω απαιτήσεις μεταφράζονται ως εξής:

- Ανοικτός κώδικας λογισμικού
- Παροχή plugin interface για την επέκταση της λειτουργικότητας της υπηρεσίας καταλόγου
- Παροχή λειτουργίας συγχρονισμού δεδομένων (replication)
- Πολλαπλούς write master δεδομένων (multi master replication)
- Παροχή κονσόλας διαχείρισης της υπηρεσίας.
- Ενσωμάτωση των στοιχείων διαμόρφωσης της υπηρεσίας εντός του δέντρου δεδομένων ώστε να είναι δυνατή η απομακρυσμένη διαχείριση της υπηρεσίας με απευθείας χρήση ldap πρωτοκόλλου
- Υποστήριξη για πρωτόκολλο ασφαλούς μετάδοσης δεδομένων SSL/TLS.
- Δυνατότητα κρυπτογράφησης συγκεκριμένων attributes
- Δυνατότητα λήψης αντιγράφων ασφαλείας της βάσης δεδομένων της υπηρεσίας και επαναφοράς της υπηρεσίας από αντίγραφο ασφαλείας (backup, restore)
- Όλες οι λειτουργίες διαχείρισης θα πρέπει να μπορούν να γίνουν και ολοκληρωθούν εν λειτουργία (online) χωρίς να απαιτείται η απενεργοποίηση της υπηρεσίας (import/export δεδομένων, backup/restore δεδομένων)
- Ορισμός των στοιχείων πρόσβασης (Access Lists) εντός του δέντρου δεδομένων ώστε η αλλαγή τους να μπορεί να γίνει απομακρυσμένα και να μην απαιτεί επανεκκίνηση της υπηρεσίας
- Υποστήριξη περιορισμών χρήσης πόρων (resource limits) για την αποφυγή κακής χρήσης της υπηρεσίας (Denial of Service attacks). Είναι επιθυμητό οι περιορισμοί αυτοί να μπορούν να τεθούν ανά συνδεόμενο χρήστη.

- Υποστήριξη της γλώσσας DSML για την πρόσβαση στην υπηρεσία καταλόγου από εξωτερικά υποσυστήματα που δεν υποστηρίζουν LDAP αλλά μόνο πρόσβαση μέσω web services.
- Δυνατότητα ορισμού πολιτικής επί των passwords των χρηστών (password policy) καθώς και απενεργοποίησης της πρόσβασης σε μία εγγραφή χωρίς να απαιτείται η πλήρη διαγραφή της (account deactivation).
- Πλήρες logging της υπηρεσίας.

4.2 Διαθέσιμα λογισμικά

Παρακάτω παρουσιάζονται εμπορικά και ελεύθερα πακέτα λογισμικού που μπορούν να χρησιμοποιηθούν για την υλοποίηση της υπηρεσίας καταλόγου. Συνοπτικά αυτά είναι τα:

- OpenLDAP Directory Server
- Fedora Directory Server
- Red Hat Directory Server
- Sun Java System Directory Server Enterprise Edition
- Windows Active Directory
- IBM Tivoli Directory Server
- Novell eDirectory

Τα χαρακτηριστικά των εν λόγω λογισμικών παρουσιάζονται συνοπτικά παρακάτω.

Λειτουργία σε περιβάλλον UNIX: Ολα τα παρουσιαζόμενα λογισμικά με την εξαίρεση του Windows Active Directory, ήτοι τα OpenLDAP, Fedora Directory Server, Red Hat Directory Server, Sun Java System Directory Server Enterprise Edition, IBM Tivoli Directory Server, Novell eDirectory, είναι δυνατόν να λειτουργήσουν σε περιβάλλον UNIX.

Λειτουργία σε περιβάλλον Windows: Από τα παρουσιαζόμενα λογισμικά δεν είναι δυνατόν να λειτουργήσουν σε περιβάλλον Windows τα Fedora Directory Server και Red Hat Directory Server.

Ενσωματωμένος μηχανισμός ελέγχου πρόσβασης (ACLs)*:* Ο ενσωματωμένος μηχανισμός ελέγχου πρόσβασης επιτρέπει τον έλεγχο πρόσβασης βάση χαρακτηριστικών που ανήκουν σε αντικείμενα ενσωματωμένα μέσα στην βάση καταλόγου. Από όλα τα παρουσιαζόμενα λογισμικά όλα υποστηρίζουν τον μηχανισμό αυτό.

Δυνατότητα επέκτασης του σχήματος: Όλα τα υπό παρουσίαση λογισμικά επιτρέπουν εύκολη επέκταση σχήματος με προσθήκη χαρακτηριστικών γνωρισμάτων και κλάσεων στα υπάρχοντα αντικείμενα απαραίτητη προϋπόθεση για την ομαλή κλιμάκωση και εξέλιξη της υπηρεσίας.

Δυνατότητα απομακρυσμένης διαχείρισης μέσω κονσόλας και/ή LDAP: Η δυνατότητα απομακρυσμένης διαχείρισης είτε μέσω εξειδικευμένου λογισμικού ή/και λειτουργικών LDAP θεωρείται απαραίτητη λόγω της κατακόρυφης μείωσης του διαχειριστικού κόστους. Και τα επτά(7) λογισμικά παρουσιάζουν την εν λόγω δυνατότητα παρέχοντας δυνατότητα αποθήκευσης των ρυθμίσεων στην ίδια την υπηρεσία καταλόγου.

Υποστήριξη multi-master replication: Για την μελλοντική επέκταση της υπηρεσίας καταλόγου θεωρείται σημαντική η δυνατότητα για multi-master replication που θα επιτρέπει σε περισσότερους του ενός κεντρικούς εξυπηρετητές να λειτουργούν σε κατάσταση master χωρίς απώλεια δεδομένων ή/και συγχρονισμού. Το λογισμικό OpenLDAP ακολουθεί διαφορετική θεώρηση στον τομέα του replication με την υποστήριξη ενός νέου πρωτοκόλλου (SyncRepl), το mirrormode host standby master (στη διαμόρφωση αυτή ένας εξυπηρετητής είναι πάντα master και ένας άλλος λειτουργεί ως hot standby σε mirror mode έτοιμος να αναλάβει όλες τις λειτουργίες σε περίπτωση δυσλειτουργίας του master). Η έκδοση 2.4 παρέχει υποστήριξη για multimaster replication.

Υποστήριξη μεγάλου αριθμού εγγραφών: Η ανάγκη για υψηλής απόδοσης υποστήριξη μεγάλου αριθμού εγγραφών καθορίζουν την επιλογή του κατάλληλου λογισμικού. Τα υπό μελέτη λογισμικά παρουσιάζουν δυνατότητες υποστήριξης αριθμού εγγραφών μεγαλύτερο του ενός εκατομμυρίου (1.000.000). Με βάση ανεξάρτητες δοκιμές επιδόσεων τα λογισμικά OpenLDAP και Sun ONE φαίνεται να προσφέρουν τις μεγαλύτερες επιδόσεις από πλευράς αριθμού ταυτόχρονων λειτουργιών και υποστήριξης μεγάλου αριθμού εγγραφών.

Συγχρονισμός με Active Directory: Δεδομένης της πιθανής ανάγκης στο μέλλον για διαλειτουργικότητα με την υπηρεσία Active Directory της εταιρείας Microsoft θεωρείται επιθυμητή η δυνατότητα συγχρονισμού με την εν λόγω υπηρεσία. Πέραν του λογισμικού Active Directory που προσφέρει την δυνατότητα εγγενώς και του λογισμικού OpenLDAP που δεν προσφέρει εγγενώς την εν λόγω δυνατότητα, όλα τα υπό μελέτη λογισμικά έχουν την δυνατότητα να συγχρονιστούν με Active Directory.

Γραφικό περιβάλλον διαχείρισης: Τέλος, πλεονέκτημα θεωρείται η ύπαρξη γραφικού περιβάλλοντος διαχείρισης. Από τα υπό μελέτη συστήματα ο OpenLDAP δεν προσφέρει εγγενώς τη συγκεκριμένη δυνατότητα. Παρόλα αυτά καθώς το σύνολο της διαμόρφωσης είναι αποθηκευμένο σε LDAP η διαχείριση του συστήματος είναι δυνατόν να γίνει απευθείας με τη χρήση κάποιου από τα διαθέσιμα λογισμικά τύπου LDAP browser.

4.3 Τελική επιλογή συστήματος

Τα προϊόντα έχουν παρεμφερείς ιδιότητες με εξαίρεση τον OpenLDAP ο οποίος υπολείπεται σε σχέση με τα εμπορικά προϊόντα σε λειτουργίες όπως το συγχρονισμό χρηστών και ομάδων μεταξύ OpenLDAP και Active Directory,και την ύπαρξη γραφικού περιβάλλοντος διαχείρισης. Ο Fedora Directory Server παρουσιάζεται συγκριτικά πλήρες σε σχέση με τα υπόλοιπα εμπορικά προϊόντα, καθώς προέρχεται από τον Netscape Directory Server και διαθέτει το σχεδιασμό, τις βασικές αλλά και τις προηγμένες λειτουργίες του. Επιπλέον η ανάπτυξη του υποστηρίζεται από την κοινότητα ανοιχτού κώδικα και διαθέτει την υποστήριξη της Red Hat Systems. Από τα εμπορικά προϊόντα ξεχωρίζουν ο Sun Java System Directory Server, ο IBM Tivoli

Directory Server και Novell eDirectory καθώς υποστηρίζουν όλες τις βασικές αλλά και αρκετές προηγμένες λειτουργίες.

Εκτός από τα παραπάνω στοιχεία σημαντικό στοιχείο που λήφθηκε υπόψη στην τελική επιλογή ήταν το τρέχον ποσοστό χρήσης κάθε συστήματος στις εγκαταστάσεις υπηρεσίας καταλόγου στην ακαδημαϊκή κοινότητα. Επιπλέον δόθηκε μεγάλη βαρύτητα στη συνολική απόδοση κάθε συστήματος, στο βαθμό υποστήριξης του από την κοινότητα, στην ωριμότητα του λογισμικού και στις προοπτικές περαιτέρω ανάπτυξης κάθε συστήματος λογισμικού.

Λαμβάνοντας υπόψη όλα τα παραπάνω, για την υλοποίηση της υπηρεσίας καταλόγου επιλέγουμε το λογισμικό υπηρεσίας καταλόγου OpenLDAP, το οποίο αποτελεί το πλέον διαδεδομένο λογισμικό υπηρεσιών καταλόγου παγκοσμίως, έχει συνεχή υποστήριξη από την κοινότητα, αδιάκοπη ανάπτυξη ενώ με βάση πλειάδα μελετών παρουσιάζει τις καλύτερες επιδόσεις ανάμεσα στα διαθέσιμα συστήματα ανοικτού λογισμικού.

4.4 Διεπαφή (interface)

Η εφαρμογή που θα αναπτυχθεί υλοποιεί ένα προγραμματιστικό interface κλήσης κατάλληλων λειτουργιών εγγραφών υπό την μορφή συναρτήσεων. Ο ορισμός του interface αυτού γίνεται με τη χρήση της γλώσσας προδιαγραφής Web Services WSDL (Web Service Description Language) την XML γραμματική δηλαδή που έχει προδιαγραφεί από το W3 Consortium για την υλοποίηση και περιγραφή web services. Η εκτέλεση των web services πραγματοποιείται με τη χρήση SOAP πάνω από το πρωτόκολλο HTTP, παρέχοντας έτσι συμβατότητα με την πλειοψηφία των αντίστοιχων πακέτων λογισμικού (.Net FrameWork, Java κτλ).

Για κάθε εξωτερική υπηρεσία που πρόκειται να προσπελάσει την εφαρμογή web service θα δημιουργηθεί κατάλληλη σελίδα που υλοποιεί το σύνολο των απαιτούμενων συναρτήσεων-λειτουργιών καθώς και η αντίστοιχη προδιαγραφή σε WS + XML. Η πρόσβαση θα πραγματοποιείται με τη χρήση περιορισμού πρόσβασης με βάση την IP του αιτούντα (ως αιτών ορίζεται ο εξυπηρετητής που φιλοξενεί την αντίστοιχη εξωτερική υπηρεσία) καθώς και κατάλληλου ζευγαριού username/password (HTTP Authentication) ανά υπηρεσία ενώ θα προστατεύεται με

χρήση του πρωτοκόλλου HTTPS. Η WSDL προδιαγραφή του κάθε web service θα παρέχεται με κατάλληλη κλήση της σελίδας ως εξής:
http://<web server>/<web service>?wsdl

4.4.1 Αρχιτεκτονική

Η εφαρμογή που θα αναπτυχθεί θα εγκατασταθεί σε κατάλληλο εξυπηρετητή της υπηρεσίας. Για κάθε υπηρεσία που αιτείται web service θα δημιουργούνται δύο αρχεία. Ένα δημοσίως προσβάσιμο στο οποίο θα πραγματοποιείται ο ορισμός όλων των συναρτήσεων-λειτουργιών του web service και ο σκελετός κλήσης τους και ένα δεύτερο ιδιωτικό στο οποίο θα πραγματοποιείται η υλοποίηση της κάθε λειτουργίας. Κάθε web service θα πρέπει να πραγματοποιεί πλήρη καταγραφή των καλούμενων λειτουργιών με διατήρηση των ακόλουθων στοιχείων:

- Ημερομηνία και ώρα κλήσης λειτουργίας
- Όνομα λειτουργίας
- Αιτών (IP Address)
- Εγγραφή στην οποία ζητείται η πραγματοποίηση της λειτουργίας (employeenumber ή άλλος μοναδικός δείκτης ανά περίπτωση)
- MD5 δεδομένων
- Αποτέλεσμα εκτέλεσης λειτουργίας
- Τυχόν μήνυμα λάθους

4.4.2 Πλατφόρμα λογισμικού

Καθώς τα web services θα είναι προσβάσιμα μέσω web (HTTP) η επιλογή της πλατφόρμας ανάπτυξης θα πρέπει να γίνει μεταξύ των διαθέσιμων γλωσσών προγραμματισμού web εφαρμογών. Από την επισκόπηση των διαθέσιμων τεχνολογιών προκύπτουν οι εξής διαθέσιμες γλώσσες προγραμματισμού:

- PHP. Μία ισχυρή, ανοικτού κώδικα και επεκτάσιμη υψηλού επιπέδου γλώσσα προγραμματισμού σε χρήση κυρίως σε περιβάλλοντα Unix/Apache. Μέσω

επεκτάσεων η γλώσσα παρέχει τη δυνατότητα εκτέλεσης πληθώρας λειτουργιών και χρήσης βάσεων δεδομένων, πλήθους πρωτοκόλλων κτλ.

- Java.
- ASP.NET

Στην περίπτωση της παρούσης διπλωματικής λόγω χρήσης πλατφόρμας Unix και με σκοπό την εύκολη και γρήγορη των εφαρμογών επιλέχθηκε η χρήση της γλώσσας PHP. Η PHP είναι μία υψηλού επιπέδου γλώσσα συγγραφής web based εφαρμογών βασισμένη στη χρήση ποικιλίας modules για την εκτέλεση πολύπλοκων λειτουργιών και την επικοινωνία με εξωτερικά συστήματα (βάσεις δεδομένων, LDAP servers κτλ). Επιτρέπει την ταχύτατη και εύκολη ανάπτυξη επεκτάσιμων εφαρμογών με μικρό κόστος.

Σε επίπεδο απόδοσης ο PHP interpreter ενσωματώνεται ως module στον Apache server με συνέπεια να εκμηδενίζεται το κόστος εκτέλεσης του ενώ αξιοποιεί τα apache server threads ώστε να παρέχει πολλαπλά instances για εκτέλεση των δυναμικών σελίδων. Παράλληλη χρήση PHP accelerators ώστε η μεταγλώττιση των PHP σελίδων να πραγματοποιείται μόνο μία φορά καθώς και εξωτερικών memory caching μηχανισμών για τη διατήρηση περιβαλλοντικών μεταβλητών αφαιρεί οποιοδήποτε επιπλέον εμπόδιο στην επίτευξη υψηλής απόδοσης και κλιμάκωσης.. Στην συγκεκριμένη περίπτωση έχει εκγατασταθεί (με την μορφή port – pecl package) ο APC Accelerator.

Μετά από δοκιμή των διαθέσιμων πακέτων ανοικτού λογισμικού επιλέχθηκε η χρήση του πακέτου **nuSoap** με γνώμονα ανάμεσα στα άλλα την ευκολία εγκατάστασης και ανάπτυξης εφαρμογών. Το πακέτο nuSoap λειτουργεί σε περιβάλλον γλώσσας προγραμματισμού εφαρμογών web PHP και παρέχει τη δυνατότητα εύκολης και γρήγορης ανάπτυξης εφαρμογών web services με το ελάχιστο κόστος προγραμματισμού. Παράλληλα, η χρήση της γλώσσας PHP δίνει τη δυνατότητα χρήσης ήδη διαθέσιμων πακέτων διασύνδεσης με μία ποικιλία πρωτοκόλλων και υπηρεσιών που επιτρέπουν τη γρήγορη και εύκολη ανάπτυξη πολύπλοκων

εφαρμογών. Το πακέτο NuSOAP είναι βασισμένο πάνω στο SOAPx4 και αντιγράφει πολλές από τις εφαρμογές του. Παρέχεται από την NuSphere και τον Dietrich Ayala. Είναι μια σειρά από PHP classes (δίχως να χρειάζονται PHP extensions) που επιτρέπει στους προγραμματιστές να δημιουργήσουν και να χρησιμοποιήσουν web services που είναι βασισμένες σε SOAP 1.1, WSDL 1.1 και HTTP 1.0/1.1.

4.5 Περιγραφή nuSoap API

Η διεπαφή προγραμματισμού εφαρμογών του nuSoap παρέχει συναρτήσεις για τη συγγραφή τόσο πελατών όσο και παρόχων υπηρεσιών ιστού.

Το παρακάτω παράδειγμα έχει κάποιες απλές συναρτήσεις που βρίσκονται μέσα στο αρχείο 'lib/nusoap.php'.

Πριν από οτιδήποτε άλλο απαιτείται η συμπερίληψη (include) του αρχείου *nusoap.php* το οποίο και περιλαμβάνει τους ορισμούς (και τον κώδικα) όλων των συναρτήσεων του πακέτου.
Η δημιουργία πελάτη (client) web services είναι αρκετά απλή και απαιτεί μόνο τα ακόλουθα βήματα:

- Κλήση της συνάρτησης δημιουργίας *new soapclient* με τη web διεύθυνση του παρόχου web service.
- Κλήση της συνάρτησης *call()* με τα κατάλληλα ορίσματα για την κλήση κάθε συναρτησιακής διεπαφής (function) που παρέχει ο πάροχος και χρήση των αποτελεσμάτων που επιστρέφονται.

Οι ορισμοί των παραπάνω συναρτήσεων είναι οι εξής:

```
soapclient($url or $wsdl, boolean is wsdl)


call($operation, $params, $namespace='http://tempuri.org', $soapAction='',
$headers=false, $rpcParam=null (not used), $style='rpc|document' (default is rpc),
$use='encoded|literal' (what to use when serializing parameters))
```

4.5.1 Παράδειγμα πελάτη για υπηρεσίες ιστού

(ο παρακάτω κώδικας βρίσκεται στο αρχείο «arithmetic.php»)

```
<?php
require_once('lib/nusoap.php');
$client = new soapclient('http://localhost/ws/arithmetic.php');

echo "<br><b>" . date('r') . "</b><br>\n";
$action = $_REQUEST['action'];
if ($action == 'add'){
echo "<br><b>AddNumbers</b><br>\n";
$result = $client->call('AddNumbers',
      array('First' => '10', 'Second' => '4'));
print_r($result);
}
if ($action == 'sub'){
echo "<br><b>SubstractNumbers</b><br>\n";
$result = $client->call('SubstractNumbers',
      array('First' => '10', 'Second' => '4'));
print_r($result);
}
?>
```

Η δημιουργία παροχέα web services είναι και αυτή απλή αλλά με μεγαλύτερες δυνατότητες:

- Κατ' αρχήν απαιτείται η αρχικοποίηση του παροχέα με κλήση της *new soap_server();*
- Εν συνεχεία πρέπει να οριστεί το WSDL και το πεδίο ονοματολογίας της υπηρεσίας ιστού. Για το WSDL ορίζεται μόνο το URL που θα το περιέχει καθώς αυτό δημιουργείται αυτόματα με βάση τις συναρτήσεις που θα οριστούν. Για τους ορισμούς χρησιμοποιούνται οι κλήσεις configureWSDL και schemaTargetNamespace.

- Το βασικό κομμάτι είναι ο ορισμός των συναρτήσεων που θα παρέχει η υπηρεσία ιστού με χρήση της κλήσης *register* στην οποία παρέχεται το όνομα της κάθε συνάρτησης, η είσοδος και έξοδος, το σχήμα καθώς και τυχόν σχόλια για τη λειτουργία της συνάρτησης.
- Τέλος, ο παροχέας ενεργοποιείται με την κλήση της *service().*

Οι ορισμοί των συναρτήσεων είναι ως εξής:

```
     * @param    string $data usually is the value of $HTTP_RAW_POST_DATA
     * @access   public
     */
     function service($data)


     * register a service function with the server
     *
     * @param    string $name the name of the PHP function, class.method or
class..method
     * @param    array $in assoc array of input values: key = param name, value =
param type
     * @param    array $out assoc array of output values: key = param name, value =
param type
     * @param       mixed $namespace the element namespace for the method or false
     * @param       mixed $soapaction the soapaction for the method or false
     * @param       mixed $style optional (rpc|document) or false Note: when
'document' is specified, parameter and return wrapp
ers are created for you automatically
     * @param       mixed $use optional (encoded|literal) or false
     * @param       string $documentation optional Description to include in WSDL
     * @param       string $encodingStyle optional (usually
'http://schemas.xmlsoap.org/soap/encoding/' for encoded)
     * @access   public
     */
```

```
    function
register($name,$in=array(),$out=array(),$namespace=false,$soapaction=false,$style=
false,$use=false,$documentation='
',$encodingStyle=)



  /**
  * Sets up wsdl object.
  * Acts as a flag to enable internal WSDL generation
  *
  * @param string $serviceName, name of the service
  * @param mixed $namespace optional 'tns' service namespace or false
  * @param mixed $endpoint optional URL of service endpoint or false
  * @param string $style optional (rpc|document) WSDL style (also specified by
operation)
  * @param string $transport optional SOAP transport
  * @param mixed $schemaTargetNamespace optional 'types' targetNamespace for
service schema or false
  */
  function configureWSDL($serviceName,$namespace = false,$endpoint =
false,$style='rpc', $transport = 'http://schemas.xmlsoap.org/
soap/http', $schemaTargetNamespace = false)
```

Παράδειγμα παροχέα υπηρεσιών ιστού (απλές συναρτήσεις αριθμητικής):

```
<?php
require_once('lib/nusoap.php');

$server = new soap_server();
$server->configureWSDL('Arithmetic',"http://localhost/ws/arithmetic.php");
$server->wsdl->schemaTargetNamespace="http://localhost/ws/arithmetic.php?wsdl";
$server->register('AddNumbers',
      array('First' => 'xsd:string','Second' => 'xsd:integer'),
      array('Status' => 'xsd:boolean','Result' => 'xsd:string'),
```

```
        "http://<hostname>/ws/arithmetic.php");
$server->register('SubstractNumbers',
        array('First' => 'xsd:string','Second' => 'xsd:integer'),
        array('Status' => 'xsd:boolean', 'Result' => 'xsd:string'),
        "http://<hostname>/ws/arithmetic.php");
function AddNumbers($First,$Second)
{
        $Result = $First + $Second;
        return array('Status' => '1', 'Result' => $Result);
}
function SubstractNumbers($First,$Second)
{
        $Result = $First - $Second;
        return array('Status' => '1', 'Result' => $Result);
}
$server->service($HTTP_RAW_POST_DATA);
?>
```

Η σελίδα ιστού που δημιουργείται στον παροχέα εκτός από διεπαφή τύπου SOAP για εκτέλεση υπηρεσιών ιστού είναι απευθείας προσβάσιμη και μέσω διαδικτύου. Στην περίπτωση αυτή παρέχει γενική και ειδική εποπτεία των παρεχόμενων συναρτήσεων καθώς και δυνατότητα μεταφόρτωσης του WSDL που περιγράφει τη λειτουργία τους. Το WSDL είναι προσβάσιμο στη σελίδα που ορίστηκε κατά την κλήση της συνάρτησης *configureWDSL()* (συνήθως http://<web service url>?wsdl

5 Πιστοποίηση και Πολιτική Ασφάλειας

Καθώς με χρήση των web services μεταφέρεται ευαίσθητη πληροφορίας (όπως προσωπικά στοιχεία χρηστών, password κτλ) είναι απαραίτητο η χρήση τους να προστατεύεται με τη χρήση ασφαλούς πρωτοκόλλου επικοινωνίας HTTPS. Παράλληλα, καθώς οι πελάτες της υπηρεσίας είναι συγκεκριμένοι και περιορισμένοι (συγκεκριμένες web εφαρμογές) προτείνεται η προσθήκη προστασίας με περιορισμό

πρόσβασης στις σελίδες των web services μόνο από τις διευθύνσεις IP των εξυπηρετητών web στις οποίες εκτελούνται οι web εφαρμογές των πελατών.
Η πιστοποίηση βασίζεται στη χρήση του μηχανισμού HTTP Authentication κατά την κλήση των web services. Ο καλών παρέχει κατάλληλο ζεύγος username/password για την πρόσβαση του στην υπηρεσία. Το ζεύγος αυτό πιστοποιείται με κατάλληλο μηχανισμό. Ο μηχανισμός αυτός μπορεί να είναι ένα απλό password file αλλά προτείνεται να πραγματοποιείται απευθείας πιστοποίηση από την υπηρεσία καταλόγου με χρήση των αντίστοιχων μεθόδων που παρέχονται από τον εξυπηρετητή web. Παρακάτω φαίνεται παράδειγμα ενεργοποίησης πιστοποίησης μέσω LDAP στον εξυπηρετητή apache με χρήση του apache module mod_auth_ldap:

```
<Directory "/usr/local/www/ws">
  AllowOverride None
  Options MultiViews Indexes FollowSymlinks
  AddType application/x-httpd-php .php .php3

  AuthType Basic
  AuthName "web-services"
  AuthLDAPURL ldap://localhost/ou=people,o=ntua,c=gr?uid
  Require valid-user
  Satisfy all
  Order deny,allow
  Deny from all
  Allow from 147.102.220.0/24
</Directory>
```

Μετά την επιτυχή πιστοποίηση του χρήστη τα στοιχεία του (username,password) είναι διαθέσιμα με την μορφή server μεταβλητών στα web services. Το username του χρήστη μπορεί να αντιστοιχιστεί με κατάλληλο τρόπο στο DN του. Αυτό μπορεί να γίνει με δύο τρόπους:

A. Στην περίπτωση στην οποία όλες οι εγγραφές χρηστών είναι κάτω από το ίδιο υποδέντρο (πχ ou=people,dc=<institution>,dc=gr), τότε η αντίστοιχιση μπορεί να γίνει άμεσα με τη φόρμουλα dn: uid=<username>,<dn υποδέντρου>.

B. Στην περίπτωση στην οποία η μορφή του DN του χρήστη δεν μπορεί να αντιστοιχιστεί απευθείας, είναι δυνατόν να πραγματοποιηθεί πρώτα αναζήτηση στην υπηρεσία καταλόγου με βάση το username του χρήστη και από την επιστρεφόμενη εγγραφή να προκύψει το DN. Η αναζήτηση μπορεί να γίνει με τα ακόλουθα στοιχεία:
    - Base: DN κάτω από το οποίο είναι αποθηκευμένες οι εγγραφές των χρηστών
    - Scope: Subtree
    - Filter: (uid=<username>)
    - Επιστρεφόμενα attributes: uid. Καθώς μας ενδιαφέρει μόνο το DN της εγγραφής δεν χρειάζεται να αιτηθούμε να επιστραφεί κάποιο attribute της εγγραφής.

    Εάν η αναζήτηση επιστρέψει περισσότερες από μία εγγραφές τότε θεωρούμε ότι απέτυχε και το web service μπορεί να επιστρέψει κατάλληλο μήνυμα λάθους στον καλώντα.

Από τη στιγμή που είναι διαθέσιμο το DN/password του χρήστη, τότε μπορεί να χρησιμοποιηθεί αυτό κατά την πραγματοποίηση αλλαγών στην υπηρεσία καταλόγου όπως για παράδειγμα στην αλλαγή του password του χρήστη. Αυτό έχει ως συνέπεια ότι:

- Δεν απαιτείται η δημιουργία ειδικού χρήστη με πλήρη δικαιώματα αλλαγών σε όλο το δέντρο των χρηστών
- Η αλλαγή στην εγγραφή ενός χρήστη ουσιαστικά πραγματοποιείται από τον ίδιο το χρήστη, κάτι που επιτρέπει τη σωστή, μινιμαλιστική και καθαρή διαμόρφωση της πολιτικής ασφάλειας της υπηρεσίας καταλόγου καθώς απαιτείται απλά ο ορισμός των attributes τα οποία έχει το δικαίωμα να αλλάξει ο χρήστης στην δική του εγγραφή

Ο παραπάνω μηχανισμός είναι χρήσιμος και βέλτιστος στην περίπτωση αλλαγών στην ίδια την εγγραφή του χρήστη (όπως αλλαγή password, στοιχείων υπηρεσιών όπως mail alias κτλ) αλλά υπάρχουν περιπτώσεις στις οποίες δεν κρίνεται σκόπιμο να χρησιμοποιηθεί όπως:

- Αλλαγή στοιχείων χρήστη τα οποία δε θα πρέπει να έχει το δικαίωμα αλλαγής τους ο ίδιος όπως το επίσημο ονοματεπώνυμο, δικαιώματα χρήσης και πρόσβασης σε υπηρεσίες, όρια χρήσης κτλ
- Προσθήκη και διαγραφή εγγραφών

Για τις περιπτώσεις αυτές είναι σκόπιμο να δημιουργηθεί χρήστης με κατάλληλα δικαιώματα ο οποίος θα χρησιμοποιείται από τον καλών. Ο χρήστης αυτός προτείνεται να έχει όνομα που θα αποκαλύπτει απευθείας το ρόλο του (πχ web-services-user) καθώς και να χρησιμοποιείται αποκλειστικά και μόνο για το σκοπό αυτό.

Το username/password του χρήστη μπορεί να χρησιμοποιηθεί στις περιπτώσεις όπου χρήστης της καλούσας εφαρμογής είναι ο ίδιος ο χρήστης, ο οποίος και αιτείται την αλλαγή για την οποία γίνεται η κλήση του web service. Η περίπτωση του μοναδικού χρήστη με πλήρη δικαιώματα συνήθως χρησιμοποιείται στις περιπτώσεις αυτόματου συγχρονισμού μεταξύ ετερογενών βάσεων δεδομένων χρηστών. Η πρωτογενής πληροφορία για τους χρήστες μίας εφαρμογής μπορεί να συντηρείται σε μία βάση δεδομένων και ο συγχρονισμός με την υπηρεσία καταλόγου να πραγματοποιείται με τη χρήση web services.

Ο παραπάνω μηχανισμός παρουσιάζει το εγγενές μειονέκτημα ότι απαιτεί την μεταφορά και χρήση των passwords των χρηστών. Η τρέχουσα πρακτική στην πιστοποίηση χρηστών σε σελίδες web προβλέπει τη χρήση τεχνολογίας Single Sign On η οποία επιτρέπει την πιστοποίηση του χρήστη σε ένα κεντρικό σημείο και την αποφυγή μεταφοράς των διαπιστευτηρίων του ανάμεσα σε εφαρμογές. Οι δορυφορικές εφαρμογές εμπιστεύονται την κεντρική υπηρεσία πιστοποίησης και λαμβάνουν ψηφιακά υπογεγραμμένη την ταυτότητα του χρήστη.
Στην περίπτωση των web services η χρήση Single Sign On δεν είναι δυνατή για τους ακόλουθους λόγους:

- Οι λειτουργίες σε πολλές περιπτώσεις πραγματοποιούνται offline και χωρίς την άμεση αλληλεπίδραση του χρήστη (πχ συγχρονισμός μεταξύ ετερογενών βάσεων δεδομένων μία φορά την μέρα).

- Η πιστοποίηση στην υπηρεσία καταλόγου (LDAP Bind) απαιτεί τη χρήση κατάλληλου ζεύγους Bind DN/password με συνέπεια να είναι απαραίτητα κατάλληλα διαπιστευτήρια.

Για την υπέρβαση του μειονεκτήματος αυτού μπορεί απλά να μη χρησιμοποιούνται τα διαπιστευτήρια των χρηστών αλλά ένας κεντρικός λογαριασμός με αυξημένα δικαιώματα στο δέντρο των χρηστών. Η web εφαρμογή – πελάτης των web services μπορεί να πιστοποιήσει μέσω Single Sign On τον χρήστη και στη συνέχεια να αποστείλει την αίτηση εκ μέρους του παρέχοντας τα διαπιστευτήρια του κεντρικού λογαριασμό για πιστοποίηση στο web service. Μετά τη σύνδεση στην υπηρεσία καταλόγου (με τα στοιχεία του κεντρικού λογαριασμού) μπορεί να χρησιμοποιηθεί ο μηχανισμός LDAP Proxy Authentication ώστε οι αλλαγές να πραγματοποιηθούν εκ μέρους του χρήστη για τον οποίο αιτούνται (ο κεντρικός λογαριασμός έχει δικαίωμα πραγματοποίησης LDAP Proxy Auth για όλους τους χρήστες του δέντρου). Κατ' αυτόν τον τρόπο απαιτείται μόνο να επιτραπεί η χρήση του LDAP Proxy Auth ενώ κατά τα άλλα η πολιτική ασφάλειας (ACL) της υπηρεσίας καταλόγου παραμένει η ίδια. Οι αιτήσεις προς την υπηρεσία ουσιαστικά εμφανίζεται να προέρχονται από τους ίδιους τους χρήστες με συνέπεια η πρόσβαση να καθορίζεται από τα δικαιώματα αλλαγής που έχουν οι χρήστες στις εγγραφές τους.

5.1 Προτάσεις Βελτίωσης

Παρακάτω παρατίθενται ορισμένες προτάσεις βελτίωσης και εξέλιξης των λειτουργιών μέσω web services ώστε να μπορούν να λειτουργήσουν καλύτερα σε πραγματικά περιβάλλοντα λειτουργίας.

5.1.1 Versioning ερωτημάτων/απαντήσεων

Μία πρόταση είναι η ενεργοποίηση υποστήριξης εκδόσεων (versioning) στα Web Services. Συνήθως αυτή η λειτουργικότητα υποστηρίζεται με την αλλαγή του targetnamespace (<xsd:schema targetNamespace="<namespace>") στο WSDL αρχείο του Web Service σε κάθε μεγάλη αλλαγή (προσθήκη/αφαίρεση/αλλαγή κλήσης συναρτήσεων) με κατάλληλη διαφοροποίηση και του soap address location (<soap:address location="http://<URL>"/>). Κατά αυτόν τον τρόπο είναι δυνατόν να δημιουργηθούν διαφορετικά WSDL αρχεία για κάθε major version του Web Service.

Κάθε WSDL θα διαθέτει διαφορετικό targetnamespace και soap address location ώστε να είναι δυνατή η υποστήριξη διαφορετικών εκδόσεων πελατών οι οποίοι θα έχουν πρόσβαση στη λειτουργικότητα που προσφέρει η κάθε έκδοση. Μάλιστα σε περίπτωση που η διαφοροποίηση μεταξύ των εκδόσεων είναι μόνο η προσθήκη νέων συναρτήσεων (χωρίς μεταβολή των ήδη υπαρχόντων) τότε είναι δυνατό το ίδιο το Web Service να παραμένει σταθερό και να μεταβάλλεται μόνο το WSDL το οποίο θα περιλαμβάνει τις συναρτήσεις που είναι διαθέσιμες σε κάθε έκδοση. Έτσι διαφορετικοί πελάτες μπορούν να συνδέονται στο ίδιο WS αλλά να απολαμβάνουν διαφορετικές δυνατότητες (καλώντας τις επιπλέον συναρτήσεις).

Ακόμα και η προσθήκη επιπλέον παραμέτρων σε ήδη υπάρχουσες συναρτήσεις (χωρίς μεταβολή των υπολοίπων arguments) είναι δυνατή διατηρώντας ουσιαστικά μία έκδοση του Web Service. Για το σκοπό αυτό, κατά τον αρχικό ορισμό των συναρτήσεων θα προστίθεται ένα τελευταίο όρισμα τύπου xsd:any το οποίο θα μπορεί να χρησιμοποιηθεί για την μελλοντική επέκταση των ορισμάτων της συνάρτησης χωρίς να δημιουργείται πρόβλημα στη λειτουργικότητα που θα παρέχεται σε πελάτες που δε θα υποστηρίξουν τα νέα ορίσματα. Απλά το νέο WSDL θα έχει ελαφρώς διαφοροποιημένο το soap address location με ένα URL της μορφής ‘http://<old-url>?version=1.1’. Με άλλα λόγια η κλήση του WS θα προσφέρει άμεση γνώση της έκδοσης που ζητείται και οι συναρτήσεις θα χρησιμοποιούν (ή όχι) τα επιπλέον ορίσματα.

5.1.2 Χρήση επεκτάσεων (WS-*)

Σε ένα πραγματικό σύστημα είναι απαραίτητο να έχουμε λειτουργίες ασφαλείας που καθορίζουν την πολιτική της εφαρμογής όσον αφορά την επικοινωνία με πολλαπλούς χρήστες σε απομακρυσμένα σημεία. Για αυτό τον λόγο έχουν αναπτυχθεί ορισμένες προδιαγραφές ή βρίσκονται ακόμα σε ανάπτυξη για την επέκταση των δυνατοτήτων των Web Services. Αυτές οι προδιαγραφές αναφέρονται γενικά ως WS-*. Μερικές από τις προδιαγραφές είναι οι εξής:

- WS-Security

Προσδιορίζει τον τρόπο χρήσης κρυπτογράφησης XML και XML Υπογραφής (XML Encryption, XML Signature) στο SOAP φάκελο για την ασφαλή ανταλλαγή μηνυμάτων, ως εναλλακτική λύση ή παράλληλα με τη χρήση HTTPS (η οποία παρέχει ασφάλεια στο κανάλι της επικοινωνίας).

Περιγράφει πώς επισυνάπτονται υπογραφές και κρυπτογραφούνται οι κεφαλίδες στα SOAP μηνύματα. Επιπλέον, περιγράφει πώς προστίθενται στα μηνύματα επιπλέον δυνατότητες ασφαλείας όπως πιστοποιητικά X.509 και Kerberos tickets.

Η WS-Security ενσωματώνει χαρακτηριστικά ασφαλείας στην κεφαλίδα ενός μηνύματος SOAP και λειτουργεί στο επίπεδο εφαρμογών (application layer). Έτσι εξασφαλίζεται end-to-end ασφάλεια σε αντίθεση με την τεχνολογία HTTPS που εξασφαλίζει ασφάλεια στο κανάλι μεταξύ των κόμβων (στην περίπτωση που η κλήση των WS γίνεται με τη μεσολάβηση περισσότερων από ένα κόμβων).

- WS-Transaction

Μια προδιαγραφή Web Υπηρεσιών που περιγράφει τους τύπους συντονισμού..

- WS-Addressing

Μια προδιαγραφή που καθορίζει τον τρόπο εισαγωγής διευθύνσεων στην κεφαλίδα του SOAP. Με αυτήν την προδιαγραφή περιγράφεται η επικοινωνία μεταξύ των web services όταν χρειάζεται να ανταλλάσσονται διευθύνσεις.
Με αυτήν τυποποιείται ο τρόπος να συμπεριλαμβάνονται δεδομένα δρομολόγησης εντός των SOAP κεφαλίδων (headers). Αντί να στηρίζονται τα web services στο network-level επίπεδο δικτύου για να μεταφέρουν πληροφορίες δρομολόγησης, χρησιμοποιώντας το WS-Addressing μπορεί να περιλαμβάνουν τα δικά τους δεδομένα δρομολόγησης σε τυποποιημένη κεφαλίδα SOAP.
Το επίπεδο δικτύου είναι υπεύθυνο μόνο για την παράδοση του μηνύματος σε έναν αποστολέα που μπορεί να υπολογίσει τα WS-Addressing μεταδεδομένα. Μόλις το μήνυμα φτάνει στο αποστολέα που ορίζεται από την URI, η εργασία του επίπεδου δικτύου έχει τελειώσει.

Το WS-Addressing υποστηρίζει τη χρήση της ασύγχρονης αλληλεπίδρασης με τον καθορισμό μιας κοινής κεφαλίδας SOAP (WSA: ReplyTo) που περιέχει το

τελικό σημείο αναφοράς (EPR) στο οποίο πρέπει να σταλεί η απάντηση. Ο φορέας παροχής υπηρεσιών μεταδίδει το απαντητικό μήνυμα σε μια ξεχωριστή σύνδεση με το WSA: ReplyTo καταληκτικό σημείο.

Το τελικό σημείο αναφοράς, Endpoint Reference (EPR), είναι ένα XML κείμενο που παρέχει πληροφορίες χρήσιμες για να δρομολογηθεί ένα μήνυμα σε ένα Web service. Αυτό περιλαμβάνει την διεύθυνση προορισμού του μηνύματος, τυχόν πρόσθετες παραμέτρους (που ονομάζονται παράμετροι αναφοράς) που είναι αναγκαίες για να δρομολογηθεί το μήνυμα προς τον προορισμό, και προαιρετικά μεταδιδόμενα (όπως WSDL ή WS-Policy) για την υπηρεσία.

Οι ιδιότητες του μηνύματος δρομολόγησης είναι οι παρακάτω:
α) ο προορισμός του μηνύματος URI (Message destination), β) το καταληκτικό σημείο της πηγής που απέστειλε αυτό το μήνυμα EPR (Source endpoint), γ) το τελικό σημείο για το οποίο τα μηνύματα θα πρέπει να αποσταλούν (EPR) (Reply endpoint), δ) το τελικό σημείο για το οποίο τα μηνύματα σφάλματος θα πρέπει να αποσταλούν (EPR) (Fault endpoint), ε) το μοναδικό αναγνωριστικό μηνύματος (URI Unique message ID), ζ) η σχέση με τα προηγούμενα μηνύματα (Ένα ζευγάρι URIs).

Uniform Resource Identifier (URI)

Για να κατανοήσουμε καλύτερα πώς λειτουργεί ένα μήνυμα δρομολόγησης είναι αναγκαία η αναφορά στο Uniform Resource Identifier ή αλλιώς URI.

Στην πληροφορική, ένας Uniform Resource Identifier (URI) αποτελείται από μια σειρά χαρακτήρων που χρησιμοποιούνται για τον εντοπισμό ή το όνομα ενός πόρου για το Διαδίκτυο. Η ταυτοποίηση αυτή επιτρέπει την αλληλεπίδραση μεταξύ των πόρων ενός δικτύου (όπως το World Wide Web) χρησιμοποιώντας ειδικά πρωτόκολλα.

Το URL (locator - εντοπισμού) και το URN (name - όνομα) αποτελούν υποσύνολα του URI. Από τεχνικής άποψης μπορεί να καθοριστεί μια διεύθυνση URL ως URI, εκτός από τον προσδιορισμό ενός πόρου, παρέχει ένα μέσο το για

να περιγραφεί η τοποθεσία του δικτύου. Για παράδειγμα, η διεύθυνση URL http://www.papei.gr/ εντοπίζει έναν πόρο (papei home page) και σημαίνει ότι ο χρήστης μπορεί να πάρει μια αναπαράσταση του εν λόγω πόρου (όπως ο κώδικας HTML της αρχικής σελίδας του) μέσω HTTP από ένα όνομα www.papei.gr. Το Uniform Resource Name (URN) περιλαμβάνει ένα URI που προσδιορίζει έναν πόρο με βάση το όνομα σε ένα συγκεκριμένο πεδίο ονομάτων. Μπορεί να χρησιμοποιηθεί ένα URN για να αναφερθεί για έναν πόρο δίχως να δηλώνεται η θέση του ή πώς να αποκτήσουν πρόσβαση. Για παράδειγμα, το URN: ISBN :7-258-45789-2 είναι ένα URI που αναφέρει το αρχείο (πόρο) που θέλουμε, δηλαδή International Standard Book Number (ISBN), καθώς και ότι μιλάμε για ένα βιβλίο, αλλά δεν δείχνει πού και πώς μπορούμε να βρούμε ένα πραγματικό αντίγραφό του.

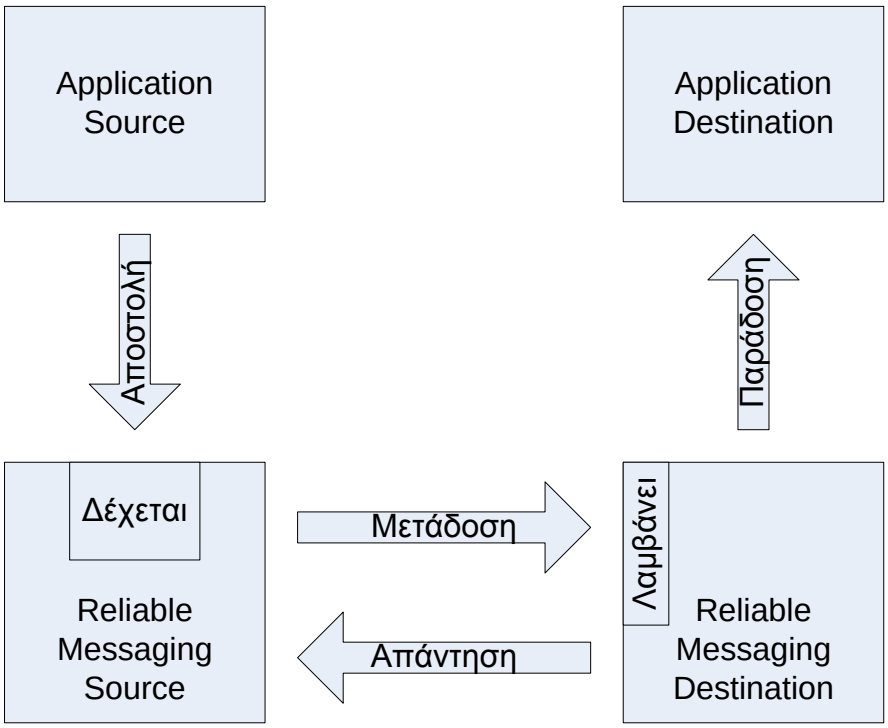
- WS-Reliability

Ένα πρωτόκολλο για αξιόπιστη ανταλλαγή μηνυμάτων μεταξύ δύο υπηρεσιών Web. Το SOAP πάνω από HTTP δεν επαρκεί, όταν ένα πρωτόκολλο ανταλλαγής μηνυμάτων πρέπει να εγγυάται κάποιο επίπεδο αξιοπιστίας και ασφάλειας. Η προδιαγραφή αυτή έχει σχεδιαστεί για χρήση σε συνδυασμό με άλλα συμπληρωματικά πρωτόκολλα και βασίζεται σε προηγούμενες προδιαγραφές, όπως π.χ., ebXML(e business) Message Service και WS-Reliable Messaging που αναλύεται παρακάτω.

- WS-ReliableMessaging

Περιγράφει ένα πρωτόκολλο που επιτρέπει στα SOAP μηνύματα να παραδοθούν αξιόπιστα μεταξύ των κατανεμημένων εφαρμογών ακόμα και όταν υπάρχει πρόβλημα στο δίκτυο.

ΤΡΟΠΟΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΔΥΟ
ΕΦΑΡΜΟΓΩΝ ΜΕΣΩ
ΑΣΦΑΛΟΥΣ ΔΡΟΜΟΥ



Μια πηγή εφαρμογής (Application Source - AS) επιθυμεί να στείλει μηνύματα με αξιοπιστία σε ένα προορισμό εφαρμογής (Application Destination - AD) πάνω από μια αξιόπιστη υποδομή. Για να επιτευχθεί αυτό κάνουν χρήση μιας αξιόπιστης πηγής μηνυμάτων (RMS) και ενός αξιόπιστου προορισμού μηνύματος (RMD). Ο AS στέλνει ένα μήνυμα προς την RMS. Η RMS χρησιμοποιεί το WS-Reliable Messaging (WS-RM) πρωτόκολλο για τη μετάδοση του μηνύματος προς τον RMD. Ο RMD παραδίδει το μήνυμα στο AD. Εάν ο RMS δεν μπορεί να μεταδώσει το μήνυμα προς τον RMD για κάποιο λόγο, θα πρέπει να υποδείξει στον AS ότι το μήνυμα δεν διαβιβάστηκε. Ο AS και ο RMS μπορούν να υλοποιηθούν εντός της ίδιας διαδικασίας ή μπορεί να υλοποιούνται σε διαφορετικά στοιχεία. Ομοίως, ο AD και ο RMD μπορούν να υλοποιηθούν εντός της ίδιας διαδικασίας ή μπορεί να υλοποιούνται σε διαφορετικά στοιχεία.

Το σημαντικό που πρέπει να αναφέρουμε είναι ότι η προδιαγραφή WS-RM ασχολείται μόνο με το περιεχόμενο και τη συμπεριφορά των μηνυμάτων που

εμφανίζονται "on the wire". Πώς στέλνονται τα μηνύματα από το AS στην RMS και πώς παραδίδονται από το RMD στο AD, αν τα μηνύματα βρίσκονται σταθερά στο δίσκο, ή περνάνε από τη μνήμη; Κανένα από αυτά δεν αποτελούν μέρος της προδιαγραφής WS-RM.

Το WS-RM πρωτόκολλο ορίζει και υποστηρίζει μια σειρά Εγγυητικών παράδοσης (Delivery Assurances). Αυτά είναι:

- AtLeastOnce - Κάθε μήνυμα θα παραδοθεί στον AD τουλάχιστον μία φορά. Αν ένα μήνυμα δεν μπορεί να παραδοθεί, ένα λάθος πρέπει να παρουσιαστεί από το RMS και / ή το RMD. Τα μηνύματα μπορούν να παραδοθούν στο AD πάνω από μία φορά (δηλαδή ο AD μπορεί να πάρει διπλά μηνύματα).
- AtMostOnce - Κάθε μήνυμα θα παραδοθεί στον AD το πολύ μία φορά. Μηνύματα μπορεί να μην παραδοθούν στο AD, αλλά ο AD ποτέ δεν θα πάρει διπλά μηνύματα.
- ExactlyOnce - Κάθε μήνυμα θα παραδοθεί στον AD ακριβώς μια φορά. Αν ένα μήνυμα δεν μπορεί να παραδοθεί, ένα λάθος πρέπει να παρουσιαστεί από το RMS και / ή το RMD. Ο AD ποτέ δεν θα πάρει διπλά μηνύματα.
- InOrder – τα μηνύματα θα πρέπει να παραδοθούν από το RMD στον AD. κατά τη σειρά που αποστέλλονται από τον AS στο RMS. Η διαβεβαίωση αυτή μπορεί να συνδυαστεί με οποιαδήποτε από τις παραπάνω εγγυήσεις.

5.2 WS-Replication

Η χρήση web services για την επικοινωνία με την υπηρεσία καταλόγου στην περίπτωση της πραγματοποίησης αλλαγών παρουσιάζει τα πλεονεκτήματα που έχουν περιγραφεί προηγουμένως. Ο τρόπος χρήσης τους όμως παρουσιάζει μία συγκεκριμένη διάκριση ανάλογα με τον πελάτη των web services:

- Η πρώτη κατηγορία είναι οι web based εφαρμογές οι οποίες χρησιμοποιούν τα web services για να εκτελέσουν λειτουργία αλλαγής η οποία έχει ζητηθεί απο χρήστη της κάθε εφαμοργής. Ο χρήστης εκτελεί εκείνη τη στιγμή την εφαρμογή και μπορεί να ανταποκριθεί άμεσα σε κάθε σφάλμα που θα παρουσιαστεί κατά την εκτέλεση της λειτουργίας που αιτήθηκε. Παράλληλα, η εκτέλεση των λειτουργιών απο τη φύση της είναι σειριακή με μεσολάβηση

ανθρώπινου παράγοντα με συνέπεια να μπορεί να ακολουθηθεί η αντίστροφη διαδικασία σε οποιαδήποτε περίπτωση λάθους. Παράδειγμα αυτής της κατηγορίας είναι web based εφαρμογές διαχείρισης στοιχείων.

- Η δεύτερη κατηγορία είναι η λειτουργία συγχρονισμού μεταξύ ετερογενών βάσεων δεδομένων και υπηρεσίας καταλόγου. Στην περίπτωση αυτή είναι απαραίτητο:
    - Ο πάροχος και ο καταναλωτής δεδομένων (βάση δεδομένων και υπηρεσία καταλόγου) να είναι εξαρχής συγχρονισμένοι ώστε οι αλλαγές να πραγματοποιούνται πάνω στις ίδιες εγγραφές.
    - Οι αλλαγές να γίνονται πάντα σειριακά και με χρονολογική σειρά καθώς σε αντίθετη περίπτωση υπάρχει το ενδεχόμενο μία αλλαγή που βασίζεται για την πραγματοποίηση της σε άλλη προγενέστερη να αποτύχει και τα δύο μέρη να αποσυγχρονιστούν.

Απο τα παραπάνω είναι προφανές ότι στη δεύτερη κατηγορία είναι εύκολο να υπάρξει αποσυγχρονισμός (ενώ υπάρχει εξαρχής απαίτηση τα δύο μέρη να εκκινούν τη διαδικασία συγχρονισμένα). Κατά συνέπεια δεν είναι επαρκές να υπάρχει διαδικασία για αποστολή αλλαγών (changelog) αλλά απαιτείται να οριστεί πλήρης διαδικασία αρχικοποίησης και επανασυγχρονισμού.

Η διαδικασία που θα οριστεί βασίζεται στη λογική του Syncronization Protocol που χρησιμοποιείται στην πλατφόρμα εξυπρετητή LDAP OpenLDAP για το συγχρονισμό των replicas. Η διαδικασία μπορεί να εφαρμοστεί πάντα ακόμα και αν δεν υπάρχει ιστορικό αλλαγών (αν και σε αυτή την περίπτωση η λειτουργία συγχρονισμού είναι πιο απλή και γρήγορη). Βασίζεται στη λειτουργία PresentEntry(). Ουσιαστικά για κάθε εγγραφή στην πάροχο βάση δεδομένων γίνεται μία παρουσιαση της στην υπηρεσία καταλόγου. Η υπηρεσία καταλόγου κάνει οποιαδήποτε αλλαγή στην εγγραφή απαιτείται, ενώ θέτει τιμή σε κατάλληλο attribute με το timestamp της αλλαγής (ουσιαστικά το timestamp της έναρξης της διαδικασίας ώστε αυτό να είναι το ίδιο για όλες τις εγγραφές). Παράλληλα, όσο χρόνο διαρκεί η διαδικασία συγχρονισμού δεν επιτρέπεται να ξεκινήσει δεύτερη διαδικασία. Με το τέλος της διαδικασίας όσες εγγραφές δεν έχουν ανανεωθεί μπορούν να διαγραφούν και τα δύο μέρη είναι πλέον συγχρονισμένα.

Σε περίπτωση που υπάρχει ιστορικό αλλαγών (changelog) αποστέλλονται μόνο οι αλλαγές και αν βρεθεί ότι τα δύο μέρη (για οποιοδήποτε λόγο) έχουν αποσυγχρονιστεί ή το ιστορικό αλλαγών δεν περιέχει όλες τις αλλαγές που απαιτούνται μπορεί να εκτελεστεί πάλι η διαδικασία πλήρους συγχρονισμού.

Για το σκοπό αυτό θα απαιτηθεί να:

- διατηρούνται ορισμένα επιπλέον στοιχεία στον πάροχο και στον καταναλωτή δεδομένων (data version, update timestamp κτλ)
- οριστούν επιπλέον web services για τη διαδικασία αρχικοποίησης/συγχρονισμού (Present Phase).

5.2.1 Πάροχος

Διατήρηση Global Data Version για ολόκληρη τη βάση: Απλώς ένας αριθμός ο οποίος αυξάνει κατά ένα σε κάθε write (add/delete/modify) operation.
Changelog: Αν διατηρούμε changelog για τα operations τότε ανά operation διατηρούμε και το αντίστοιχο dataversion.

5.2.2 Υπηρεσία Καταλόγου

dc=<root>,dc=gr
dataversion=<DataVersion>
updatetimestamp=<timestamp>
ongoingupdatetimestamp=<timestamp>

Directory Entries:
datasource=<source name>
updatetimestamp=<timestamp>

Πριν την έναρξη της διαδικασία συγχρονισμού με bulk τρόπο απαιτείται να τεθεί το datasource σε όλους τους τύπους εγγραφών οι οποίες έχουν ως πηγή δεδομένων την πάροχο βάση (για τον αρχικό συγχρονισμό των δύο βάσεων).

5.2.3 Διαδικασία

GetDataVersion(). Επιστρέφει το Data Version για όλη το δέντρο.

- Αν δεν υπάρχει changelog ή έχει χαθεί το synchronization ή το changelog δεν

περιέχει επαρκή πληροφορία ώστε να καλύψει τη διαφορά μεταξύ του Data Version του παρόχου και της υπηρεσίας καταλόγου τότε καλείται η:

- StartTotalUpdate(DataVersion, UpdateTimestamp): Το UpdateTimestamp είναι απλά το timestamp της έναρξης του update. Ενημερώνεται το ongoingupdatetimestamp στη ρίζα του δέντρου πληροφοριών. Επιστρέφει:
  - DataCurrent: No action needed
  - UpdateStarted
  - UpdateInProgress: Another update is running
  - Error
- PresentEntry(EntryQualifier, EntryType, [...Attributes...]): Ενημερώνει την εγγραφή αν απαιτείται καθώς και το updatetimestamp, datasource. Εκτελείται σειριακά για όλες τις εγγραφές στη βάση του Παρόχου.
- EndTotalUpdate(DataVersion, UpdateTimestamp): Ενημερώνει τα attributes updatetimestamp, dataversion στη ρίζα του δέντρου. Σε όλες τις εγγραφές με (datasource=<source name>) && (updatetimestamp < <UpdateTimestamp>): Ορίζονται ως 'προς διαγραφή'. Διαγραφή του ongoingupdatetimestamp

- Αν υπάρχει το changelog
  - StartUpdate(DataVersion, UpdateTimestamp): Ανανέωση του ongoingupdatetimestamp. Επιστρέφει:
    - DataCurrent: No action needed
    - UpdateStarted
    - UpdateInProgress: Another update is running
    - Error
  - Normal Operations: Εκτέλεση των ήδη ορισμένων λειτουργιών με προσθήκη του UpdateTimestamp ώς argument
  - EndUpdate(DataVersion, UpdateTimestamp): Ενημέρωση των updatetimestamp και dataversion καθώς και διαγραφή του ongoingupdatetimestamp στη ρίζα του δέντρου.

5.3 Ουρές Αναμονής

Η μέχρι τώρα περιγραφή της υλοποίησης του μηχανισμού των web services εμπεριείχε την υπόθεση ότι τα αντίστοιχα interface είναι συνεχώς διαθέσιμα ώστε κάθε κλήση των συναρτήσεων να λαμβάνει άμεσα απάντηση με μόνη καθυστέρηση τον χρόνο επεξεργασίας και εκτέλεσης των λειτουργιών. Η πραγματικότητα όμως είναι ότι τα interface είναι δυνατόν να μην είναι διαθέσιμα για περιορισμένο ή αυξημένο χρονικό διάστημα για διάφορους λόγους όπως:

- Απώλεια παροχής ρεύματος στον εξυπηρετητή που τα φιλοξενεί
- Πρόβλημα δικτύου στη διαδρομή μεταξύ πελάτη και παρόχου των interface
- Πρόβλημα λογισμικού στο λειτουργικού του εξυπηρετητή

- Πρόβλημα στη λειτουργία του λογισμικού web πάνω στο οποίο εκτελούνται τα interface

Η διαθεσιμότητα των web services μπορεί να αυξηθεί με τη χρήση διατάξεων υψηλής διαθεσιμότητας όπως UPS (για αδιάλειπτη παροχή ρεύματος), διπλά στοιχεία εξυπηρετητών (τροφοδοτικά, κάρτες δικτύου, RAID-X συστοιχίες δίσκων) και πολλαπλούς εξυπηρετητές και στοιχεία δικτύου (switches, routers).
Όλες αυτές οι διατάξεις μειώνουν την πιθανότητα μη διαθεσιμότητας της υπηρεσίας αλλά δεν την εξαφανίζουν. Παράλληλα, είναι δυνατόν παροδικά προβλήματα (πχ δικτύου) να αυξήσουν το χρόνο εξυπηρέτησης μίας λειτουργίας ή ακόμα και να οδηγήσουν στην προσωρινή αποτυχία της με συνέπεια να απαιτείται η επανεκτέλεση της.

Με βάση τα παραπάνω, είναι σημαντικό να δημιουργηθεί κατάλληλη ουρά αναμονής από την πλευρά του πελάτη κατά την κλήση των web services. Κάτι τέτοιο είναι σημαντικό ιδιαίτερα στις περιπτώσεις που δεν υπάρχει ανθρώπινη αλληλεπίδραση όπως στις περιπτώσεις αυτόματου συγχρονισμού μεταξύ βάσεων δεδομένων. Ο μηχανισμός που θα υλοποιηθεί θα πρέπει να προβλέπει κατάλληλη δομή ουράς αναμονής στην οποία θα αποθηκεύονται οι κλήσεις που θα πρέπει να πραγματοποιηθούν από τις κατάλληλες διεργασίες συγχρονισμού. Εξωτερική διεργασία θα αναλαμβάνει να εκτελέσει τις λειτουργίες αυτές (αναλαμβάνοντας να πραγματοποιήσει αυτόματα τυχόν failover μεταξύ των παρόχων web services εάν απαιτείται) μέχρι να λάβει απάντηση. Οι λειτουργίες θα πρέπει να εκτελούνται σειριακά με τη σειρά αποθήκευσης τους, ώστε να λαμβάνεται πρόνοια για λειτουργίες οι οποίες για να εκτελεστούν βασίζονται στην επιτυχή εκτέλεση προηγούμενης λειτουργίας (πχ προσθήκη χρήστη κάτω από μονάδα μόνο μετά την επιτυχή προσθήκη της μονάδας) και να εκτελούνται αυστηρά μόνο μία φορά (χρησιμοποιώντας τις δυνατότητες που προσφέρει το πρότυπο WS-Reliability/WS-RM). Η ουρά αναμονής θα πρέπει να επιτρέπει τον ορισμό λειτουργιών βασιζόμενων σε προηγούμενες λειτουργίες (πχ με την απόδοση transaction id ανά λειτουργία και τον ορισμό transaction id στο οποίο βασίζεται η λειτουργία) ώστε μόνο επιτυχής ολοκλήρωση μίας λειτουργίας να οδηγεί στην εκτέλεση επόμενης λειτουργίας. Σε περίπτωση που μία λειτουργία είναι ανεπιτυχής η διεργασία αποστολής θα πρέπει να

καταγράφει το μήνυμα λάθους, να μην εκτελεί τυχόν λειτουργίες που βασίζονται σε αυτή και να ενημερώνει το διαχειριστή για την επίλυση του προβλήματος.

6 Ολοκληρωμένη Υπηρεσία

6.1 Αρχιτεκτονική υπηρεσίας

Η εφαρμογή εγκαθίσταται σε εξυπηρετητή ο οποίος διαθέτει ήδη εγκατεστημένο web server Apache με υποστήριξη για PHP. Για παροχή υψηλής διαθεσιμότητας, προτείνεται να εγκατασταθεί σε δύο κατάλληλους εξυπηρετητές, σε αρχιτεκτονική υψηλής διαθεσιμότητας. Μία λειτουργεί ως κύρια εφαρμογή και η δεύτερη ως αντίγραφο ασφαλείας για την παροχή υψηλής διαθεσιμότητας.

Ο κατάλογος στον οποίο είναι εγκατεστημένα τα web services προτείνεται να είναι ο ./usr/local/www/data/ws (συνήθως στον κατάλογο /usr/local/www/data εγκαθίστανται οι δημοσίως προσβάσιμες σελίδες web του apache)..

Για κάθε υπηρεσία που αιτείται υπηρεσία ιστού θα δημιουργούνται δύο αρχεία. Ένα δημοσίως προσβάσιμο στο οποίο θα πραγματοποιείται ο ορισμός όλων των συναρτήσεων-λειτουργιών της υπηρεσίας ιστού και ο σκελετός κλήσης τους και ένα δεύτερο ιδιωτικό στο οποίο θα πραγματοποιείται η υλοποίηση της κάθε λειτουργίας. Κάθε υπηρεσία ιστού πραγματοποιεί πλήρη καταγραφή των καλούμενων λειτουργιών με διατήρηση των ακόλουθων στοιχείων (μία γραμμή ανά κλήση συνάρτησης).

- Ημερομηνία και ώρα κλήσης λειτουργίας
- Όνομα λειτουργίας
- Αιτών
- Εγγραφή στην οποία ζητείται η πραγματοποίηση της λειτουργίας (όνομα χρήστη-username- ή άλλος μοναδικός δείκτης ανά περίπτωση)
- Αποτέλεσμα εκτέλεσης λειτουργίας
- Τυχόν μήνυμα λάθους

Η καταγραφή αυτή γίνεται σε ένα αρχείο ανά υπηρεσία κάτω από τον κατάλογο /var/log/web_services.

Fri, 06 Mar 2009 16:28:05 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 16:28:05 +0200 Reader WS: GetUserRolesDesc: User was not found

Fri, 06 Mar 2009 16:33:46 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 16:34:25 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 16:39:43 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 16:54:29 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 16:59:47 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 17:00:55 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 17:06:11 +0200 Reader WS: GetUserRolesDesc(Client=192.168.240.133): Bind Username= 'cn=manager, dc=vkolyvas,dc=gr',UserName='admin1'

Fri, 06 Mar 2009 17:06:11 +0200 Reader WS: GetUserRolesDesc: User was not found

root@vkolyvas

6.2 Ταχύτητα απόκρισης

Προκειμένου να είναι εύκολος ο έλεγχος της απόδοσης της υπηρεσίας δημιουργήθηκε μία απλή σελίδα μέτρησης απόκρισης (benchmarking) η οποία λαμβάνει ως όρισμα τον αριθμό των εκτελέσεων (*count* argument) και (με χρήση της βιβλιοθήκης nusoap) δημιουργεί ένα web service client και εκτελεί την ίδια λειτουργία ανάγνωσης (ώστε να μην υπάρχει καθυστέρηση εγγραφής στην υπηρεσία καταλόγου) πολλές φορές (<count> φορές). Καθώς ζητείται πάντα η ίδια εγγραφή θεωρείται ότι δεν υπάρχει καθυστέρηση από την υπηρεσία καταλόγου (η εγγραφή επιστρέφεται απευθείας από την μνήμη cache μετά την πρώτη εκτέλεση). Με την ακόλουθη διαμόρφωση:

- Ενεργοποιημένος ο APC accelerator
- Σύνδεση στα web services στο localhost
- Σύνδεση στην υπηρεσία καταλόγου στο localhost (από το LUMS).

και χωρίς να λαμβάνεται υπόψη η πρώτη εκτέλεση (ώστε να γίνουν cache οι σελίδες στον accelerator και η εγγραφή στην υπηρεσία καταλόγου) προκύπτουν τα ακόλουθα νούμερα:

- Εκτέλεση μία φορά: 0,05 sec
- Εκτέλεση 10 φορές: 0,5 sec
- Εκτέλεση 100 φορές: 5,13 sec

Βλέπουμε λοιπόν ότι η εκτέλεση της λειτουργίας (ακόμα και με την καθυστέρηση κλήσης της ίδιας της σελίδας του benchmarking, ενεργοποίησης του client, μεταφόρτωσης και μεταγλώττισης του web service wsdl από τον client και παρουσίασης των αποτελεσμάτων) απαιτεί μόνο 0,05 δευτερόλεπτα ενώ ο χρόνος πολλαπλής εκτέλεσης είναι απολύτως γραμμικός (100 εκτελέσεις σε 5,13 δευτερόλεπτα). Καθώς οι εκτελέσεις ήταν σειριακές ο χρόνος αυτός είναι ο χρόνος εξυπηρέτησης για **ένα** thread του web server. Κατά συνέπεια προκύπτει ρυθμός εξυπηρέτησης περίπου 19 calls/thread. Με δεδομένο ότι σε ένα web server μπορούν να οριστούν πολλαπλά threads (τουλάχιστον 10 – 20) προκύπτει ότι είναι δυνατό να εξυπηρετηθούν τουλάχιστον 200 – 500 κλήσεις/sec.

Προκειμένου να ελέγξουμε και την καθυστέρηση του δικτύου στην κλήση των web services έγινε η ίδια διαδικασία με τις ακόλουθες αλλαγές:

- Σύνδεση στα web services από απομακρυσμένο πελάτη (όχι localhost).
- Σύνδεση στην υπηρεσία καταλόγου στο απομακρυσμένα (από το LUMS).

Στην περίπτωση αυτή προκύπτουν τα ακόλουθα νούμερα:

- Εκτέλεση μία φορά: 0,2 sec
- Εκτέλεση 10 φορές: 1,86 sec
- Εκτέλεση 100 φορές: 19,19 sec

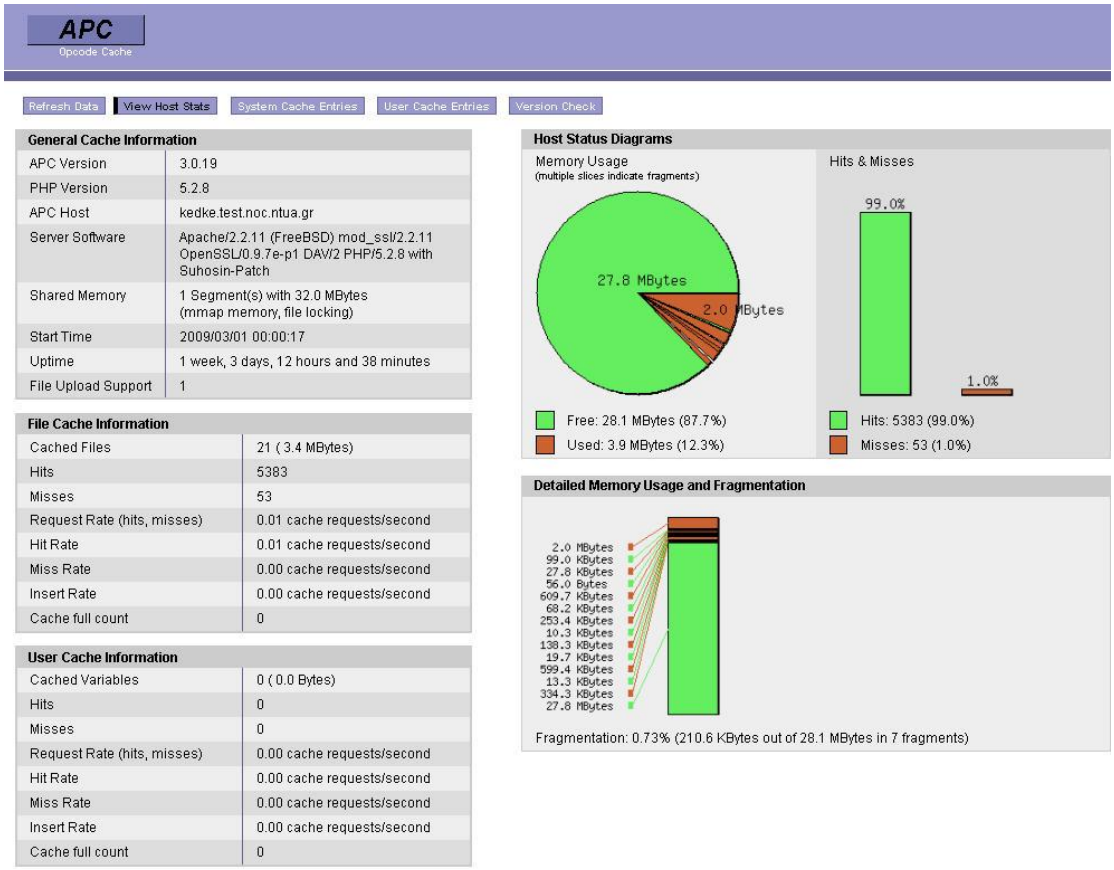
Βλέπουμε ότι και πάλι η εκτέλεση της λειτουργίας παρουσιάζει τα ίδια χαρακτηριστικά με μειωμένο ρυθμό λόγω καθυστέρησης δικτύου. Στην περίπτωση αυτή προκύπτει ρυθμός εξυπηρέτησης περίπου 5 calls/thread και περίπου 50 – 100 κλήσεις/sec.

6.2.1 Στατιστικά υπηρεσίας

Στατιστικά στοιχεία για την υπηρεσία ιστού είναι δυνατόν να εξαχθούν από τα log files της υπηρεσίας.

Στατιστικά στοιχεία του php accelerator είναι διαθέσιμα μέσω ειδικής σελίδας που παρέχεται από το πακέτο λογισμικού APC.

Παράδειγμα εκτέλεσης της σελίδας:



6.3 Βιβλιοθήκη λειτουργιών LUMS

Η υλοποίηση του κώδικα πραγματοποίησης αλλαγών στην υπηρεσία καταλόγου (προσθήκη εγγραφής, αλλαγή, διαγραφή) παρουσιάζει ιδιαίτερες απαιτήσεις όπως:

- Δυνατότητα για παραγόμενες τιμές σε πεδία (attributes) με βάση τις τιμές άλλων (εισαγόμενων από το χρήστη) και με δυνατότητα εκτέλεσης οποιασδήποτε δυνατής έκφρασης ή συνάρτησης PHP για την παραγωγή των τιμών.
- Ορισμό περιορισμών στον τύπο των τιμών των πεδίων (τύπος email, τηλέφωνο κτλ) όσο και επιπλέον περιορισμών (πχ επιλογή δυνατών τιμών από συγκεκριμένη λίστα τιμών)
- Ειδικούς τύπους πεδίων όπως αυτόματης παραγωγής αυξανόμενων τιμών (auto-increment)

- Βοηθητικές λειτουργίες για τη διατήρηση της ακεραιότητας των δεδομένων στο δέντρο πληροφοριών όπως:

    A. Αναφορική Ακεραιότητα (Referential Integrity): Σε περίπτωση διαγραφής εγγραφής, αυτή διαγράφεται από το σύνολο των αντικειμένων στα οποία έχει προστεθεί ως αναφορά (συνήθως, ομάδες χρηστών)

    B. Μοναδικότητα Πεδίου (Attribute Uniqueness): Σε περιπτώσεις απαιτείται η τιμή ενός πεδίου να είναι μοναδική σε ολόκληρο το δέντρο πληροφοριών. Παραδείγματα είναι το όνομα χρήστη (username), διεύθυνση ηλεκτρονικού ταχυδρομείου (email) και κωδικός εργαζόμενου (employeenumber) ενός χρήστη.

- Δυνατότητα πραγματοποίησης εξωτερικών διεργασιών πριν και μετά την πραγματοποίηση των αλλαγών σε εγγραφή στην υπηρεσία καταλόγου.

Για την καλύτερη ικανοποίηση των παραπάνω απαιτήσεων, την μείωση του απαιτούμενου κώδικα και τη δυνατότητα επαναχρησιμοποίησης του σε οποιαδήποτε νέα εφαρμογή υπηρεσίας ιστού, δημιουργήθηκε κατάλληλη διεπαφή προγραμματισμού εφαρμογών σε PHP με όνομα LDAP User Management Service (LUMS). Η διεπαφή αυτή δίνεται με την μορφή βιβλιοθήκης η οποία μπορεί να συμπεριληφθεί απευθείας από οποιαδήποτε σελίδα PHP απαιτείται. Η βιβλιοθήκη παρέχει βασικές συναρτήσεις αλληλεπίδρασης με την υπηρεσία καταλόγου και ένα ιδιαίτερα εξελιγμένο αρχείο διαμόρφωσης της βιβλιοθήκης που παρέχει πληθώρα δυνατοτήτων..

6.3.1 Συναρτήσεις

Οι συναρτήσεις που είναι διαθέσιμες είναι οι ακόλουθες:

- function LUMS_ldap_search($L_binddn, $L_bindpassword, $L_basedn, $L_scope, $L_filter, $L_attrs_array)
    - Περιγραφή: Πραγματοποιεί αναζήτηση LDAP
    - Είσοδος:

        - L_binddn: Το διακεκριμένο όνομα χρήστη κατά τη διαδικασία δέσμευσης (Bind Distinguish Name)
        - L_bindpassword: Το συνθηματικό που θα χρησιμοποιηθεί κατά τη διαδικασία δέσμευσης (bind)
        - L_basedn: Η βάση (base) της αναζήτησης
        - L_scope: Το πεδίο (scope) της αναζήτησης
        - L_filter: Το φίλτρο της αναζήτησης
        - L_attrs_array: Προαιρετική δομή τύπου array με τα πεδία που θα επιστραφούν. Αν δε δοθεί επιστρέφονται όλα τα πεδία
    - Έξοδος: Οι εγγραφές που βρέθηκαν (όπως επιστρέφονται από τη συνάρτηση ldap_search() της PHP σε μορφή πίνακα), είτε κατάλληλο αλφαριθμητικό με το μήνυμα λάθους σε περίπτωση αποτυχίας
- function LUMS_ldap_add_entry($L_binddn, $L_bindpassword, $L_object_type, $L_entrydn, $L_entry_info).
    - Περιγραφή: Προσθήκη εγγραφής
    - Είσοδος:
        - L_binddn: Το διακεκριμένο όνομα χρήστη κατά τη διαδικασία δέσμευσης (Bind Distinguish Name)
        - L_bindpassword: Το συνθηματικό που θα χρησιμοποιηθεί κατά τη διαδικασία δέσμευσης (bind)
        - L_object_type: Ο τύπος του αντικειμένου
        - L_entrydn: Το διακεκριμένο όνομα της νέας εγγραφής
        - L_entry_info: Τα στοιχεία της νέας εγγραφής υπό την μορφή πίνακα
    - Έξοδος: 0 για επιτυχία ή κατάλληλο αλφαριθμητικό με το μήνυμα λάθους σε περίπτωση αποτυχίας
- function LUMS_ldap_change_password($L_binddn, $L_bindpassword, $L_entrydn, $L_newpassword)
    - Περιγραφή: Αλλαγή του συνθηματικού εγγραφής

- Είσοδος:
    - L_binddn: Το διακεκριμένο όνομα χρήστη κατά τη διαδικασία δέσμευσης (Bind Distinguish Name)
    - L_bindpassword: Το συνθηματικό που θα χρησιμοποιηθεί κατά τη διαδικασία δέσμευσης (bind)
    - L_entrydn: Το διακεκριμένο όνομα της εγγραφής
    - L_newpassword: Το νέο συνθηματικό της εγγραφής
  - Έξοδος: 0 για επιτυχία ή κατάλληλο αλφαριθμητικό με το μήνυμα λάθους σε περίπτωση αποτυχίας

- function LUMS_ldap_modify_entry($L_binddn, $L_bindpassword, $L_object_type, $L_entrydn, $L_change_info)
  - Περιγραφή: Αλλαγή στοιχείων εγγραφής (με τον ίδιο τρόπο όπως η συνάρτηση ldap_modify() της PHP)
  - Είσοδος
    - L_binddn: Το διακεκριμένο όνομα χρήστη κατά τη διαδικασία δέσμευσης (Bind Distinguish Name)
    - L_bindpassword: Το συνθηματικό που θα χρησιμοποιηθεί κατά τη διαδικασία δέσμευσης (bind)
    - L_object_type: Ο τύπος του αντικειμένου
    - L_entrydn: Το διακεκριμένο όνομα της εγγραφής
    - L_change_info: Τα στοιχεία της εγγραφής που απαιτούν αλλαγή
  - Έξοδος: 0 για επιτυχία ή κατάλληλο αλφαριθμητικό με το μήνυμα λάθους σε περίπτωση αποτυχίας

- function LUMS_ldap_delete_entry($L_binddn, $L_bindpassword, $L_object_type, $L_entrydn)
  - Περιγραφή: Διαγραφή εγγραφής
  - Είσοδος

        - L_binddn: Το διακεκριμένο όνομα χρήστη κατά τη διαδικασία δέσμευσης (Bind Distinguish Name)
        - L_bindpassword: Το συνθηματικό που θα χρησιμοποιηθεί κατά τη διαδικασία δέσμευσης (bind)
        - L_object_type: Ο τύπος του αντικειμένου
        - L_entrydn: Το διακεκριμένο όνομα της εγγραφής
    - Έξοδος: 0 για επιτυχία ή κατάλληλο αλφαριθμητικό με το μήνυμα λάθους σε περίπτωση αποτυχίας

- function LUMS_ldap_rename_entry($L_binddn, $L_bindpassword, $L_object_type, $L_entrydn, $L_newrdn, $L_newparent, $L_deleteoldrdn)
    - Περιγραφή: Μετονομασία της εγγραφής (πραγματοποίηση ModRDN)
    - Είσοδος
        - L_binddn: Το διακεκριμένο όνομα χρήστη κατά τη διαδικασία δέσμευσης (Bind Distinguish Name)
        - L_bindpassword: Το συνθηματικό που θα χρησιμοποιηθεί κατά τη διαδικασία δέσμευσης (bind)
        - L_object_type: Ο τύπος του αντικειμένου
        - L_entrydn: Το διακεκριμένο όνομα της εγγραφής
        - L_newrdn: Το νέο σχετικό διακεκριμένο όνομα (Relative Distinguished Name) της εγγραφής
        - L_newparent: Ο νέος πατέρας της εγγραφής
        - L_deleteoldrdn: Δυαδική μεταβλητή που ορίζει κατά πόσον θα διαγραφεί το παλαιό σχετικό διακεκριμένο όνομα.
    - Έξοδος: 0 για επιτυχία ή κατάλληλο αλφαριθμητικό με το μήνυμα λάθους σε περίπτωση αποτυχίας

- function LUMS_ldap_plain_modify_entry($L_binddn, $L_bindpassword, $L_entrydn, $L_change_info)
    - Περιγραφή: Ίδια συνάρτηση με την LUMS_ldap_modify_entry() μόνο που πραγματοποιούνται απευθείας οι αλλαγές που παρέχονται στο

$L_change_info χωρίς την επιπλέον εξυπνάδα του framework (ουσιαστικά ldap_modify() με connection management).

- function LUMS_ldap_plain_add_entry($L_binddn, $L_bindpassword, $L_entrydn, $L_entry_info)
    - Περιγραφή: Ίδια συνάρτηση με την LUMS_ldap_add_entry() μόνο που πραγματοποιούνται απευθείας η προσθήκη της εγγραφής που παρέχεται στο $L_entry_info χωρίς την επιπλέον εξυπνάδα του framework (ουσιαστικά ldap_add() με connection management).

Σε όλες τις περιπτώσεις εφόσον το L_binddn παραμείνει κενό θα χρησιμοποιηθούν τα στοιχεία που έχουν οριστεί στο αρχείο διαμόρφωσης της βιβλιοθήκης.

6.3.2 Μορφή Αρχείων

Η βιβλιοθήκη προτείνεται να εγκατασταθεί στον κατάλογο /usr/local/lums. Κάτω από τον κατάλογο αυτό περιλαμβάνονται δύο κατάλογοι:

- Ο κατάλογος config ο οποίος περιέχει το αρχείο config.php το οποίο και είναι το αρχείο διαμόρφωσης της βιβλιοθήκης
- Ο κατάλογος lib ο οποίος περιέχει τον κώδικα της βιβλιοθήκης:
    - Το αρχείο main.php είναι το κύριο αρχείο το οποίο πρέπει να γίνεται include για τη χρήση της βιβλιοθήκης
    - Το αρχείο functions.php το οποίο είναι βοηθητικό αρχείο της βιβλιοθήκης
    - Το αρχείο extras.php στο οποίο μπορούν να προστεθούν οι βοηθητικές συναρτήσεις που ορίζονται από το χρήστη της βιβλιοθήκης. Περισσότερες πληροφορίες για τις συναρτήσεις αυτές περιέχονται παρακάτω στο παρών κείμενο.

6.3.3 Αρχείο Διαμόρφωσης

Το αρχείο διαμόρφωσης επιτρέπει τον ορισμό όλων λειτουργιών που περιγράφηκαν παραπάνω όσο και άλλων επιπλέον. Το αρχείο είναι σε μορφή μεταβλητών PHP ώστε να είναι εύκολη και άμεση η ανάγνωση του και χωρίζεται σε τρία μέρη.

Στο πρώτο ορίζονται ορισμένα βασικά στοιχεία για τη βιβλιοθήκη. Το charset που θα χρησιμοποιηθεί για τιμές που δεν είναι σε αγγλικό αλφάβητο, ο τύπος της κρυπτογράφησης που χρησιμοποιείται για τα passwords και το attribute το οποίο ορίζει τον τύπο των αντικειμένων (είσοδος $L_object_type στις συναρτήσεις):

```
$LUMS_Config[Main][objecttype_attribute] = 'edupersonorgunitdn';

$LUMS_Config[Main][encryption_scheme] = 'crypt';

$LUMS_Config[Main][interfaceid] = '1';

$LUMS_Config[Main][config_format_version] = '1.1';

$LUMS_Config[Main][non_english_charset] = 'iso-8859-7';

$LUMS_Config[Main][debug] = 0;
```

Στο δεύτερο μέρος ορίζονται τα στοιχεία για τον εξυπηρετητή καταλόγου που θα χρησιμοποιηθεί όπως το hostname, το Bind DN και Password, το base και το DN κάτω από το οποίο θα διατηρούνται τυχόν αντικείμενα counters για την υποστήριξη autoincrement μεταβλητών:

```
$LUMS_Config[LDAP][server] = 'ldap://localhost';

$LUMS_Config[LDAP][binddn] = '';

$LUMS_Config[LDAP][bindpassword] = '';

$LUMS_Config[LDAP][base] = 'ou=people,o=papei,c=gr';

$LUMS_Config[LDAP][countersdn] =
'ou=sequences,ou=config,dc=company,dc=com';
```

Στο τρίτο μέρος γίνεται ο ορισμός των αντικειμένων των εγγραφών. Οι εγγραφές χωρίζονται ανά τύπο αντικειμένου και στη συνέχεια γίνεται ο ορισμός των ακόλουθων στοιχείων:

- Λειτουργίες (Operations) οι οποίες μπορούν να εκτελεστούν πριν και μετά την πραγματοποίηση των αλλαγών στις εγγραφές. Οι λειτουργίες αυτές ορίζονται με την μορφή συναρτήσεων και αναφέρονται στις περιπτώσεις προσθήκης (add), ενημέρωσης (modify), μετονομασίας (rename) και διαγραφής (delete) εγγραφών.
- Mappings που χρησιμοποιούνται στην περίπτωση των mapped attributes (περιγράφονται παρακάτω στο κείμενο).
- Attributes εγγραφής και στοιχείων που αφορούν αυτά. Τα στοιχεία αυτά αναλύονται στη συνέχεια.

Παράδειγμα ορισμού λειτουργιών:

```
#$LUMS_Config[Object][parent][operations][preadd] =
'LUMS_operations_parent_preadd';

#$LUMS_Config[Object][parent][operations][postadd] =
'LUMS_operations_parent_postadd';

#$LUMS_Config[Object][parent][operations][premodify] =
'LUMS_operations_parent_premodify';

#$LUMS_Config[Object][parent][operations][postmodify] =
'LUMS_operations_parent_postmodify';

#$LUMS_Config[Object][parent][operations][prerename] =
'LUMS_operations_parent_prerename';

#$LUMS_Config[Object][parent][operations][postrename] =
'LUMS_operations_parent_postrename';

#$LUMS_Config[Object][parent][operations][predelete] =
'LUMS_operations_parent_predelete';

#$LUMS_Config[Object][parent][operations][postdelete] =
'LUMS_operations_parent_postdelete';
```

Τα mappings ορίζονται ως εξής:

[mappings][<index attribute name>][<mapped attribute name>][<index attribute value>] = '<mapped attribute value>'

Παράδειγμα ορισμού mappings:

```
$LUMS_Config[Object][parent][mappings][indexattr][businesscategory][a] = 'Undergraduate Student';
$LUMS_Config[Object][parent][mappings][indexattr][businesscategory][b] = 'Administrative Personnel';
$LUMS_Config[Object][parent][mappings][indexattr][edupersonaffiliation][a] = 'student';
$LUMS_Config[Object][parent][mappings][indexattr][edupersonaffiliation][b] = 'employee';
```

Για τον ορισμό των attributes ακολουθείται η ακόλουθη λογική:

$LUMS_Config[Object][<object type name>][<attribute name>][<directive>]

Δυνατές επιλογές για την τιμή directive:

- Required = 1 or 0: Ορίζει αν το attribute είναι απαιτούμενο (και πρέπει να υπάρχει στην είσοδο κατά τη δημιουργία της εγγραφής)
- Multivalue = 1 or 0: Ορίζει αν το attribute είναι πλειότιμο ή όχι
- Type = 'string|dn|binary|mail|telephonenumber|password': Ορίζει τον τύπο του attribute
- Checktypefunction = <function name>: Ορίζει επιπλέον συνάρτηση ελέγχου της τιμής του attribute. Η συνάρτηση πρέπει να έχει οριστεί στο αρχείο lib/extras.php και επιστρέφει 0 σε περίπτωση αποτυχίας ή θετική τιμή εάν η τιμή είναι επιτρεπόμενη.
- Valuetype = 'constant|uservalue|callfunc|autoincrement|virtual|mapping|copyattr': Ορίζει τον τύπο της τιμής για το attribute:
    - Constant: Η τιμή είναι σταθερή και ορίζεται στο ίδιο το αρχείο διαμόρφωσης.

- Uservalue: Η τιμή εισάγεται ως είσοδος κατά την κλήση των συναρτήσεων
- Callfunc: Η τιμή λαμβάνεται από την έξοδο κατάλληλης συνάρτησης
- Autoincrement: Η τιμή προκύπτει από αυτόματη αυξανόμενη αρίθμηση την οποία πραγματοποιεί η ίδια η βιβλιοθήκη.
- Virtual: Η τιμή είναι εικονική. Χρησιμοποιείται μόνο ως δείκτης για τον προσδιορισμό τιμών στα mappings.
- Mapping: Η τιμή δημιουργείται μέσω κατάλληλου πίνακα. Η τιμή ενός virtual attribute χρησιμοποιείται ως δείκτης για την επιλογή της τιμής από τον πίνακα
- Copyattr: Η τιμή προκύπτει απλά με την αντιγραφή της τιμής άλλου attribute.

Η σειρά του evaluation για τις παραπάνω κατηγορίες είναι:

1. constant
2. uservalue
3. callfunc (η συνάρτηση μπορεί να πραγματοποιήσει τα πάντα οπότε θεωρούμε ότι μπορεί να περιλάβει και αντιγραφή της λειτουργίας του mapping και copyattr)
4. mapping
5. copyattr (ώστε να είναι διαθέσιμα όλα τα attributes τα οποία μπορεί να αντιγραφούν).

6.3.4 Constant

Στην περίπτωση των σταθερών τιμών μπορούν να δωθούν με την μορφή:

[constant][values][<index>] = <value>

Όπου index είναι νούμερο στοιχείου πίνακα (αρίθμηση ξεκινάει από το 0). Κατ' αυτόν τον τρόπο είναι δυνατόν να δωθούν πολλαπλές τιμές για την περίπτωση πλειότιμων attributes.

6.3.5 Callfunc

Στην περίπτωση αυτή ορίζεται η συνάρτηση προς κλήση με την μορφή:

[callfunction] = <function name>

6.3.6 Autoincrement

Στην περίπτωση των counters αυτοί υλοποιούνται από την ίδια τη βιβλιοθήκη με τη βοήθεια κατάλληλων εγγραφών στην υπηρεσία καταλόγου. Για το λόγο αυτό ορίστηκε αρχικά το countersdn στο αρχείο διαμόρφωσης. Για κάθε counter δημιουργείται διαφορετική εγγραφή κάτω από το countersdn ενώ στη διαμόρφωση πρέπει να οριστούν τα εξής:

6.3.7 Unique

Για κάθε attribute είναι δυνατόν να οριστεί ότι πρέπει να είναι μοναδικό για ένα συγκεκριμένο υποδέντρο. Ο ορισμός γίνεται ως εξής:

[constraint_unique] = '1'

[unique][base] = "<base>" όπου <base> είναι το υποδέντρο κάτω από το οποίο θα πρέπει να λαμβάνει μοναδικές τιμές το attribute.

6.3.8 Mapping

Για τις περιπτώσεις των mapped attributes απαιτείται να οριστεί το attribute το οποίο θα χρησιμοποιηθεί ως δείκτης σε πίνακα για την εξαγωγή τιμών. Οι τιμές αυτές είναι οι τελικές που θα λάβει το mapped attribute:

[mapping][indexattribute] = '<index attribute name>'

6.4 Παράδειγμα Αρχείου Διαμόρφωσης

Στο Παράρτημα ΙΙ περιέχεται αναλυτικό παράδειγμα πλήρους αρχείου διαμόρφωσης για την υπηρεσία.

6.4.1 Περιορισμοί Τιμών

Είναι δυνατό να οριστούν περιορισμοί στις τιμές που μπορεί να λάβει ένα attribute. Σε αυτή τη φάση οι περιορισμοί είναι ένας πίνακας από δυνατές τιμές. Η διαμόρφωση γίνεται ως εξής:

[constraint][type] = 'arrayOfValues'

[constraint][values][#] = '<val>'

Όπου # είναι array index αριθμός (ξεκινά από το 0) και <val> είναι η δυνατή τιμή του attribute.

6.4.2 Συναρτήσεις που χρησιμοποιούνται

Οι παρακάτω βοηθητικές συναρτήσεις μπορούν να οριστούν από το χρήστη και προστίθονται στο αρχείο lib/extras.php όπως περιγράφηκε στη δομή της βιβλιοθήκης. Οι τύποι των διαθέσιμων συναρτήσεων είναι οι ακόλουθοι:

- **Οι συναρτήσεις ελέγχου τύπου:**

function LUMS_checktypefun_<attribute name>($attribute_values)

Το $attribute_values είναι μορφής πίνακα αν το attribute είναι πλειότιμο. Η συνάρτηση επιστρέφει 1 για επιτυχία και 0 για αποτυχία ελέγχου

- **Οι συναρτήσεις callfunc:**

function LUMS_callfun_<name>_<attribute>($info)

Το $info είναι $entry_info για τις συναρτήσεις προσθήκης εγγραφών και $change_info για τις συναρτήσεις αλλαγής εγγραφών. Η συνάρτηση επιστρέφει πίνακα. Το πρώτο στοιχείο είναι ένα από τα 'notset', 'empty', 'ok', 'error'. notset επιστρέφεται αν τα attributes που είναι απαραίτητα δεν είναι διαθέσιμα, empty αν είναι διαθέσιμα αλλά οι τιμές τους είναι κενές, error αν υπήρξε σφάλμα (στην περίπτωση αυτή το δεύτερο στοιχείο είναι περιγραφή του σφάλματος) και ok αν όλα ήταν επιτυχή. Το δεύτερο στοιχείο είναι η επιστρεφόμενη τιμή.

- **Οι συναρτήσεις auto increment:**

function LUMS_incrementfun_<name>($LDAP_Conn, $CountersDN, $CounterName, $InterfaceID)

Η ήδη διαθέσιμη συνάρτηση LUMS_incrementfun_incrementbyone είναι ήδη διαθέσιμη και επιστρέφει την αμέσως επόμενη τιμή. Το $InterfaceID είναι διαθέσιμο στη διαμόρφωση του LUMS και πρέπει να είναι μοναδικό ανά interface. Χρησιμοποιείται για να παράγει μοναδικές τιμές ακόμα και αν υπάρχουν πολλαπλές εγκαταστάσεις του LUMS. Η εγγραφή του counter έχει την μορφή:

dn: cn=$CounterName, $CountersDN

objectclass: top

objectclass: person

cn: $CounterName

sn: <value increment by one>.$InterfaceID

7 Έλεγχοι λειτουργίας συστήματος

7.1 Κεντρική υπηρεσία καταλόγου

Για την διασφάλιση της ομαλής λειτουργίας και αποδοχής της κεντρικής υπηρεσίας LDAP προτείνονται οι παρακάτω διαδικασίες.

- Έλεγχος συνδεσιμότητας. Αποστολή ενός ICMP echo request πακέτου με σκοπό τον έλεγχο της δικτυακής ύπαρξης του υλικού εξοπλισμού. Εάν αυτός ο έλεγχος αποτύχει σημαίνει πως ο εξοπλισμός και κατά συνέπεια και το λογισμικό βρίσκονται εκτός δικτύου.
- Έλεγχος διεργασίας. Μέσω των εργαλείων του λειτουργικού συστήματος που φιλοξενεί το λογισμικό LDAP θα πρέπει να ελεγχθεί η ύπαρξη της διεργασίας του εξυπηρετητή LDAP.
- Επιτυχής σύνδεση. Το λογισμικό εξυπηρετητή LDAP επιτρέπει την σύνδεση από εξωτερικούς φορείς στις TCP πόρτες 389,636 . Μέσω αυτού του μηχανισμού διεξάγεται κάθε επικοινωνία του λογισμικού με τις εφαρμογές που πρέπει να έχουν πρόσβαση στην υπηρεσία. Σύνδεση με αυτόν τον τρόπο στην υπηρεσία επιβεβαιώνει ότι η υπηρεσία είναι προσβάσιμη τουλάχιστον στο επίπεδο μεταφοράς δεδομένων.
- Εκτέλεση λειτουργιών. Η συμπεριφορά του λογισμικού σε επίπεδο εφαρμογής, δηλαδή η ανταπόκριση βασικές λειτουργίες που πρέπει να επιτυγχάνει, ήτοι αναζήτηση, προσθήκη, μεταβολή, διαγραφή εγγραφών, θα πρέπει να ελεγχθεί. Για αυτόν τον έλεγχο αρκεί ένας μηχανισμός που θα αναζητά, προσθέτει, μεταβάλλει και διαγράφει συγκεκριμένες εγγραφές.
- Εκτέλεση αλλαγής συνθηματικού. Ειδική υποπερίπτωση των παραπάνω ελέγχων είναι η αλλαγή συνθηματικού σε τουλάχιστον μία εγγραφή. Αυτός ο έλεγχος είναι απαραίτητος και χρήζει ειδικής μνείας αν και είναι δυνατόν να ενσωματωθεί στον παραπάνω μηχανισμό.
- Έλεγχος των αρχείων καταγραφής. Η παρακολούθηση των αρχείων καταγραφής των εξυπηρετητών μέσω της γραμμής εντολών επιτρέπει να πιστοποιηθεί με τον πλέον σίγουρο τρόπο η εύρυθμη λειτουργία της υπηρεσίας..

- Επιτυχής επανεκκίνηση της υπηρεσίας. Η εφαρμογή θα πρέπει να επανεκκινηθεί τουλάχιστον μία φορά μέσω της γραμμής εντολών και των εργαλείων που προσφέρει για αυτή την περίπτωση το πακέτο λογισμικού.
- Επιτυχής απόκτηση αντιγράφου ασφαλείας. Θα πρέπει να πραγματοποιηθεί η επιτυχημένη απόκτηση αντιγράφου ασφαλείας και η ορθότητα των περιεχομένων του. Αυτή η διαδικασία δεν πρέπει να συγχέεται με υπάρχουσα υποδομή αντιγράφων ασφαλείας και θα πρέπει να εκτελεστεί χειροκίνητα από τους διαχειριστές μέσω της γραμμής εντολών.
- Επιτυχής ανάκτηση δεδομένων από αντίγραφο ασφαλείας. Θα πρέπει να πραγματοποιηθεί επιτυχής ανάκτηση των δεδομένων από προηγούμενο αντίγραφο ασφαλείας. Η υπηρεσία μετά την ανάκτηση δεδομένων θα πρέπει να ικανοποιήσει τους ελέγχους που προδιαγράφηκαν προηγουμένως και να περιέχει πλήρη και ορθά δεδομένα, όμοια με αυτά που λήφθηκαν κατά τη διαδικασία απόκτησης αντίγραφου ασφαλείας.

7.2 Υπηρεσίες ιστού

Στην περίπτωση των web services που θα διατίθενται στις εξωτερικές υπηρεσίες απαιτείται ο έλεγχος των εξής στοιχείων:

- **Έλεγχος πρόσβασης μέσω Web (HTTP).** Ο έλεγχος αυτός μπορεί να πραγματοποιηθεί πολύ απλά με τη δοκιμαστική πρόσβαση στη σελίδα web που αντιστοιχεί στο κάθε web service

- **Έλεγχος αρχείου WSDL.** Για την ανάπτυξη και ολοκλήρωση της διαλειτουργικότητας μέσω web services είναι απαραίτητο να είναι διαθέσιμος ο ορισμός των λειτουργιών σε μορφή WSDL. Επιπλέον ο ορισμός αυτός θα πρέπει να είναι συντακτικά σωστός και πλήρης. Η πρόσβαση στο WSDL αρχείο κάθε web service μπορεί να γίνει απευθείας μέσω web με την κλήση της σελίδας web με την προσθήκη του argument *?wsdl* στο URL. Εν συνεχεία

το αρχείο μπορεί να ελεγχθεί αν είναι συντακτικά και λειτουργικά σωστό και πλήρες.

- **Έλεγχος λειτουργικότητας web service.** Το πλέον βασικό στάδιο ελέγχου είναι η επιβεβαίωση της καλής λειτουργίας του ίδιου του web service. Για το σκοπό αυτό θα δημιουργηθεί δοκιμαστική σελίδα για κάθε web service που θα αναλάβει να εκτελέσει κάθε προβλεπόμενη λειτουργία με κατάλληλο δοκιμαστικό input. Η σελίδα αυτή θα πρέπει να πραγματοποιεί HTTP Authentication με τα προβλεπόμενα στοιχεία χρήστη που θα χρησιμοποιηθούν στην εν λειτουργία χρήση των web services. Ο έλεγχος θα είναι επιτυχής εφόσον:
    - Η σελίδα επιστρέφει επιτυχώς για κάθε προβλεπόμενη λειτουργία.
    - Το web service έχει καταγράψει επιτυχώς και πλήρως όλες τις εκτελεσμένες λειτουργίες στο προβλεπόμενο log file.
    - Ο έλεγχος των δεδομένων της υπηρεσίας καταλόγου επιβεβαιώνει την επιτυχή εκτέλεση των αντίστοιχων λειτουργιών.
    - Η λειτουργία ολοκληρώνεται σε εύλογο χρονικό διάστημα μερικών δευτερολέπτων.

  Καθώς οι λειτουργίες που προβλέπεται να υποστηρίξουν τα web services δεν προβλέπεται να είναι μαζικές αλλά συγκεκριμένες και χρονικά περιορισμένες (μικρός αριθμός αλλαγών εγγραφών χρηστών/μέρα) δεν κρίνεται σκόπιμη η δοκιμή υψηλού φόρτου των web services. Εφόσον η ολοκλήρωση κάθε λειτουργίας γίνεται σε επαρκώς μικρό χρονικό διάστημα (επιθυμητά λίγα δευτερόλεπτα) δε θεωρείται ότι σε συνθήκες πραγματικής λειτουργίας τα web services θα παρουσιάσουν πρόβλημα υποστήριξης του προβλεπόμενου φόρτου εργασίας.

Στο ΠΑΡΑΡΤΗΜΑ ΙΙΙ παρουσιάζεται το αρχείο που δημιουργήθηκε για την επικοινωνία και τον έλεγχο των δεδομένων της υπηρεσίας καταλόγου.

8 Εγκατάσταση

Σε ένα εγκατεστημένο instance του fedora core 11 – i386 δίνουμε προσοχή πως αυτά που χρειαζόμαστε δεν υπάρχουν και θα πρέπει να την εμπλουτίσουμε με εγκατάσταση των απαραίτητων αρχείων για να τρέξει η εφαρμογή μας.

8.1 Apache

Ο Apache HTTP Server είναι ένα δωρεάν πρόγραμμα/ανοικτού κώδικα web server για το Fedora και για τα Unix-like συστήματα.

Η εγκατάσταση γίνεται με την παρακάτω εντολή:

```
# yum install httpd
```

Εκκίνηση

```
# chkconfig httpd on

# /etc/init.d/httpd start
```

8.2 PHP (Hypertext Preprocessor)

Η PHP είναι μια server-side γλώσσα προγραμματισμού Ιστού που μπορεί να ενσωματωθεί σε σελίδες HTML. Όταν ένας χρήστης αποκτήσει πρόσβαση σε μια σελίδα βασισμένη σε PHP,η PHP δημιουργεί δυναμικά μια ιστοσελίδα που μετά περνάει στο πρόγραμμα περιήγησης.

Η PHP δουλεύει με Apache, Lighttpd και άλλους webservers. Η PHP προσφέρει ενσωματωμένη βάση δεδομένων για την ολοκλήρωση πολλών εμπορικών και μη συστημάτων διαχείρισης βάσεων δεδομένων. Έχει επίσης τη δυνατότητα να εκτελεί πολλά χρήσιμα Web-tasks χρησιμοποιώντας ένα μεγάλο σύνολο ενσωματωμένων λειτουργιών.

Η PHP τρέχει σε γενικές γραμμές σε ένα web server, λαμβάνοντας PHP κώδικα ως είσοδο της και δημιουργεί ιστοσελίδες, αλλά και εντολές δέσμης ενεργειών γραμμής (command-line scripting) και client-side GUI εφαρμογές αποτελούν μέρος από τις τρεις κύριες χρήσης της PHP. Η PHP μπορεί να αναπτυχθεί σε οποιοδήποτε web server και σε σχεδόν κάθε πλατφόρμα OS δωρεάν. Η εγκατάσταση της PHP είναι εύκολη και γίνεται με τις παρακάτω εντολές:

```
# yum install php
```

Η αναβάθμιση της όταν απαιτείται μπορεί να γίνει μέσω της εντολής:

```
# yum update php
```

8.3 PECL και APC

Η PECL είναι μια αποθήκη για PHP Extensions, παρέχοντας έναν κατάλογο όλων των γνωστών επεκτάσεων για τη λήψη και την ανάπτυξη PHP επεκτάσεων.

Η Alternative PHP Cache (APC) είναι μια ελεύθερη και ανοιχτή opcode cache για την PHP. Στόχος της είναι να παρέχει ένα ελεύθερο, ανοικτό, και ισχυρό πλαίσιο για την προσωρινή αποθήκευση και τη βελτιστοποίηση του ενδιάμεσου κώδικα που παράγεται από την PHP.

```
# yum install php-pecl-apc
```

8.4 OpenLDAP και PHP-LDAP

```
# yum install openldap-servers

# yum install php-ldap
```

Μπορούσαμε να είχαμε εγκαταστήσει και τον «openldap-client», αλλά το πακέτο του πελάτη δεν χρειάζεται στον server. Παρεμπιπτόντως το πακέτο «nss_ldap», εγκαθίσταται από μόνο του, το οποίο περιέχει το «libnss_ldap» και το «pam_ldap», τα οποία χρειάζονται για έναν πελάτη. Το «pam_ldap» θα βοηθήσει στην

ολοκλήρωση του LDAP με το email, SSH, FTP, Samba, όπου στην συγκεκριμένη περίπτωση δεν είναι απαραίτητο για την ανάπτυξη της εφαρμογής, αλλά για μελλοντική επέκτασή της.

8.5 **Παραμετροποίηση του LDAP**

Η αρχή κάθε παραμετροποίησης μιας βάσης δεδομένων είναι η δημιουργία ενός κωδικού ασφαλείας για τον root χρήστη του LDAP server. Συγκεκριμένα με το «slappasswd» μπορούμε να αντικαταστήσουμε τον κωδικό με έναν κωδικοποιημένο με SHA. Δημιουργούμε πρώτα τον χρήστη και μετά προσθέτουμε κωδικό ασφαλείας και στον LDAP χρήστη.

Οι εντολές είναι οι παρακάτω:

```
Create a root Password:
  slappasswd
New password:
Re-enter new password:
 {SHA}b2hqheVCg/H6xXArKpwnlR3eelg=
```

Αντιγράφουμε το αποτέλεσμα του slappasswd μέσα στο /etc/openldap/slapd.conf με τις παρακάτω εντολές:

```
# rootdn directive for specifying a superuser on the database. This is needed
# for syncrepl.
rootdn          "cn=manager,dc=<root>,dc=gr"
rootpw          {SHA}b2hqheVCg/H6xXArKpwnlR3eelg=
```

Πριν εκκινήσουμε τον LDAP πρέπει να δηλώσουμε τον τύπο της βάσης #1, το suffix του domain μας, να δηλώσουμε το rootdn,το rootdn password και το που έχουμε εγκαταστήσει τον ldap (που βρίσκονται τα αρχεία μας).

Κάνουμε τις απαραίτητες αλλαγές στο slapd.conf εκτελώντας τα παρακάτω:

```
vi /etc/openldap/slapd.conf
database      bdb
suffix        " dc=<root>,dc=gr"
rootdn        "cn=manager, dc=<root>,dc=gr"
rootpw        {SHA}b2hqheVCg/H6xXArKpwnlR3eelg=
directory     "/var/lib/ldap"
```

πριν προσθέσουμε το «init.ldif» χρειάζεται να διαγράψουμε τους παλιούς καταλόγους (εάν είχαμε παλιές εγγραφές), τώρα είναι όμως νέα, καθαρή εγκατάσταση και δεν χρειάζεται η παρακάτω εντολή:

```
# rm -rf /var/lib/ldap/*
```

Δημιουργούμε ένα αρχείο «/var/lib/ldap/DB_CONFIG» με τις παρακάτω παραμέτρους:

```
set_cachesize 0 15000000 1
set_lg_regionmax 262144
set_lg_bsize 2097152
set_flags DB_LOG_AUTOREMOVE
```

Εκκίνηση του LDAP

```
service ldap start
```

ΠΑΡΑΡΤΗΜΑ Ι

Το αρχείο διαμόρφωσης “config.php” είναι αυτό που χρησιμοποιείται για να οριστούν περιορισμοί στις τιμές που μπορεί να λάβει ένα attribute.

(config.php στο συνοδευτικό υλικό)

ΠΑΡΑΡΤΗΜΑ ΙΙ

Το αρχείο “testing.webservice.php” είναι αυτό που χρησιμοποιείται για τον έλεγχο της υπηρεσίας.

(testing.webservice.php στο συνοδευτικό υλικό)

ΠΑΡΑΡΤΗΜΑ ΙΙΙ

Περιγραφή λειτουργιών και διαμόρφωσης στο υποέργο LDAP του έργου E-University

Στα πλαίσια του έργου e-University της ‘Κοινωνίας της Πληροφορίας’ και πιο συγκεκριμένα του υποέργου ‘Υπηρεσίας Καταλόγου – LDAP’ απαιτήθηκε η δημιουργία κατάλληλου web services interface για την επικοινωνία του κεντρικού portal κάθε ακαδημαϊκού ιδρύματος (u-Portal) με την υπηρεσία καταλόγου στις περιπτώσεις πραγματοποίησης αλλαγών.

Σύνδεση με εφαρμογή U-Portal

Στην περίπτωση του u-Portal έχουν οριστεί οι ακόλουθες γενικές λειτουργίες προς χρήση από την εφαρμογή:

- Επιβεβαίωση ύπαρξης χρήστη
- Δημιουργία username
- Αλλαγή password χρήστη
- Ενεργοποίηση υπηρεσίας email χρήστη
- Ενεργοποίηση υπηρεσίας dialup χρήστη
- Ενεργοποίηση υπηρεσίας προσωπικής ιστοσελίδας χρήστη

Ο τρόπος χρήσης των παραπάνω λειτουργιών είναι ο ακόλουθος:

- Ο χρήστης αποκτά εγγραφή στην υπηρεσία καταλόγου μέσω εγγραφής του σε ένα από τα υποσυστήματα υποστήριξης χρηστών
- Αιτείται μέσω του uportal της απόκτησης username
- Το uportal μέσω της λειτουργίας ύπαρξης χρήστη επιβεβαιώνει ότι ο χρήστη υπάρχει στην υπηρεσία καταλόγου με βάση των κωδικό αριθμό του στο υποσύστημα εγγραφής
- Το uportal μέσω της λειτουργίας δημιουργίας username δημιουργεί το username του χρήστη στην υπηρεσία καταλόγου
- Τυχόν πρόσβαση σε επιπλέον υπηρεσίες (email, dialup, προσωπική ιστοσελίδα) ενεργοποιείται με την εκτέλεση των αντίστοιχων λειτουργιών
- Σε περίπτωη που απαιτηθεί η αλλαγή του password του χρήστη αυτή μπορεί να γίνει από τη λειτουργία αλλαγής password.

Η εφαρμογή περιέχεται στο αρχείο uportalinterface.php ενώ ο κώδικας των λειτουργιών στο αρχείο uportalinterface.functions.php
Αναλυτικότερα για τις παραπάνω λειτουργίες ισχύουν τα εξής:

- Επιβεβαίωση ύπαρξης χρήστη
    - Όνομα: ExistsUser
    - Είσοδος
        - SubsystemID (τύπος integer, υποχρεωτικό): Περιέχει το νούμερο 0-7 του υποσυστήματος υποστήριξης χρηστών για στο οποίο είχε εγγραφεί ο χρήστης (και είναι η πηγή της εγγραφής του χρήστην υπηρεσία καταλόγου)
        - UserNumber (τύπος string, υποχρεωτικό): Περιέχει τον αριθμό μητρώου του χρήστη στο υποσύστημα που αντιστοιχεί στο SubsystemID
    - Έξοδος
        - Status (τύπος integer): Επιστρέφεται 0 αν ο χρήστης δε βρέθηκε,. 1 αν υπάρχει και -1 σε περίπτωση λάθους κατά την εκτέλεση της λειτουργίας

        - ErrorMessage (τύπος string): Σε περίπτωση αποτυχίας περιέχει κατάλληλο μήνυμα λάθους
        - UserName (τύπος string): Σε περίπτωση που ο χρήστης διαθέτει ήδη username περιέχει το username του
    - Περιγραφή: Αναζητεί το χρήστη στην υπηρεσία καταλόγου με βάση το υποσύστημα στο οποίο ανήκει και τον αριθμό μητρώου στο υποσύστημα αυτό

- Δημιουργία Username
    - Όνομα: CreateUserName
    - Εισοδος
        - . SubsystemID (τύπος integer, υποχρεωτικό): Περιέχει το νούμερο 0-7 του υποσυστήματος υποστήριξης χρηστών για στο οποίο είχε εγγραφεί ο χρήστης (και είναι η πηγή της εγγραφής του χρήστην υπηρεσία καταλόγου)
        - UserNumber (τύπος string, υποχρεωτικό): Περιέχει τον αριθμό μητρώου του χρήστη στο υποσύστημα που αντιστοιχεί στο SubsystemID
        - UserName (τύπος string, υποχρεωτικό): Περιέχει το επιθυμητό username για το χρήστη. Αν το username υπάρχει ήδη η λειτουργία θα αποτύχει.
        - UserPassword (τύπος string, υποχρεωτικό): Περιέχει το επιθυμητό password για το χρήστη. Η λειτουργία είναι δυνατό να ελέγξει το password για ελάχιστο μήκος χαρακτήρων και αν είναι αρκετά προβλέψιμο.
        - UserType (τύπος string, υποχρεωτικό): Ο τύπος του χρήστη. Λαμβάνει τις δυνατές τιμές του ldap attribute edupersonaffiliation
    - Έξοδος
        - Status (τύπος Boolean): Αναφέρει εάν η λειτουργία ήταν επιτυχής ή όχι

        - ErrorMessage (τύπος string): Εάν η λειτουργία δεν ήταν επιτυχής περιέχει κατάλληλο μήνυμα λάθους
    - Περιγραφή: Δημιουργία username για το χρήστη με βάση το υποσύστημα στο οποίο ανήκει και τον αριθμό μητρώου στο υποσύστημα αυτό. Αν το username υπάρχει ήδη αποτυγχάνει

- Αλλαγή Password χρήστη
    - Όνομα: ChangePassword
    - Είσοδος
        - UserName (τύπος string, υποχρεωτικό): Το username του χρήστη
        - UserPassword (τύπος string, υποχρεωτικό): Το νέο επιθυμητό password του χρήστη
    - Έξοδος
        - Status (τύπος Boolean): Αναφέρει εάν η λειτουργία ήταν επιτυχής ή όχι
        - ErrorMessage (τύπος string): Εάν η λειτουργία δεν ήταν επιτυχής περιέχει κατάλληλο μήνυμα λάθους
    - Περιγραφή: Αλλαγή του password του χρήστη σε νέο.
- Ενεργοποίηση υπηρεσίας Email
    - Όνομα: ActivateServiceEmail
    - Είσοδος
        - UserName (τύπος string, υποχρεωτικό): Το username του χρήστη
        - UserEmail (τύπος string, υποχρεωτικό): Το επιθυμητό email του χρήστη. Συνήθως είναι της μορφής <username>@<κάποιο domain>
        - UserAlternateEmail (τύπος string, προαιρετικό): Το προαιρετικό επιθυμητό alternate email του χρήστη
    - Έξοδος

        - Status (τύπος Boolean): Αναφέρει εάν η λειτουργία ήταν επιτυχής ή όχι
        - ErrorMessage (τύπος string): Εάν η λειτουργία δεν ήταν επιτυχής περιέχει κατάλληλο μήνυμα λάθους
    - Περιγραφή: Ενεργοποίηση υπηρεσίας email χρήστη
- Ενεργοποίηση υπηρεσίας Dialup
    - Όνομα: ActivateServiceDialup
    - Είσοδος
        - UserName (τύπος string, υποχρεωτικό): Το username του χρήστη
    - Έξοδος
        - Status (τύπος Boolean): Αναφέρει εάν η λειτουργία ήταν επιτυχής ή όχι
        - ErrorMessage (τύπος string): Εάν η λειτουργία δεν ήταν επιτυχής περιέχει κατάλληλο μήνυμα λάθους
    - Περιγραφή: Ενεργοποίηση υπηρεσίας dialup χρήστη
- Ενεργοποίηση υπηρεσίας προσωπικής ιστοσελίδας
    - Όνομα: ActivateServiceUserPage
    - Είσοδος
        - UserName (τύπος string, υποχρεωτικό): Το username του χρήστη
    - Έξοδος
        - Status (τύπος Boolean): Αναφέρει εάν η λειτουργία ήταν επιτυχής ή όχι
        - ErrorMessage (τύπος string): Εάν η λειτουργία δεν ήταν επιτυχής περιέχει κατάλληλο μήνυμα λάθους
    - Περιγραφή: Ενεργοποίηση υπηρεσίας προσωπικής ιστοσελίδας χρήστη. Θεωρείται ότι ο χρήστης δεν έχει δικαίωμα επιλογής της

μορφής της ιστοσελίδας (θα είναι μορφής users.<domain>/<username>).

ΠΑΡΑΡΤΗΜΑ IV

Paper επί του θέματος όπως παρουσιάστηκε στο πρώτο LDAP Conference στην Κολωνία της Γερμανίας το 2008

Title

Moving LDAP Writes to Web Services

Authors

Kostas Kalevras, Network Operations Centre, National Technical University of Athens, Iroon Polytexneiou 9, 15780 Zografou, Greece
E-Mail: kkalev@noc.ntua.gr

Dimitrios Kalogeras, Network Operations Centre, National Technical University of Athens, Iroon Polytexneiou 9, 15780 Zografou, Greece
E-Mail: dkalo@noc.ntua.gr

Keywords

LDAP, Web Services, SOAP, WSDL

Abstract

The authors administer the Greek School Network Directory Service (based on the LDAP Directory Server by Sun One) which currently contains more than 170,000 entries including school accounts (for email and dialup/ADSL access), teacher accounts and several student accounts. Available services include Email, network access (dialup and ADSL service), personal home pages, VoIP provision and others. Student accounts are scheduled to become available to every middle and high-school student of Greece moving the total Directory Service user population to one million. User administration is done through a feature-full web administration interface (written in PHP) which includes features like:

- Creating attributes based on the value of other attributes or expressions. Any valid php expression (or even function) can be used when calculating the attribute values.
- Maintaining attribute uniqueness (independent of any possible related features of the LDAP service itself).

- Maintaining referential integrity for configurable attributes.
- Performing post operation tasks like creating user directories, sending welcome emails and so on.

Greek School Network is moving towards the e-school framework which apart from the currently available services will include:

- A web portal (sPortal) for student parents. Through this portal, parents can register with the Greek School Network (their account is linked with their children account(s)) and check out their child's progress and status.
- A school administration platform which will move all school operations (student enrollment, classroom management, grading) to the electronic world. The platform will include a central personnel and operations database and full-featured interfaces (written in .NET) running on the school computers. This platform when activated will become the primary source of personnel information for the Directory Service. Therefore, the teacher, administrative personnel and student accounts will first be created on this platform and later be synchronized with the Directory Service.

These new services create new sources of information for the existing Directory Service. Parents will obtain accounts in the web portal while the school administration platform will create accounts for all students and teachers. Allowing these services to administer these entries through plain LDAP poses some serious drawbacks:

- Each service only has knowledge of its own little world (and attributes). The sPortal just needs to create simple parent username/password for access to the Portal. It is not concerned with the fact that the created account might also be entitled to email or VoIP access. Any advanced feature (like attribute uniqueness, referential integrity, dynamic attribute values) must either be implements by the directory server itself or be added in the outside service logic and codebase.
- There is no way to perform post operation tasks like creating user directories.
- Each service is given too much power over the Directory Service. There's almost no control (apart from ACIs) of what is added to the directory and no constraints can be set on the incoming attribute values (except if constraint checking plugin is written for the Directory Service itself).
- The Directory Service needs to cooperate with these new services in order to provide any new service to it's users. Any new ldap attributes required will need to be administered by the outside services (and not only by the Directory Service itself) each time an operation is performed on an account (creation, modification, deletion).

We decided to overcome the above difficulties by creating a web service interface around the already existing user interface. The web service uses WSDL and SOAP over HTTP(S) to create a function interface to all abstract operations needed by the external services. Each time a parent has to be created in sPortal, the portal will call the *CreateParent()* function with appropriate arguments. This function will perform all necessary checks on the arguments and call the internal object creation function of the user administration interface. That way:

- We use the same function backend for both the user administration interface and the web services.
- Complete and configurable logging of all operations is available with much more detail than that provided in LDAP server logs. Logging is contained in one place (on the web service provider) instead of being scattered throughout the directory servers comprising the directory servers. LDAP servers log all operations performed on them (including all of the million searches performed on them), making it very difficult to get a clear and precise picture of all write operations performed on them.
- Computed attributes values are available using any valid php function or expression for computing values.
- Pre and Post operation tasks can be performed through the backend (which can call outside scripts or other web services).
- All operations pass through a single point where we have complete control over what happens and by whom. We can set constraints on attribute values and do extra checks on these values.
- Outside services don't need to have deep knowledge of our entry scheme. They just need to call already defined functions (with the minimum set of arguments) and the web services/backend handles the rest. We are free to change the entry schema whenever we want, adding or removing computed and static attributes to the ones sent by the web service.
- We can impose our own entry expiration policy. The *EntryDelete()* web service function might end up just setting an *active=false* attribute inside the entry allowing us to decide when to actually delete the entry and/or perform any other tasks necessary. This can be really helpful in cases where we want to give an expired account the chance to backup it's files (mailbox, personal web pages and files etc) and notify it's contacts about the related changes.
- A clear, precise and minimal function interface is exported to outside services instead of an abstract protocol like LDAP which demands creating agreements between the Directory Service and outside services on how to perform operations. Outside services don't even have to know what LDAP stands for. They are only concerned with calling the necessary functions (using a commonly used and standard protocol like web serives/WSDL) corresponding to the operations they are performing each time.

In special cases the functional interface could also be used in the case of LDAP reads as well. This can be beneficial in circumstances where an outside service has clearly defined, specific and limited LDAP read needs. Instead of educating the service on the LDAP protocol, how to use it and the LDAP schema used, a functional interface can be agreed and created. Each function exported will provide the caller with the equivalent of a specific LDAP search. This can allow for complex and/or multiple LDAP searches to be performed by the function code and only provide the caller with the computed results (group membership and ldap browsing are strong examples of complex operations which can be hidden to the caller). In order to return the entry attributes the DSML XML language can be used to transfer them back (probably striped off of any unneeded attributes like objectclass and others).

Authentication for the web services is based on HTTP authentication. The web service caller is given a service username/password pair on the Directory Service and uses this when calling the web services through HTTP authentication (except from special cases where the web service caller will use the credentials of the user that the operation corresponds to). The web service itself when binding to the Directory Service will use the credentials provided during http authentication. There are cases where the web service will bind as the user entry for which the operation is requested (such as in the case of the ChangePassword() operaion) and others where the web service will always bind as the service user entry (such as in the case where we just act as a synchronization mechanism between an outside service and the directory service).

A PHP API has also been created as a backend for these web services called LDAP User Management Service (LUMS). It basically provides a set of basic LDAP API functions (search, add, delete, modify, rename, change password), and a strong configuration language. The language allows the administrator to define ldap object types along with their corresponding attributes. For each attribute a number of options is available:

- define an attribute as required, multivalued
- set the attribute type (string, binary, dn, telephone, mail etc)
- define the attribute value source. Can be user inserted, constant, auto increment, function created
- allow for attribute uniqueness
- define extra syntax checking functions
- automatically handle auto incremented attribute values
- define virtual attributes which are used to create attribute mappings

Moreover, pre and post operation functions can be defined while the interface takes care of handling non English char-set attribute values.

The authors believe that LDAP and XML integration will be even tighter in the future. DSML is already available and the XML Enabled Directory Internet drafts envision moving all LDAP operations to the XML space. Creating a Web Service function interface around a Directory Service can prove highly beneficial in centralizing control of ldap write operations while providing a lightweight, well-known, clean and minimal interface for outside services to use.

References

LDAP User Management Service : http://www.sourceforge.net/projects/lums/

Vitae

Kostas Kalevras is a network engineer for the Network Operations Centre of the NTUA. Among other things he is in charge of the LDAP and RADIUS services for the NTUA, Greek School Network and GRNET. He is also a primary developer for the FreeRADIUS project having both developed and maintained a large number of server modules as well as the web based administration front-end dialupadmin. He is also participating in other RADIUS related open-source software projects.

Dimitris Kalogeras was born in Athens, Greece in 1967. He received the Diploma in Electrical Engineering from the National Technical University of Athens (NTUA), Greece in 1990 and PhD in Electrical and Computer Engineering from NTUA in 1996.

From 1993 to 1995 he worked for the NTUA Network Management Center as Data Network Engineer. From 1995 to 1996 he worked for INTRACOM S.A. as an Engineer in Research & Development. From 1997 to 1999 he worked as Technical Consultant for NTUA NMC and GRNET. From 1999 till today he is a Researcher of Institute of Communication and Computer Systems, Department of Electrical and Computer Engineering

9 Ακρωνύμια

AAP	Attribute Acceptance Policies
ACL	Access List
ADSL	Asymmetric Digital Subscriber Line
AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Attribute Release Policies
B2B	Business to Business
BER	Basic Encoding Rules
CA	Certification Authority
CARP	Common Address Redundancy Protocol
CPU	Central Processing Unit
CRL	Certificate Revocation List
DAP	Directory Access Protocol
DEN	Directory Enabled Networking
DER	Distinguished Encoding Rules
DIT	Directory Information Tree
DN	Distinguished Name
DNS	Domain Name System
DoS	Denial of Service
DSML	Directory Services Markup Language
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
OS	Operating System
PKI	Public Key Infrastructure
RADIUS	Remote Authentication Dial In User Service
RAM	Random Access Memory
RFC	Request for Comments
SAML	Security Assertion Markup Language
SASL	Simple Authentication and Security Layer
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SSO	Single Sign On
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDDI	Universal Description, Discovery & Integration
VRRP	Virtual Routing Redundancy Protocol
WS	Web Services
WSDL	Web Service Discription Language

XML	eXtensble Markup Language
AAI	Authentication and Authorization Infrastructure

10 Αναφορές

[1] http://www.openldap.org/

[2] http://sourceforge.net/projects/nusoap/

[3] http://directory.fedora.redhat.com/

[4] http://www.redhat.com/software/rha/directory/

[5] http://www.sun.com/software/products/directory_srvr_ee/

[6] http://www.microsoft.com/

[7] http://www-306.ibm.com/software/tivoli/products/directory-server/

[8] http://www.novell.com/products/edirectory/

[9] http://en.allexperts.com/e/l/li/lightweight_directory_access_protocol.htm

[10] http://www.mozilla.org/mailnews/arch/LDAP_replication2.html

[11] The LDAP Content Synchronization Operation <draft-zeilenga-ldup-sync-05.txt>.

[12] http:///www.go-online.gr

[13] http://www.switch.ch/aai/

[14] http://www.faqs.org/rfcs/rfc4520.html

[15] http://www.faqs.org/rfcs/rfc4521.html

[16] http://www.educause.edu/eduperson/

[17] http://www.freeradius.org/

[18] http://www.utoronto.ca/ns/stats/ldap.html

[19] http://www.bris.ac.uk/is/computing/applications/email/directory/stats.html

[20] Heartbeat: http://www.linux-ha.org/Heartbeat

[21] http://www.w3.org/TR/ws-arch/

[22] http://www.zytrax.com/books/ldap/apa/