ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανακάλυψη υπηρεσιών σε κατανεμημένα περιβάλλοντα με βάση το επίπεδο της ποιότητας των υπηρεσιών χρησιμοποιώντας την πλατφόρμα GRIA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δήμητρα Α. Κόλλια

Επιβλέπων : Συμεών Παπαβασιλείου
Επ. Καθηγητής Ε.Μ.Π

Αθήνα, Ιανουάριος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανακάλυψη υπηρεσιών σε κατανεμημένα περιβάλλοντα με βάση το επίπεδο της ποιότητας των υπηρεσιών χρησιμοποιώντας την πλατφόρμα GRIA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δήμητρα Α. Κόλλια

Επιβλέπων : Συμεών Παπαβασιλείου
Επ. Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Ιανουαρίου 2008.

.....	.....	.....
Σ.Παπαβασιλείου	Β.Μάγκλαρης	Δ.Καλογεράς
Επ. Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Ερευνητής ΕΠΙΣΕΥ

Αθήνα, Ιανουάριος 2008

.....

Δήμητρα Α. Κόλλια

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

Copyright © Δήμητρα Α. Κόλλια
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

ΠΕΡΙΛΗΨΗ

Η χρήση πόρων που παρέχονται σε κατανεμημένα περιβάλλοντα έκανε επιτακτική την ανάγκη για ανάπτυξη εργαλείων που θα περιγράφουν και θα εξασφαλίζουν το απαιτούμενο επίπεδο ποιότητας υπηρεσιών κατά την ανάθεση και δέσμευση των πόρων που αυτές παρέχουν. Το επίπεδο ποιότητας μιας υπηρεσίας μπορεί να περιγραφεί με την χρήση Συμφωνιών Ποιότητας Υπηρεσιών (Service Level Agreements SLAs). Σκοπός της διπλωματικής είναι η αναζήτηση πόρων και υπηρεσιών σε ένα κατανεμημένο περιβάλλον με βάση το επίπεδο της ποιότητας των υπηρεσιών που προσφέρουν και το οποίο θα εκφράζεται μέσω ενός SLA που συμφωνείται μεταξύ του παροχέα και του πελάτη. Για την υλοποίηση χρησιμοποιείται η πλατφόρμα GRIA και συγκεκριμένα τα εργαλεία που παρέχει για την δημιουργία GRIA υπηρεσιών καθώς και οι υπηρεσίες που προσφέρει για τη δημιουργία SLAs.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Grid, distributed environment, Service Level Agreement (SLA), Contextualised Registry, GRIA, Web Service, resource, concept, Registry Domain Model (RDM), ooXmlQL query, service discovery

ABSTRACT

The use of the resources provided in distributed environments has made it an urgent need to develop tools that will describe and ensure the required quality of service during the commitment and managing of the resources they provide. The level of quality service can be described by the use Service Level Agreements (Service Level Agreements SLAs).

The purpose of this thesis is the discovery of resources and services in a distributed environment based on the level of quality of services they offer and which will be expressed through an SLA which has already be negotiated between the service provider and the client. Platform GRIA has been used for the implementation and particularly the tools provided for the development of GRIA services and the services it offered for creation and management of SLAs.

KEYWORDS

Grid, distributed environment, Service Level Agreement (SLA), Contextualised Registry, GRIA, Web Service, resource, concept, Registry Domain Model (RDM), ooXmlQL query, service discovery

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΕΙΣΑΓΩΓΗ

Σκοπός αυτής της διπλωματικής εργασίας είναι η αναζήτηση πόρων σε δίκτυο πλέγματος διαφόρων υπολογιστών σύμφωνα με ένα συγκεκριμένο επίπεδο ποιότητας υπηρεσιών που ζητείται από τον χρήστη. Τα αποτελέσματα της αναζήτησης θα είναι οι υπηρεσίες που μπορούν να παρέχουν τους ζητούμενους πόρους, τα SLA Templates που θα ικανοποιούν τα ζητούμενα κριτήρια ποιότητας της υπηρεσίας και ο πάροχος (υπολογιστής) που παρέχει τα παραπάνω. Ο χρήστης μπορεί στη συνέχεια να συμφωνήσει το SLA για την συγκεκριμένη υπηρεσία. Για την δημιουργία και παροχή των παραπάνω πόρων χρησιμοποιήθηκε η πλατφόρμα GRIA.

Στα επόμενα κεφάλαια, αναλύεται η δημιουργία των απαιτούμενων πόρων στο περιβάλλον του GRIA, η δημιουργία του χώρου όπου καταχωρούνται, η υλοποίηση της διαδικασίας αναζήτησης και ο τρόπος χρήσης των αποτελεσμάτων.

Πιο συγκεκριμένα, στο πρώτο κεφάλαιο αναλύεται η έννοια του Πλέγματος, η έννοια του SLA και πώς χρησιμοποιείται ένα SLA για την εύρεση υπηρεσιών σε ένα Πλέγμα.

Στο δεύτερο κεφάλαιο, γίνεται ανάλυση της πλατφόρμας GRIA και των βασικών υπηρεσιών της που θα χρησιμοποιηθούν στην υλοποίηση.

Στο τρίτο κεφάλαιο, περιγράφεται λεπτομερώς η υλοποίηση των υπηρεσιών του GRIA και ο τρόπος χρησιμοποίησης τους, καθώς και η δημιουργία μίας καινούριας GRIA υπηρεσίας.

Στο τέταρτο κεφάλαιο, περιγράφεται η αρχιτεκτονική του συστήματος και η υλοποίηση του η οποία περιλαμβάνει την δημιουργία ενός registry, την καταχώρηση των πόρων σε αυτό καθώς και την αναζήτηση τους στην συνέχεια από τον χρήστη.

Τέλος στο πέμπτο κεφάλαιο παρουσιάζονται τα σενάρια που υλοποιήθηκαν και τα αποτελέσματα τους. Στα σενάρια αυτά ο χρήστης υποβάλλει ερωτήματα στο registry στα οποία εκφράζει τους περιορισμούς που θέλει να ικανοποιούν οι ζητούμενες υπηρεσίες. Στην συνέχεια ο χρήστης μπορεί να χρησιμοποιήσει τα αποτελέσματα για να συμφωνήσει ένα SLA.

ΚΕΦΑΛΑΙΟ 1

1. Η Ποιότητα των Υπηρεσιών σε κατανεμημένα περιβάλλοντα

1.1. Τι είναι Πλέγμα (Grid)

Πλέγμα (Grid) είναι η συνεργατική, κοινή χρήση πόρων (resources) και επίλυση προβλημάτων σε εικονικούς οργανισμούς που αποτελούνται από πολλούς επιμέρους οργανισμούς. Σύμφωνα με το [1] η λειτουργία ενός Πλέγματος συνοψίζεται στις παρακάτω τρεις λειτουργίες:

1) Ένα πλέγμα συγκεντρώνει και συνδυάζει πόρους και χρήστες που δεν υποβάλλονται σε κάποιον κεντρικό έλεγχο αλλά ανήκουν σε διαφορετικά περιβάλλοντα ελέγχου και διαχείρισης όπως για παράδειγμα ένας υπολογιστής γραφείου του χρήστη έναντι ενός κεντρικού υπολογιστή που πραγματοποιεί τον έλεγχο. Σε ένα Πλέγμα μπορεί να υπάρχουν διαφορετικές μονάδες διαχείρισης της ίδιας επιχείρησης ή διαφορετικών επιχειρήσεων όπου γίνεται ο χειρισμός διαφόρων σημαντικών θεμάτων όπως είναι η ασφάλεια, οι αρχές πρόσβασης, οι πληρωμές, δημιουργία ομαδοποιήσεων για βελτίωση της διαχείρισης.

2) Ένα Πλέγμα χρησιμοποιεί πρωτόκολλα τα οποία είναι πρότυπα, ανοιχτά και γενικού σκοπού και διεπαφές για την επικοινωνία με τον χρήστη. Δομείται από πρωτόκολλα για πολλαπλές χρήσεις και διεπαφές μέσω των οποίων γίνεται ο χειρισμός βασικών θεμάτων όπως η πιστοποίηση και η εξουσιοδότηση, η ανακάλυψη πόρων και η πρόσβαση στους παρεχόμενους πόρους. Τα πρωτόκολλα αυτά πρέπει να είναι πρότυπα και ανοιχτά.

3) Σκοπός του είναι να συνδυάζει τους πόρους που παρέχει έτσι ώστε να προσφέρονται στους χρήστες διαφορετικά επίπεδα ποιότητας υπηρεσιών. Τα επίπεδα αυτά ποιότητας υπηρεσιών μπορεί να έχουν σχέση για παράδειγμα με τον χρόνο απόκρισης, με τον ρυθμό μετάδοσης των δεδομένων , με την ασφάλεια και την ταυτόχρονη ανάθεση πολλών και διαφορετικών τύπων πόρων ώστε να ικανοποιούν ακόμα και τις πιο σύνθετες απαιτήσεις των χρηστών. Έτσι η χρησιμότητα του συνδυασμένου συστήματος καταλήγει να είναι σημαντικά μεγαλύτερη από το άθροισμα των επιμέρους τμημάτων της. Με αυτό τον τρόπο η χρησιμότητα και η ευελιξία αυτού του συνδυασμένου συστήματος καταλήγει να είναι μεγαλύτερη από το απλό άθροισμα των επιμέρους τμημάτων του συστήματος.

1.2. Διαχείριση της ποιότητας υπηρεσιών στο Grid Computing

Το Πλέγμα εμφανίστηκε ως ένα πρότυπο παράδειγμα διαμοιραζόμενων μέσων με σκοπό την καλύτερη συνεργασία μεταξύ των διαφορετικών συστημάτων παροχής υπηρεσιών και άλλων πόρων αλλά και την βελτιστοποίηση της ανάθεσης και χρησιμοποίησης πόρων. Ένα Πλέγμα αποτελείται από ένα σύνολο κόμβων. Κάθε κόμβος μπορεί να είναι ένα σύστημα που διαχειρίζεται ένα σύνολο από πόρους. Κάθε κόμβος μπορεί να είναι ένα μοναδικό σύστημα ή μία ομάδα συστημάτων (cluster). Οι πόροι, τους οποίους διαχειρίζονται οι κόμβοι, μπορεί να είναι ένα δίκτυο, ένα σύστημα, κάποια εφαρμογή, ένα αποθηκευτικό μέσο, CPU ή γενικά κάποιο άλλο είδος εξοπλισμού.

Το Grid ξεκίνησε να εφαρμόζεται σε ακαδημαϊκά περιβάλλοντα και για ακαδημαϊκούς σκοπούς για αυτό η βασική αρχή για την λειτουργία του ήταν αυτή της "βέλτιστης προσπάθειας" ("best effort"). Στην συνέχεια όμως άρχισε να χρησιμοποιείται όλο και περισσότερο για εμπορικούς σκοπούς. Εμφανίστηκε έτσι η ανάγκη για παροχή πολύ αυστηρότερων εγγυήσεων στους χρήστες από ότι μπορούσε να παρέχει η αρχή της «βέλτιστης προσπάθειας» για τους πόρους που θα τους παρείχε.

Οι εγγυήσεις και δεσμεύσεις που απαιτούνται προς τους χρήστες των πόρων μπορεί να γίνει με την χρήση Συμφωνιών Ποιότητας Υπηρεσιών (Service Level Agreements SLAs).

1.3. Τι είναι ένα SLA, γιατί και πώς πρέπει να χρησιμοποιηθεί σε κατανεμημένο περιβάλλον

Ένα SLA είναι μία επίσημα ορισμένη συμφωνία μεταξύ δύο μερών που έχει προκύψει μετά από διαπραγμάτευση. Αποτελεί συμβόλαιο ανάμεσα σε καταναλωτές και στους παρόχους υπηρεσιών (service providers) που παρέχουν στους καταναλωτές τις υπηρεσίες τους ή ανάμεσα σε πάροχους υπηρεσιών [2]. Σε γενικές γραμμές στο SLA καθορίζεται το γενικό πλαίσιο για την παροχή των υπηρεσιών, οι προτεραιότητες που πρέπει να τεθούν στην εξυπηρέτηση των καταναλωτών και την ανάθεση των πόρων, ο καταμερισμός των ευθυνών σε καταναλωτές και παρόχους, οι εγγυήσεις που παρέχονται στους καταναλωτές για την χρησιμοποίηση των πόρων από τους παρόχους και οτιδήποτε άλλο προσδιορίζει το επίπεδο της ποιότητας των παρεχόμενων υπηρεσιών. Αυτό σημαίνει ότι μπορεί να καθορίζει τα επίπεδα διαθεσιμότητας ενός πόρου, το επίπεδο εξυπηρετικότητας, η απόδοση, η λειτουργικότητα, η παρεχόμενη ασφάλεια, τα επίπεδα ανεκτής καθυστέρησης. Σε ένα SLA μπορεί ακόμα να καθορίζεται η χρέωση για την χρήση των πόρων που παρέχει η υπηρεσία καθώς επίσης και οι κυρώσεις που θα επιβάλλονται σε περίπτωση παραβίασης των όρων που έχουν συμφωνηθεί στο SLA από οποιοδήποτε από τα δύο μέρη.

Σύντομη ιστορική αναδρομή

Τα SLAs χρησιμοποιούνται από τα τέλη της δεκαετίας του '80 από τηλεπικοινωνιακούς παρόχους ως μέρος των συμβολαίων τους με τους πελάτες τους. Στην συνέχεια τμήματα του Information Technology (IT) σε μεγαλύτερες

επιχειρήσεις υιοθέτησαν την ιδέα να χρησιμοποιήσουν SLAs με τους πελάτες τους – χρήστες σε άλλα τμήματα μέσα στην ίδια εταιρεία ώστε να υπάρχει η δυνατότητα να γίνεται σύγκριση της ποιότητας υπηρεσιών που παρέχεται με αυτήν που έχουν δεσμευθεί να παρέχουν και έτσι να θεωρήσουν και την εναλλακτική να παρέχουν IT υπηρεσίες σε κάποια εξωτερική εταιρεία.

Δομή SLA

Υπάρχουν αρκετές προτεινόμενες δομές αλλά όλες έχουν ένα κοινό βασικό τμήμα το οποίο και περιγράφεται παρακάτω.

Ένα SLA Template αποτελείται από ένα σύνολο τεχνικών και μη τεχνικών παραμέτρων και παρέχεται από μία διαχειριστική περιοχή (domain). Ο ορισμός του SLA έχει ένα βασικό μέρος το SLS (Service Level Specification) που αποτελείται από το σύνολο των τεχνικών παραμέτρων με τις αντίστοιχες τιμές τους. Το SLS υλοποιείται στο τμήμα SLO (Service Level Object) [3] και καθορίζονται οι στόχοι οι οποίοι πρέπει να επιτευχθούν από το SLS [2]. *Στην υλοποίηση της εργασίας χρησιμοποιήθηκε η μορφή των SLA Templates που παρέχεται από την πλατφόρμα GRIA και η οποία εξηγείται αναλυτικά παρακάτω.*

Γιατί μπορούν να χρησιμοποιηθούν τα SLAs στα GRID

Η τεχνολογία Πλέγματος κινείται όλο και περισσότερο προς την τεχνολογία των διαδικτυακών υπηρεσιών (Web Services) (Open Grid Services Architecture) και αφού τα SLAs ήδη χρησιμοποιούνται ευρέως με τις υπηρεσίες διαδικτύου, εφαρμόζονται με επιτυχία και για την διασφάλιση της ποιότητας υπηρεσιών και πόρων προς τους πελάτες [4].
Στο [5] αναφέρεται ότι στην περίπτωση του Grid το SLA θα είναι η συμφωνία που θα ορίζει τι πόροι θα παρέχονται στον πελάτη και πόσο θα χρεώνεται για την χρήση των πόρων της υπηρεσίας. Για παράδειγμα ένα SLA μπορεί να ορίζει ότι θα επιτρέπεται να φορτωθούν μέχρι 1 Tb δεδομένων ανά μήνα σε κάποιο αποθηκευτικό χώρο, ότι ο χρήστης θα χρεώνεται με 1 ευρώ ανά Gb και ότι μπορούν να τρέχουν μέχρι 30 εφαρμογές ταυτόχρονα με χρέωση ένα ευρώ ανά CPU ανά ώρα.

Πώς χρησιμοποιούνται τα SLAs σε Grid περιβάλλοντα

Για να μπορεί ένα Πλέγμα να παρέχει αυτές τις εγγυήσεις μέσω SLAs θα έπρεπε να αναπτυχθούν εργαλεία που θα έδιναν την δυνατότητα να συλλέγονται δεδομένα και πληροφορίες για τον βαθμό και το είδος της χρησιμοποίησης των πόρων από τους χρήστες και να γίνονται μετρήσεις πάνω σε αυτά τα δεδομένα. Θα έπρεπε να παρέχουν επιπλέον την δυνατότητα, μέσω των αποτελεσμάτων τους να παρακολουθείται η χρησιμοποίηση και η διαθεσιμότητα των πόρων και να γίνεται διαχείριση της απόδοσης και της λειτουργίας τους. Επιπλέον τα αποτελέσματα από την επεξεργασία των παραπάνω δεδομένων θα πρέπει να αξιολογούνται και να συγκρίνονται με τις δεσμεύσεις προς τους καταναλωτές που έχουν προηγηθεί. Η αυτοματοποίηση της παραπάνω διαδικασίας είναι πολύ σημαντική εξαιτίας της μεγάλης της πολυπλοκότητας.

Έχουν γίνει διάφορες μελέτες και εργασίες πάνω σε αυτά τα θέματα που αφορούν την χρήση και διαχείριση των SLAs σε Grid περιβάλλοντα και για την επίλυση διαφόρων συναφών προβλημάτων. Παρακάτω δίνονται κάποιες από αυτές.

Ένα σημαντικό θέμα είναι ο ορισμός των SLAs και η διαχείριση τους σε περιβάλλοντα πλέγματος. Ένα πρόβλημα που υπάρχει στον τομέα αυτό είναι ότι στα εμπορικά περιβάλλοντα Πλέγματος μπορεί να υπάρχουν πολλές ομάδες συστημάτων (clusters) που σχηματίζουν το Πλέγμα μιας επιχείρησης ή ακόμα και πολλά συνεργαζόμενα Πλέγματα διαφορετικών επιχειρήσεων που παρέχουν όμως μία κοινή διεπαφή για την παροχή πόρων στους πελάτες καθώς επίσης και πολλοί πελάτες. Αυτό συνεπάγεται ότι υπάρχουν πολλά SLAs με πολλά και διαφορετικά metrics (μονάδες μέτρησης για κάθε πόρο) το καθένα, ακόμα και για την ίδια εφαρμογή για τα οποία όμως οι μετρήσεις γίνονται τοπικά όπου ορίζονται. Επομένως θα πρέπει να υπάρχει ένα Σύστημα Διαχείρισης SLA που θα συγκεντρώνει τα διανεμημένα SLAs και θα τα συναθροίζει σε ένα γενικότερο SLA. Στο [4] παρουσιάζεται η αρχιτεκτονική ενός τέτοιου συστήματος διαχείρισης SLA για ορισμό και έλεγχο SLAs ώστε να αντιμετωπίζεται το παραπάνω πρόβλημα. Η αρχιτεκτονική αυτή βασίζεται σε ένα δίκτυο από πληρεξούσιους (proxies) που μπορούν να επικοινωνούν μεταξύ τους. Καθένας από αυτούς διατηρεί SLAs τα οποία πρέπει να τηρούνται μέσα στο διαχειριστικό περιβάλλον (administrative domain) που διαχειρίζεται ο proxy. Τα SLAs είτε έχουν συμφωνηθεί μεταξύ των διαχειριστικών proxies ή ορίζονται σε αυτούς μεταξύ των πελατών. Οι διαχειριστικοί πληρεξούσιοι (management proxies) είναι υπεύθυνοι για τον αυτόματο χειρισμό και έλεγχο των δεδομένων, για την αξιολόγηση και εκτίμηση των καταχωρημένων SLAs. Επίσης στο [4] παρουσιάζεται μία σαφής και ευέλικτη γλώσσα για την δημιουργία και τον ορισμό των SLAs ώστε να είναι κατανοητά σε όλους τους διαφορετικούς παρόχους υπηρεσιών αλλά και για να μπορεί το σύστημα διαχείρισης να τα καταλαβαίνει πλήρως, να τα ελέγχει και να τα βελτιώνει. Ο management proxy επιτρέπει την αιτιολόγηση για την γενική κατάσταση των SLAs που σχετίζεται με το γενικό πλαίσιο κάποιας εφαρμογής σε πολλαπλά διαχειριστικά περιβάλλοντα, επικοινωνώντας και ρωτώντας τους αντίστοιχους management proxies, αποκτώντας πληροφορίες από μετρήσεις από πολλούς proxies αν χρειάζεται και πραγματοποιώντας μία εμπεριστατωμένη εκτίμηση του SLA. Η διαδικασία συλλογής μετρήσεων και εκτίμησης του SLA αυτοματοποιείται και βασίζεται στην τεχνολογία των διαδικτυακών εφαρμογών (Web Services) αφού όπως ειπώθηκε η τεχνολογία πλέγματος κινείται προς αυτήν την κατεύθυνση. (Το σενάριο θεωρεί το HP Utility Data Center ως ένα τυπικό Εμπορικό Grid περιβάλλον ανάπτυξης.)

Ένα άλλο βασικό θέμα είναι ότι οι εφαρμογές για τα Grid νέας γενιάς (Next Generation Grid) απαιτούν την ύπαρξη ενός Grid middleware δηλαδή ενός λογισμικού που θα μεσολαβεί μεταξύ των διεπαφών προς τους χρήστες και του συστήματος ανάθεσης πόρων που θα παρέχει έναν ευέλικτο μηχανισμό διαπραγμάτευσης που θα υποστηρίζει διάφορους τρόπους εγγυήσεως της ποιότητας των υπηρεσιών (Quality of Service – QoS). Η εγγυήσεις για το QoS πρέπει να καλύπτει ταυτόχρονες αναθέσεις και κατανομές διαφόρων πόρων όπως ταχύτητα επεξεργαστή, χωρητικότητα αποθήκευσης, ή εύρος ζώνης δικτύου τα οποία θα καθορίζονται κατά την συμφωνία του SLA.
Έως τώρα υπάρχει ένα κενό μεταξύ των δυνατοτήτων του υπάρχοντος Grid middleware και των συστημάτων διαχείρισης πόρων τα οποία βρίσκονται κάτω από αυτό, όσο αναφορά την υποστήριξη τους για QoS και διαπραγμάτευση SLA. Στα [6],

[7] παρουσιάζεται μία προσέγγιση η οποία κλείνει αυτό το κενό. Εισάγεται η αρχιτεκτονική του Virtual Resource Manager (Εικονικός Διαχειριστής Πόρων), και δίνεται έμφαση στα βασικά του χαρακτηριστικά για διαχείριση του QoS όπως run-time responsibility, ταυτόχρονη ανάθεση πόρων και ανεκτικότητα σε σφάλματα, προσομοίωση πόρων, απόκρυψη πληροφοριών, αυτόνομη παροχή και ελαφρά ομαδοποίηση (intergration) των ήδη υπαρχόντων εγκαταστάσεων συστημάτων διαχείρισης πόρων.
Στο [8] παρουσιάζεται μια άλλη αρχιτεκτονική στα πλαίσια του project HPC4U που επίσης ασχολείται με το παραπάνω θέμα και παρουσιάζει επιπλέον μηχανισμούς για την αντιμετώπιση παραβιάσεων του SLA και τις ενέργειες που πρέπει να γίνουν για την εξασφάλιση πόρων με άλλο τρόπο όπως είναι η μετακίνηση από τον κόμβο εξυπηρέτησης σε άλλο κόμβο μέσα σε συστήματα που αποτελούνται από ομάδες συστημάτων (cluster) και για επεξεργασίες που γίνονται σε έναν κόμβο. Πολλές ερευνητικές εργασίες εστιάζουν στην κατανόηση των απαιτούμενων μηχανισμών στο επίπεδο του Grid middleware. Μόνο αυτό όμως δεν αρκεί. Τα συστήματα διαχείρισης πόρων που υπόκεινται σε αυτό το Grid middleware πρέπει να παρέχουν ένα αυξανόμενο επίπεδο ποιότητας υπηρεσιών (QoS) από τη στιγμή που παρέχουν τους πόρους τους σε περιβάλλοντα Πλέγματος.

Το Grid Computing όπως είπαμε και παραπάνω θεωρείται το μελλοντικό παράδειγμα εφαρμογής του Grid για επιχειρηματικές εφαρμογές. Μια επιχειρηματική εφαρμογή που τρέχει σε ένα Πλέγμα χωρίζεται σε ένα σύνολο υποεφαρμογών που περιορίζονται από SLAs και απαιτούν διαφορετικά είδη υπηρεσιών και πόρων όπως επεξεργαστές, αποθήκευση δεδομένων, πάροχοι υπηρεσιών (service providers), και δικτυακές συνδέσεις. Στο [9] θεωρείται μια γενική περίπτωση στην οποία οι εφαρμογές διασπώνται σε επιμέρους υποεφαρμογές και δείχνουν τις σχέσεις προτεραιότητας μεταξύ τους. Το πρόβλημα έγκειται στην εύρεση της βέλτιστης ανάθεσης πόρων που ελαχιστοποιεί το συνολικό κόστος ενώ παράλληλα τηρούνται τα Service Level Agreements κατά τον χρόνο εκτέλεσης. Παρέχει ένα πλαίσιο εργασίας για την δημιουργία ευριστικών λύσεων για αυτό το NP-hard πρόβλημα και παρουσιάζει ένα παράδειγμα μίας τέτοιας ευριστικής λύσης.

Ο διαμοιρασμός των πόρων στα συμμετέχοντα μέρη μέσα σε συνεργασίες Πλέγματος συνήθως γίνεται με βάση καθορισμένους μηχανισμούς. Σε τέτοιες περιπτώσεις προκύπτουν σημαντικά θέματα ανταγωνιστικής πολιτικής, που οδηγούν πολλές φορές σε ενοποίηση των συμμετεχόντων και των πόρων μέσω της φυσικής συνένωσης διαφορετικών οργανισμών. Έτσι οι κάτοχοι των πόρων μπορεί να θέλουν να εξουσιοδοτήσουν σε έναν ή περισσότερους εικονικούς οργανισμούς (VOs) το δικαίωμα να χρησιμοποιήσουν συγκεκριμένους πόρους που υπόκεινται σε τοπικές πολιτικές χρήσης και SLAs και κάθε εικονικός οργανισμός (VO) μπορεί να θέλει να χρησιμοποιήσει εκείνους τους πόρους που υπόκεινται στις δικές του πολιτικές χρήσης. Στο [11] περιγράφεται το GRUBER το οποίο είναι ένα μεσολαβητικό σχήμα, μία αρχιτεκτονική και συλλογή εργαλείων για ορισμό SLAs, για χρήση διαφόρων πόρων και επιβολή τους σε ένα περιβάλλον πλέγματος που επιτρέπει στους πόρους που βρίσκονται σε ανεξάρτητες τοποθεσίες να μπορούν να διαμοιράζονται μεταξύ πολλών κοινοτήτων χρηστών.

Στο [12] παρουσιάζεται μία επέκταση του Gruber, ο DI-Gruber που αναπτύχθηκε ως ένας κατανεμημένος διαμεσολαβητής (broker) Πλέγματος για πόρους βασιζόμενος σε USLAs (SLAs για την χρησιμοποίηση των πόρων) και επιτρέπει να

συνυπάρχουν και να συνεργάζονται πολλά σημεία απόφασης σε πραγματικό χρόνο. Ο DI-Gruber αντιμετωπίζει θέματα όπως, ο τρόπος με τον οποίο τα USLAs μπορούν να αποθηκευτούν, να ανακτηθούν από εκεί που είναι αποθηκευμένα και να διανεμηθούν αποδοτικά σε ένα μεγάλο κατανεμημένο περιβάλλον.

Ένα άλλο σημαντικό θέμα είναι η ανάθεση των πόρων να γίνεται με βάση τις ανάγκες των χρηστών και των όρων του SLA. Να γίνεται δηλαδή εξέταση του SLA ώστε να διαπιστωθεί αν έχει την δυνατότητα μια ομάδα συστημάτων (cluster) να δεχτεί την διεκπαιρέωση μιας εργασίας ή εφαρμογής (job) ώστε να την πραγματοποιήσει σύμφωνα με το SLA που έχει συμφωνηθεί για αυτήν.

Οι εργασίες (jobs) που υποβάλλονται σε ένα cluster έχουν διαφορετικές απαιτήσεις μεταξύ τους που εξαρτώνται από τις ανάγκες και τις προσδοκίες των χρηστών. Επομένως στην υπολογιστικότητα ομάδων συστημάτων (cluster computing) που βασίζεται στην ωφελιμότητα, τα συστήματα διαχείρισης πόρων για cluster (RMSs) πρέπει να γνωρίζουν αυτές τις απαιτήσεις ώστε να αναθέσουν τους πόρους όσο πιο αποδοτικά γίνεται. Τα SLAs μπορούν να χρησιμοποιηθούν για να διαφοροποιήσουν την αξία των διαφόρων εργασιών (jobs) που υποβάλλονται στο cluster αφού καθορίζουν τις συνθήκες και τα επίπεδα υπηρεσίας που το RMS του cluster συμφωνεί να προσφέρει για κάθε διαφορετική εργασία. Άρα το SLA λειτουργεί ως συμβόλαιο μεταξύ ενός χρήστη και του cluster μέσω του οποίου ο χρήστης μπορεί να διεκδικήσει αποζημίωση σε περίπτωση που το σύστημα διαχείρισης πόρων RMS του cluster δεν δύναται να διαθέσει την ζητούμενη υπηρεσία. Στο [13] παρουσιάζεται μία τεχνική διάθεσης πόρων με αναλογικό μοίρασμα που λέγεται LibraSLA και η οποία λαμβάνει υπ'όψιν της το πόσο ωφέλιμο είναι να γίνουν αποδεκτές νέες εργασίες (jobs) μέσα στο cluster, με βάση τα όσα έχουν οριστεί στο SLA τους. Οι απαιτήσεις των SLAs που λαμβάνει υπ'όψιν του το LibraSLA μπορεί να είναι:

1) το είδος του χρονικού περιθωρίου (deadline) για την εκτέλεση του job, αν δηλαδή μπορεί να καθυστερήσει η εκτέλεση του job.
2) Το τελικό χρονικό όριο για το πότε το job πρέπει να τελειώσει
3) το χρηματικό ποσό που θα ξοδευτεί για να τελειώσει το job
4) το ποσοστό κυρώσεων για αποζημίωση στον χρήστη σε περίπτωση αποτυχίας ικανοποίησης του χρονικού περιθωρίου.

Για την πραγματοποίηση των διαπραγματεύσεων μεταξύ των δύο μερών για την δημιουργία ενός SLA είναι απαραίτητη η ύπαρξη ενός πρωτοκόλλου. Την λειτουργία αυτή επιτελεί το πρωτόκολλο SNAP. Ένα βασικό πρόβλημα στην κατανεμημένη υπολογιστικότητα (distributed computing), είναι να αντιστοιχήσουμε δραστηριότητες όπως είναι ο υπολογισμός του ρυθμού μεταφοράς των δεδομένων σε πόρους που ικανοποιούν συγκεκριμένες απαιτήσεις απόδοσης όπως κόστος, ασφάλεια ή άλλες μονάδες μετρήσεις (metrics) της ποιότητας υπηρεσιών (QoS). Η δημιουργία τέτοιων αντιστοιχίσεων απαιτεί την διαπραγμάτευση μεταξύ της εφαρμογής και των πόρων, ώστε να ανακαλύψει, να δεσμεύσει, να αποκτήσει, να διαμορφώσει και να ελέγξει τους πόρους που θέλουν. Οι ως τώρα προσεγγίσεις για διαχείριση πόρων τείνουν να ειδικεύονται σε συγκεκριμένες τάξεις πόρων και προχωρούν σε συντονισμό μεταξύ πόρων μόνο σε περιορισμένες περιπτώσεις. Παρουσιάζεται στο [14] ένα μοντέλο διαχείρισης πόρων που ξεχωρίζει τρία είδη SLAs ανεξάρτητα από τους πόρους: διανομή διαθεσιμότητας, εκτέλεση δραστηριοτήτων και σύνδεση των δραστηριοτήτων σε διαθεσιμότητες αντίστοιχα. Καθορίζεται επίσης το πρωτόκολλο

Negotiation and Acquisition Protocol (SNAP) που υποστηρίζει αξιόπιστη διαχείριση απομακρυσμένων SLAs. Εξηγείται επίσης πώς το SNAP μπορεί να οριστεί στο πλαίσιο του Globus Toolkit.

1.4. Αναζήτηση πόρων με βάση την Ποιότητα Υπηρεσιών που προσφέρουν

Ένα άλλο βασικό ζήτημα, το οποίο αποτελεί και το κύριο θέμα αυτής της εργασίας, είναι η αναζήτηση και ανακάλυψη διαδικτυακών υπηρεσιών (Web Services) και κυρίως διαδικτυακών υπηρεσιών που παρέχονται σε περιβάλλοντα πλέγματος. Η αναζήτηση επιθυμείται να γίνεται με βάση το επίπεδο της ποιότητας των υπηρεσιών (QoS) που παρέχουν στους πόρους τους και το οποίο έχει οριστεί στο SLA, που έχει συμφωνηθεί μεταξύ του παρόχου της υπηρεσίας με τον πελάτη που θα την χρησιμοποιήσει.

Η ανακάλυψη Διαδικτυακών Υπηρεσιών (Web services) χρησιμοποιώντας τεχνικές που στηρίζονται σε λέξεις κλειδιά , οι οποίες προσφέρονται από τα υπάρχοντα UDDI APIs (όπως π.χ Inquiry API), μπορεί να μην παρέχουν αποτελέσματα ακριβώς σύμφωνα με τις ανάγκες των πελατών (clients). Οι πελάτες όταν ψάχνουν για Διαδικτυακές υπηρεσίες, ψάχνουν για αυτές που ικανοποιούν τις απαιτήσεις τους κυρίως για την ολική λειτουργικότητα και την παρεχόμενη ποιότητα υπηρεσίας (QoS). Πρότυπα όπως το UDDi, WSDL, SOAP έχουν επεκταθεί ώστε να μπορούν να παρέχουν αναζήτηση διαδικτυακών υπηρεσιών με βάση την ποιότητα των υπηρεσιών. Να μπορεί ο πελάτης να ελέγχει και να διαχειρίζεται την ανακάλυψη διαδικτυακών υπηρεσιών σε προσβάσιμα μητρώα υπηρεσιών (service registries). Τα ήδη υπάρχοντα μητρώα υπηρεσιών όπως το UDDI δεν υποστηρίζουν αναζήτηση διαδικτυακών υπηρεσιών με κριτήρια ποιότητας υπηρεσιών αλλά κυρίως με βάση λέξεις κλειδιά που έχουν να κάνουν με το είδος και την λειτουργικότητα των υπηρεσιών. Έτσι δεν είναι δυνατή η αναζήτηση διαδικτυακών υπηρεσιών που έχουν παρόμοια λειτουργικότητα αλλά σε διαφορετικό επίπεδο ποιότητας υπηρεσιών. Αλλά ακόμα και σε επεκτάσεις των μητρώων ώστε να υποστηρίζουν και αναζήτηση με βάση την ποιότητα των υπηρεσιών, υπάρχει το πρόβλημα ότι τα κριτήρια ποιότητας υπηρεσιών δίνονται και εξασφαλίζονται από τους παρόχους των υπηρεσιών με αποτέλεσμα να υπάρχουν πολλές φορές αμφιβολίες ως προς την αξιοπιστία τους και την ανανέωση τους όταν υπάρξουν αλλαγές. Στο [15] προτείνεται μία λύση σε αυτό το πρόβλημα που εισάγει την Συνάρτηση Σχετικότητας Διαδικτυακής υπηρεσίας (Web Service Relevancy Function - WsRF) που χρησιμοποιείται για την μέτρηση της κατάταξης της συνάφειας ή της σχετικότητας μιας συγκεκριμένης διαδικτυακής υπηρεσίας (web service) με βάση τις προτιμήσεις του πελάτη (client) και των μονάδων μέτρησης της ποιότητας υπηρεσίας (QoS metrics) τα οποία υπολογίζονται αυτόνομα και ανεξάρτητα από τον πάροχο της υπηρεσίας.

Στο [16] γίνεται μια επέκταση στην Αρχιτεκτονική Ανοιχτών Υπηρεσιών Πλέγματος (Open Grid Services) για χαρακτηριστικά ποιότητας υπηρεσιών. Η ικανοποίηση του απαιτούμενου QoS συχνά απαιτεί μηχανισμούς όπως προηγμένη δέσμευση πόρων ή δέσμευση πόρων την στιγμή που ζητούνται (on-demand) που διαφοροποιούνται σε είδος και υλοποίηση και πρέπει να ελέγχονται και να γίνεται η διαχείριση τους (monitor) ανεξάρτητα το ένα από το άλλο. Προτείνεται η αρχιτεκτονική GARA. Η GARA βιβλιοθήκη παρέχει ένα περιορισμένο σχήμα

αναπαράστασης για κωδικοποίηση των ιδιοτήτων των πόρων και του σχετικού ελέγχου των SLAs. Δίνεται έμφαση στο επίπεδο/στρώμα εφαρμογών όπου μια δεδομένη υπηρεσία μπορεί να δείχνει ποια χαρακτηριστικά Ποιότητας υπηρεσιών QoS μπορεί να προσφέρει ή όπου μια υπηρεσία μπορεί να ψάξει για άλλες υπηρεσίες με βάση συγκεκριμένες ιδιότητες QoS. Αυτό που ουσιαστικά γίνεται είναι επέκταση της wsdl περιγραφής της υπηρεσίας. Τα χαρακτηριστικά ποιότητας υπηρεσιών που μπορεί να υποστηρίξει μία υπηρεσία καθώς και τα ελάχιστα απαιτούμενα όρια στα επίπεδα της ποιότητας υπηρεσιών ενσωματώνονται στην περιγραφή της διεπαφής της υπηρεσίας (wsdl description) και στην συνέχεια η υπηρεσία δημοσιοποιείται στο UDDI (Universal Description, Discovery and Integration). Στην συνέχεια γίνεται επέκταση και στο UDDI, προσθέτοντας μια νέα κατηγορία κριτηρίων αναζήτησης με βάση το επίπεδο ποιότητας υπηρεσιών στα ήδη υπάρχοντα κριτήρια του UDD, οπότε για την αναζήτηση χρησιμοποιούνται οι ήδη υπάρχοντες μηχανισμοί του UDDI. Επίσης όπως αναφέρεται και στο [17] δεν υποστηρίζεται η δυναμική καταχώρηση και ανακάλυψη πόρων με βάση το διαχειριστικό περιεχόμενο τους.

Στην εργασία αυτή υλοποιείται η ανακάλυψη υπηρεσιών και πόρων σε κατανεμημένο περιβάλλον με βάση το επίπεδο της ποιότητας των υπηρεσιών, μέσω της υλοποίησης ενός registry το οποίο θα επιτρέπει την δυναμική καταχώρηση οντοτήτων και αντικειμένων που θα αντιπροσωπεύουν τους πόρους και μέσω κατάλληλων ερωτημάτων θα μπορούν να ανακτούνται. Τα ερωτήματα αυτά θα αναφέρονται σε κριτήρια ποιότητας που επιθυμεί ο χρήστης να ικανοποιούν οι πόροι τους οποίους αναζητά. Στο κεφάλαιο 4 περιγράφεται η υλοποίηση αυτή.

Στην συνέχεια γίνεται μία ανάλυση της αρχιτεκτονικής του GRIA, το οποίο χρησιμοποιήθηκε για την δημιουργία των πόρων που χρησιμοποιήθηκαν για την υλοποίηση της αναζήτησης των υπηρεσιών.

ΚΕΦΑΛΑΙΟ 2

2. Πλατφόρμα GRIA

2.1. Εισαγωγή

Το GRIA μεσισμικό Πλέγματος (Grid middleware) βασίζεται σε διαδικτυακές υπηρεσίες (Web Services) και σχεδιάστηκε για να ικανοποιήσει τις ανάγκες της βιομηχανίας για ασφάλεια και για υπηρεσίες που παρέχονται από μία επιχείρηση προς μία άλλη επιχείρηση (business-to-business B2B υπηρεσίες) για προμήθεια και διάφορες άλλες επιχειρησιακές λειτουργίες, διαμέσου οργανισμών με έναν ασφαλή, διαδραστικό και ευέλικτο τρόπο. Παρέχει καλά ορισμένα B2B μοντέλα για συμφωνίες λογαριασμών και ποιότητας υπηρεσίας (QoS) και αντιπροσωπεία χωρίς πληρεξούσιους (proxy-free) για την υποστήριξη της διαχείρισης των λογαριασμών και των ομοσπονδιών υπηρεσιών.
Το GRIAv3 λογισμικό χρησιμοποιείται ήδη στις επιχειρήσεις.

2.2. Αρχιτεκτονική GRIA

Η 3η έκδοση του Gria συστήματος βασίζεται στην τεχνολογία του Apache Axis [19] και χρησιμοποιεί μόνο στοιχεία των διαδικτυακών υπηρεσιών που επιτρέπονται στο WS-I Basic Security (link-ref) Profile 1.0. Το μεσισμικό έχει τις παρακάτω πέντε βασικές λειτουργίες:

1) πάροχοι υπηρεσιών και καταναλωτές-χρήστες υπηρεσιών
2) ασφάλεια
3) διαχείριση πόρων
4) επεξεργασία δεδομένων
5) API για την πλευρά του client

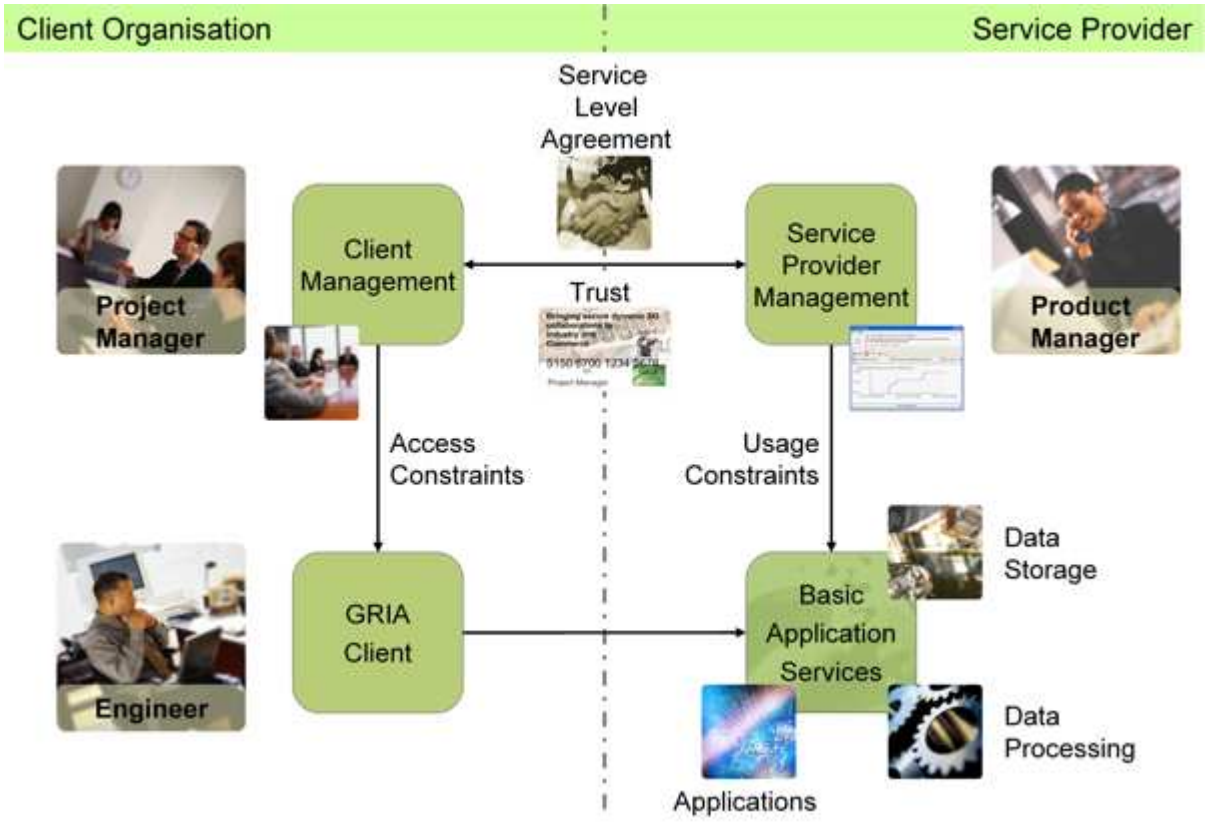
1) Το GRIA διαχωρίζει τον πάροχο από τον χρήστη. Ένας πάροχος υπηρεσιών είναι μία νόμιμη οντότητα (για παράδειγμα μία επιχείρηση) που παρέχει ένα σύνολο υπηρεσιών για πρόσβαση σε αποθηκευμένα δεδομένα και δυνατότητες επεξεργασίας αυτών. Ο χρήστης υπηρεσιών είναι μία νόμιμη οντότητα που χρησιμοποιεί αυτές τις δυνατότητες και λειτουργεί μέσω ενός πελάτη (client) υπηρεσιών που ελέγχεται από έναν αντιπρόσωπο (agent).
Η αλληλεπίδραση μεταξύ χρήστη και παρόχου αποτελεί μέρος μιας καλά καθορισμένης επιχειρησιακής διαδικασίας η οποία καθορίζει τα ανεξάρτητα βήματα

τα οποία απαιτούνται για να συμμορφωθεί ο χρήστης με μία καθορισμένη επιχειρησιακή ροή εργασίας.

2) Το GRIA υποστηρίζει τα παρακάτω στοιχεία ασφάλειας:

Ασφαλής μεταφορά: Το GRIA χρησιμοποιεί HTTPS ως πρωτόκολλο μεταφοράς για να παρέχει αξιοπιστία στα μηνύματα με ενεργοποίηση πιστοποίησης τόσο του πελάτη όσο και της υπηρεσίας.
Υπογραφές/Πιστοποιήσεις μηνυμάτων παρέχονται από WS-Security επικεφαλίδες επιτρέποντας έτσι να γίνεται έλεγχος στα μηνύματα για την ακεραιότητα τους και για το κατά πόσο αξιόπιστοι είναι οι αποστολείς τους ανεξάρτητα από το πρωτόκολλο μεταφοράς.
Πιστοποίηση: Το GRIA παρέχει ένα υποσύστημα ελέγχου πρόσβασης σε μια διαδικασία που ονομάζεται PBAC (Process Based Access Control) στο οποίο οι υπηρεσίες μπορούν να έχουν πρόσβαση για να ελέγξουν αν η ζητούμενη ενέργεια επιτρέπεται στην καθορισμένη διαδικασία από τον καλούντα χρήστη ή να ανανεώσουν την πολιτική για μεμονομένες ενέργειες και χρήστες . Κάθε προσπάθεια από μη πιστοποιημένο client να έχει πρόσβαση σε μία GRIA Service θα πέσει στην χειραψία στο στρώμα SSL η οποία μειώνει τις επιπτώσεις οποιασδήποτε άρνησης υπηρεσίας επίθεσης σε κάθε assigned context. Η απαίτηση για υπογραφή κάθε μηνύματος παρέχει ασφάλεια σε επίπεδο μηνύματος. Παρέχονται δύο τύποι ελέγχου, οι πιστοποιήσεις και οι περιορισμοί. Οι πιστοποιήσεις καθορίζουν τι απαιτείται και από ποιον ενώ οι περιορισμοί καθορίζουν ενέργειες που δεν επιτρέπονται και σε ποιον δεν επιτρέπονται.
3) Υπηρεσίες για την δημιουργία λογαριασμών και για την διαχείριση των πόρων. Αυτές οι υπηρεσίες υλοποιούν την υποστήριξη για τις εμπορικές και επιχειρησιακές διαδικασίες.
4) Υπηρεσίες που παρέχουν αποθήκευση δεδομένων και επεξεργασία δεδομένων. Αυτές οι υπηρεσίες υποστηρίζουν τις διαδικασίες στο επίπεδο των εφαρμογών του GRIA.
5) Αντικειμενοστραφές Java API για την υλοποίηση της πλευράς του πελάτη.
6) Το GRIA εισάγει νέα τεχνική για την διαχείριση εικονικών οργανισμών (VOs) αλλά δεν θα γίνει περαιτέρω ανάλυση καθώς δεν αποτελεί αντικείμενο της παρούσας εργασίας.

Το μεσισμικό πλέγματος GRIA όπως φαίνεται στην παρακάτω εικόνα αποτελείται από τέσσερα (4) αυτόνομα πακέτα λογισμικού τα οποία υλοποιούν τις λειτουργίες που περιγράφηκαν παραπάνω.



Εικόνα 1: Τα βασικά πακέτα υπηρεσιών του GRIA

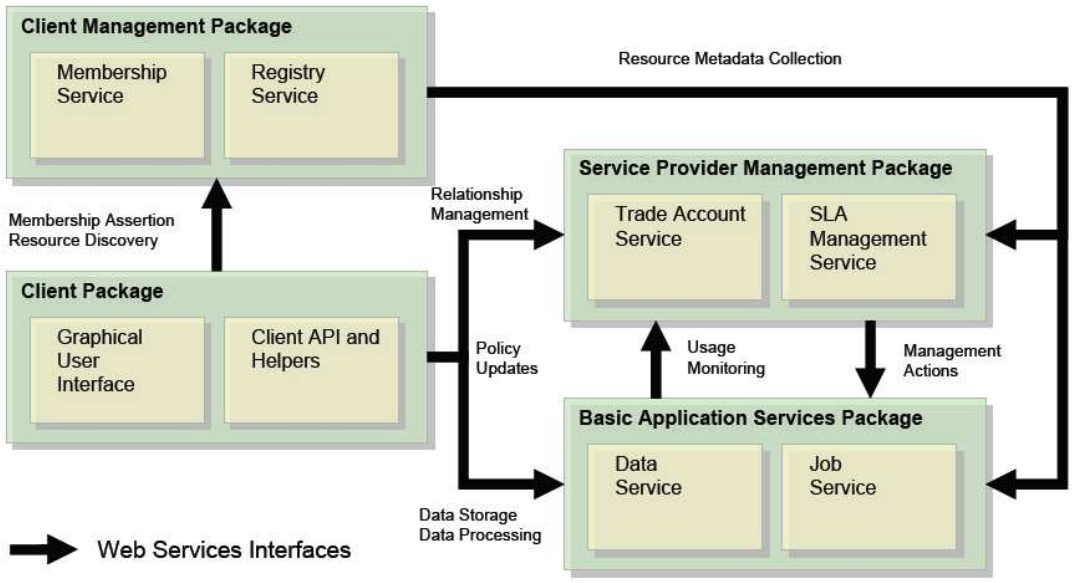
- Πακέτο Υπηρεσιών Βασικών Εφαρμογών (Basic Application Services package)επιτρέπει σε οργανισμούς που έχουν υποδομές ομάδων υπολογιστικών συστημάτων να παρέχουν αποθήκευση και επεξεργασία δεδομένων.

- Πακέτο Διαχείρισης Παρόχου Υπηρεσιών (Service Provider Management package) παρέχει υποστήριξη για διαχείριση και χρέωση υπηρεσιών με χρήση SLAs και βασίζεται σε ένα απλό πρωτόκολλο διαχείρισης για GRIA και υπηρεσίες εφαρμογών 3ης οντότητας.

- Πακέτο Πελάτη-Χρήστη (Client package) επιτρέπει την πρόσβαση των χρηστών σε υπηρεσίες διαχείρισης και εφαρμογών μέσω εφαρμογών υπολογιστή και περιλαμβάνει ένα client API toolkit για αναβάθμιση των εφαρμογών πελάτη.
- Πακέτο Διαχείρισης Πελάτη (Client Management package) παρέχει υποστήριξη για διαχείριση χρηστών υπηρεσιών σε επίπεδο ενός οργανισμού με κεντρικό έλεγχο και διαχείριση της παροχής και της μεταχείρισης των υπηρεσιών.

Παρέχεται επίσης το πακέτο Service Developers Toolkit το οποίο επιτρέπει την ενσωμάτωση των δικών μας υπηρεσιών εφαρμογών με τις δυνατότητες ασφάλειας και διαχείρισης του GRIA.

Στο παρακάτω σχήμα φαίνονται οι υπηρεσίες που παρέχονται στο GRIA και οι αλληλεπιδράσεις μεταξύ τους.



Εικόνα 2: Οι υπηρεσίες του GRIA και οι διασυνδέσεις μεταξύ τους

Παρατηρούμε στο παραπάνω σχήμα ότι η υπηρεσία διαχείρισης SLA διαχειρίζεται τις υπηρεσίες δεδομένων και επεξεργασίας μέσω των αναφορών χρησιμοποιήσης που της στέλνουν οι υπηρεσίες αυτές. Ο πελάτης μέσω του πακέτου Πελάτη του GRIA μπορεί να χρησιμοποιεί αυτές τις υπηρεσίες αλλά και να δημιουργεί λογαριασμούς και να συμφωνεί SLAs

2.3. Πακέτο διαχείρισης παρόχου υπηρεσιών (Service Provider Management Package)

Το πακέτο Διαχείρισης Παρόχου Υπηρεσιών επιτρέπει σε έναν πάροχο υπηρεσιών να απαιτεί SLAs ή ακόμα και να χρεώνει για την χρήση των υπηρεσιών του, μέσω ενός απλού πρωτοκόλλου διαχείρισης. Το πρωτόκολλο αυτό μπορεί να χρησιμοποιηθεί με τις βασικές εφαρμογές του GRIA αλλά και με άλλες (NON-GRIA) υπηρεσίες που δεν παρέχονται από το GRIA αν αυτό απαιτηθεί. Στην συνέχεια αναλύονται οι υπηρεσίες από τις οποίες αποτελείται.

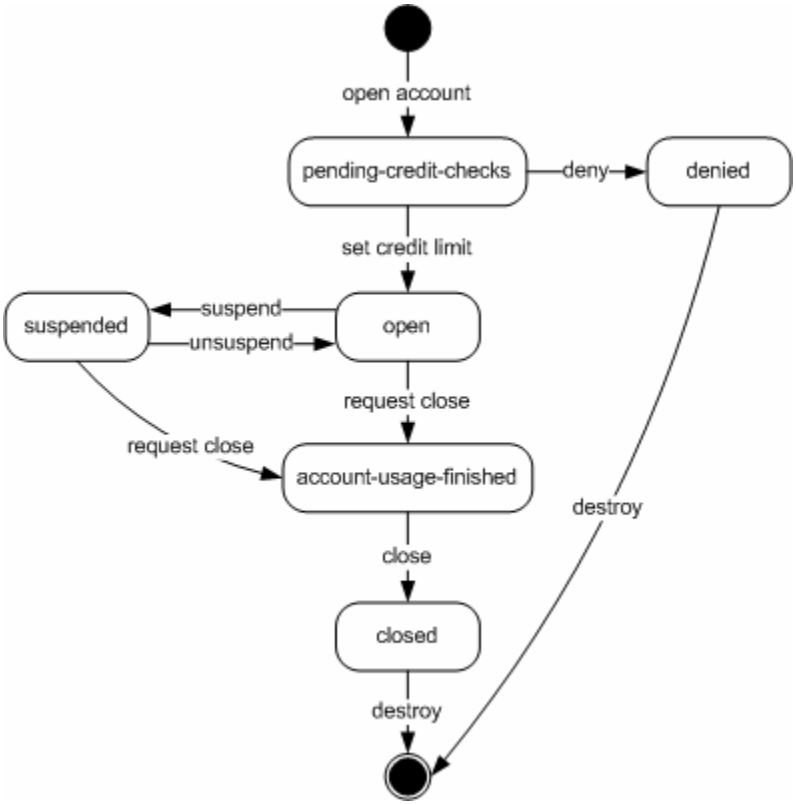
2.3.1. Υπηρεσία Λογαριασμών (Trade Account Service)

Αυτή η υπηρεσία υποστηρίζει την δημιουργία και την διαχείριση λογαριασμών-για εμπορικές συναλλαγές. Ο λογαριασμός αντιπροσωπεύει μια φερέγγυα σχέση μεταξύ του παροχου της υπηρεσίας και του πελάτη. Ο πελάτης είναι και ο κάτοχος του ποσού που αντιστοιχεί στον λογαριασμό και είναι υπεύθυνος για την πρόσβαση στον λογαριασμό του από τις υπηρεσίες που απαιτούν την χρήση λογαριασμού. Ο κάτοχος του λογαριασμού μπορεί να επιτρέψει και σε άλλους να έχουν πρόσβαση σε

υπηρεσίες μέσω του δικού του λογαριασμό και να χρεώνουν αυτόν ενώ μπορεί και να ελέγχει την χρησιμοποίηση του λογαριασμού του και τις χρεώσεις προς αυτόν ώστε να ελέγχει αν είναι λογικές και νόμιμες. Ο πάροχος της υπηρεσίας εμπιστεύεται τον κάτοχο του λογαριασμού ότι θα κάνει σωστή και υπεύθυνη διαχείριση πρόσβασης όποιος και να είναι ο ρόλος του είτε είναι δηλαδή ο εργαζόμενος που ηγήται σε ένα έργο σύμπραξης είτε είναι ένας εξωτερικός χρήστης που απλά πληρώνει για τις υπηρεσίες που του παρέχονται. Ο λογαριασμός αποτελεί επομένως την βάση εμπιστοσύνης μεταξύ του παρόχου και των απομακρυσμένων χρηστών.

Ο πάροχος αναθέτει ένα όριο πίστωσης (credit limit) σε κάθε καινούριο λογαριασμό που ανοίγει και καταγράφει τις χρεώσεις και τις πληρωμές που έχουν γίνει σε αυτόν. Η υπηρεσία λογαριασμών κρατάει στοιχεία για τις υποχρεώσεις του κάτοχου του λογαριασμού, όπως είναι οι εκρεμμότητες πληρωμών και παρέχει έναν μηχανισμό για τον έλεγχο της πίστωσης. Για παράδειγμα, καθορίζει αν είναι απαραίτητο να καλυφθεί κάποια χρέωση με ποσό που υπερβαίνει το όριο πίστωσης. Σε περίπτωση που το ποσό που χρεώνεται ξεπερνά το όριο πίστωσης, η χρέωση θα συνεχίσει να υφίσταται και έτσι ο λογαριασμός συνεχίζει να καταγράφει το ποσό που οφείλεται στον πάροχο. Η βασική ιδέα είναι ότι μια άλλη υπηρεσία θα πρέπει να ελέγχει την πίστωση πριν γίνει η παροχή μιας υπηρεσίας σε έναν χρήστη και αν ο έλεγχος δείξει ότι το όριο πίστωσης είναι εντάξει μπορεί μετά να κάνει τις χρεώσεις για τις παρεχόμενες υπηρεσίες. Η υπηρεσία αυτή είναι η υπηρεσία διαχείρισης SLA η οποία αναλύεται παρακάτω.
Αξίζει να σημειωθεί ότι υπηρεσία λογαριασμών δεν αντικαθιστά την τραπεζική συναλλαγή. Απλά γίνεται μια καταγραφή των χρεώσεων για την χρήση των υπηρεσιών και των χρημάτων που οφείλει ο πελάτης στον πάροχο της υπηρεσίας που έχει χρησιμοποιήσει και των πληρωμών που έχει κάνει ο χρήστης. Δεν μπορεί να παρέχει πλήρης εγγύηση ότι έχουν γίνει αυτές οι πληρωμές, ούτε να διαχειριστεί τα χρήματα του πελάτη.

Στο παρακάτω διάγραμμα ροής παρουσιάζονται οι καταστάσεις ενός λογαριασμού καθώς και οι μεταβάσεις μεταξύ τους.



Εικόνα 3: Οι δυνατές καταστάσεις ενός λογαριασμού και οι μεταβάσεις μεταξύ τους

- *Pending-credit-checks*: Όταν ένας χρήστης ζητήσει την δημιουργία ενός λογαριασμού, δίνει τα απαιτούμενα στοιχεία του. Ο λογαριασμός δημιουργείται αυτόματα αλλά παραμένει σε αυτήν την κατάσταση και ο πελάτης δεν μπορεί να τον χρησιμοποιήσει δηλαδή δεν μπορούν να καταγραφούν ακόμα χρεώσεις. Στην σελίδα της υπηρεσίας λογαριασμών, ο λογαριασμός βρίσκεται στην ενότητα “Accounts awaiting credit checks”. Ο διαχειριστής της υπηρεσίας λογαριασμών θα αποφασίσει αν θα επιτρέψει ή όχι την δημιουργία του λογαριασμού, ανάλογα με τους ελέγχους για την αξιοπιστία του χρήστη. Αν τον απορρίψει, ο λογαριασμός περνάει στην κατάσταση denied. Αν τον δεχτεί πρέπει να ορίσει και το όριο πίστωσης ανάλογα με τον βαθμό εμπιστοσύνης που θέλει να ορίσει στην σχέση του με τον χρήστη και τότε ο λογαριασμός περνάει στην κατάσταση open.
- *Denied*: Απορρίπτεται η αίτηση του χρήστη για άνοιγμα νέου λογαριασμού. Ο λογαριασμός μπορεί να σβηστεί από την σελίδα της υπηρεσίας λογαριασμών από τον διαχειριστή (destroy).
- *Open*: Η κατάσταση στην οποία η αίτηση για άνοιγμα νέου λογαριασμού έχει γίνει αποδεκτή και ο λογαριασμός μπορεί πλέον να χρησιμοποιηθεί. Μπορούν να καταγραφούν χρεώσεις και πληρωμές. Από αυτήν την κατάσταση ο λογαριασμός μπορεί να κάνει μετάβαση στην κατάσταση suspended αν ο διαχειριστής θελήσει να παγώσει για κάποιο διάστημα τον λογαριασμό ή στην κατάσταση closed αν ο διαχειριστής ή ο κάτοχός του θελήσουν να τον κλείσουν.

- *Suspended*: Σε αυτήν την κατάσταση ο λογαριασμός δεν μπορεί να χρεωθεί. Ο διαχειριστής μπορεί να τον να τον ανοίξει πάλι ή να τον κλείσει. Μπορεί να τον κλείσει και ο κάτοχος του.
- *Account –usage –finished*: Σε αυτήν την κατάσταση μπορούν ακόμα να καταγράφονται χρεώσεις και πληρωμές στον λογαριασμό. Ο διαχειριστής θα πρέπει να επιβεβαιώσει ότι έχουν καταγραφεί όλες οι χρεώσεις από την χρήση άλλων υπηρεσιών και να στείλει μία τελική χρέωση στον χρήστη. Όταν λάβει και την τελική πληρωμή θα πρέπει να την καταγράψει στον λογαριασμό και το υπόλοιπό του λογαριασμού να μηδενιστεί. Μετά μπορεί να τον κλείσει.
- *Closed*: Σε αυτήν την κατάσταση δεν μπορούν να καταγραφούν χρεώσεις και πληρωμές. Τα δεδομένα του λογαριασμού παραμένουν στο σύστημα μέχρι ο διαχειριστής να τον καταστρέψει οπότε και όλες οι καταχωρήσεις για τον λογαριασμό σβήνονται από το σύστημα.

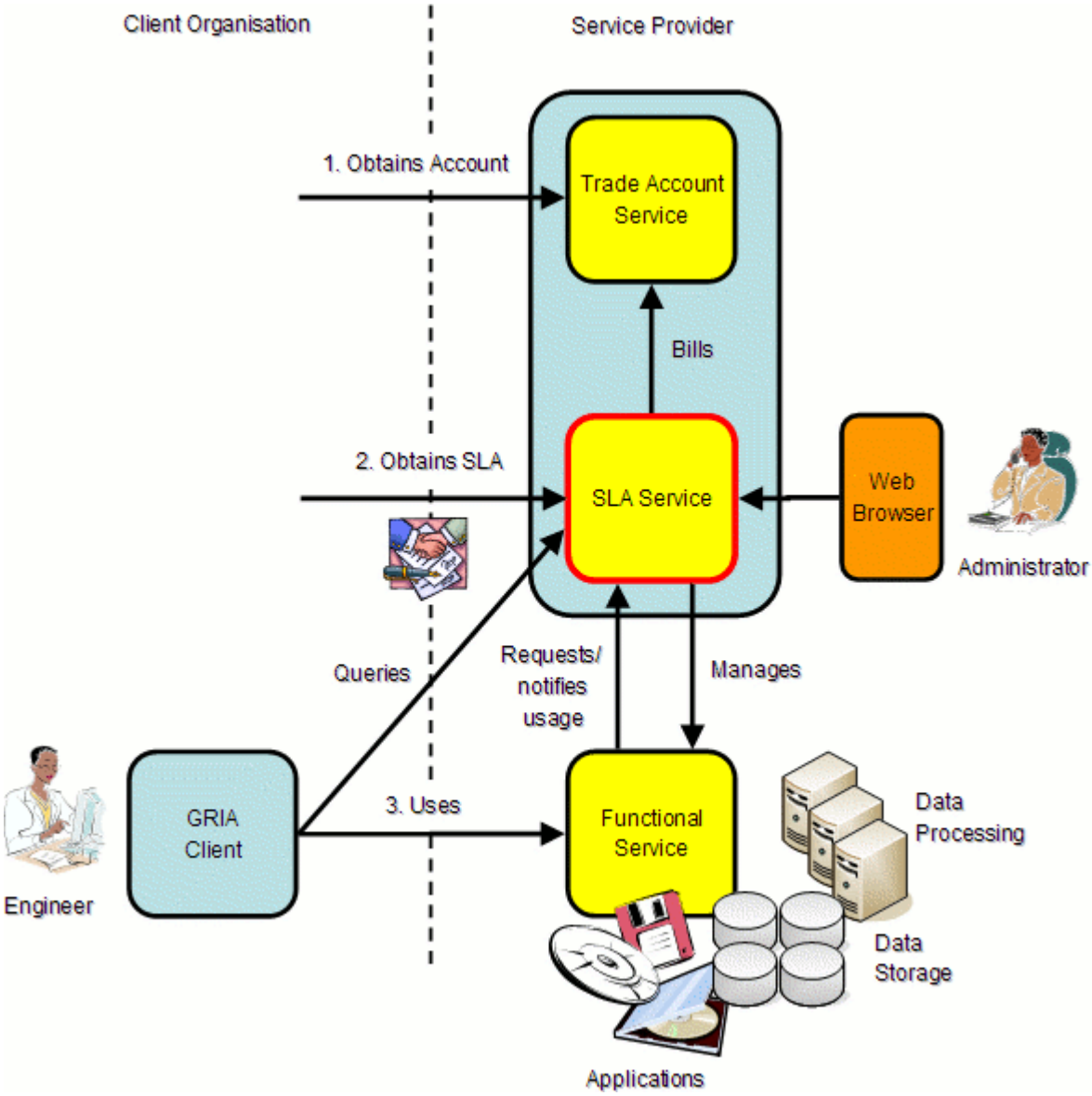
Μόνο κάποιος με τον ρόλο ‘account-service-admin’ μπορεί να κάνει καταγραφές των πληρωμών και αυτό καθορίζεται μόνο από τον διαχειριστή οποίος μπορεί να αναθέσει σε κάποιον άλλο τον ρόλο αυτόν αν τον εμπιστεύεται, δηλαδή αν εμπιστεύεται το πιστοποιητικό του.

2.3.2. Υπηρεσία Διαχείρισης SLA (SLA Management Service)

Εισαγωγή στην υπηρεσία SLA

Η υπηρεσία αυτή παρέχει στον πάροχο μιας υπηρεσίας την δυνατότητα να προσδιορίσει τους διαθέσιμους πόρους του, να αναθέσει μέρη των πόρων του σε πελάτες μέσω της χρήσης Service Level Agreements (SLAs) και να χρεώσει για την χρησιμοποίηση αυτών των πόρων ανάλογα με τους όρους χρέωσης που στα αντίστοιχα SLAs.

Η αλληλεπίδραση της SLA υπηρεσίας με τις άλλες υπηρεσίες φαίνεται στο παρακάτω διάγραμμα:



Εικόνα 4: Χρήση της υπηρεσίας SLA

Επεξήγηση σχήματος:

Ο πελάτης, ο οποίος μπορεί να είναι είτε απλός εργαζόμενος είτε ο διαχειριστής ενός έργου (project manager) στον οργανισμό

- αποκτά έναν λογαριασμό
- συμφωνεί ένα SLA
- χρησιμοποιεί μία υπηρεσία την οποία διαχειρίζεται το SLA της υπηρεσίας SLA, η οποία με τη σειρά της θα χρεώσει τον λογαριασμό του πελάτη για κάθε χρησιμοποίηση της υπηρεσίας.

Η SLA υπηρεσία διαχειρίζεται λειτουργίες, όπως για παράδειγμα αποθήκευση δεδομένων, εργασίες, βάσεις δεδομένων, λειτουργικές υπηρεσίες και σε κάποιο μικρότερο βαθμό η υπηρεσία λογαριασμών διαχειρίζεται την υπηρεσία SLA.

Χρησιμοποιείται ο όρος λειτουργική υπηρεσία για να χαρακτηρίσει την υπηρεσία που επιτελεί κάποια λειτουργία και είναι αυτή την οποία μπορεί να διαχειριστεί η υπηρεσία SLA.

Ο διαχειριστής της SLA Service γράφει και δημοσιεύει τα "SLA templates". Τα SLA templates περιέχουν όποιους πόρους προσφέρει ο πάροχος της λειτουργικής υπηρεσίας. Χρησιμοποιώντας τα SLA templates ο πάροχος μπορεί να προσφέρει διάφορα επίπεδα υπηρεσιών σε διαφορετικές τιμές. Υπάρχουν για παράδειγμα τα πακέτα 'gold', 'silver' και 'bronze'. Το Σύστημα ελέγχου πρόσβασης του GRIA (GRIA access control system) μπορεί να ελέγξει ποιοι πελάτες έχουν πρόσβαση σε κάθε SLA template. Έτσι μπορούν να προσφερθούν ειδικά επίπεδα υπηρεσιών σε συγκεκριμένους πελάτες όπως για παράδειγμα ένα SLA template που προσφέρει ένα συγκεκριμένο επίπεδο υπηρεσιών αλλά σε ειδικά χαμηλότερες τιμές και είναι ορατό σε συγκεκριμένη κατηγορία πελατών.

Διαδικασία συμφωνίας SLA

Για να χρησιμοποιήσει ένας πελάτης μια υπηρεσία η οποία είναι διαχειριζόμενη από την SLA υπηρεσία, πρέπει να συμφωνήσει ένα SLA με την αντίστοιχη SLA υπηρεσία.
Ο πελάτης βλέπει τα SLA templates που προσφέρονται από τους παρόχους μέσω της υπηρεσίας SLA.
Διαλέγει το SLA template που ικανοποιεί τις απαιτήσεις του και το προτείνει στον πάροχο της λειτουργικής υπηρεσίας. Αν ο πάροχος συμφωνήσει με την πρόταση και έχει αρκετούς πόρους για να ικανοποιήσει την συμφωνία τότε η συμφωνία πραγματοποιείται και το SLA δημιουργείται.
Η απόφαση για την πραγματοποίηση της συμφωνίας γίνεται αυτόματα από την υπηρεσία SLA ανάλογα με την διαθεσιμότητα των πόρων και λαμβάνοντας υπόψιν και τις απαιτήσεις των άλλων SLA που έχουν ήδη συμφωνηθεί.
Το SLA μεταξύ του πελάτη και του παρόχου περιγράφει τις χρεώσεις για την χρήση των πόρων που παρέχει η λειτουργική υπηρεσία και οι περιορισμοί στους οποίους υπόκειται ο χρήστης για την δημιουργία νέων πόρων.
Όταν ο πελάτης χρησιμοποιεί την λειτουργική υπηρεσία παραθέτει το μοναδικό αναγνωριστικό (ID) του SLA που έχει συμφωνήσει στην αίτηση του για την υπηρεσία αυτή. Μέσω αυτού του SLA ID η υπηρεσία SLA μπορεί να διαχειριστεί την λειτουργική υπηρεσία και την χρησιμοποίηση των πόρων της.

Πώς διαχειρίζεται η SLA Service την λειτουργική υπηρεσία:

- Η υπηρεσία ρωτά την SLA service πότε μπορεί να αρχίσει να εκτελεί κάποια ενέργεια η οποία μπορεί να απαιτεί μεγάλη χρησιμοποίηση πόρων.
- Ειδοποιεί την SLA Service για την χρησιμοποίηση πόρων έγκαιρα.
- Υπακούει στις οδηγίες της SLA Service για καταστροφή λειτουργιών.

Παράδειγμα:
Όταν ένας πελάτης ζητήσει έναν νέο πόρο από την υπηρεσία, η υπηρεσία ζητά από την υπηρεσία SLA να ελέγξει αν ο πελάτης δικαιούται και άλλο πόρο σύμφωνα με το SLA που έχει συμφωνήσει. Αν η SLA Service αποφασίσει ότι ο πελάτης δεν δικαιούται άλλο πόρο τότε η υπηρεσία πρέπει να μην συνεχίσει στην

παροχή και άλλου πόρου στον πελάτη. Όταν ο πελάτης χρησιμοποιήσει τον πόρο που του έχει διατεθεί, η υπηρεσία στην οποία ανήκει ο πόρος ειδοποιεί την SLA Service για την χρησιμοποίηση του πόρου. Αν η λειτουργική υπηρεσία είναι υπηρεσία δεδομένων για διαχείριση δεδομένων, ο πόρος που παρέχεται από την υπηρεσία είναι ένα data stager δηλαδή ένας χώρος για αποθήκευση ενός μόνο αρχείου , και αν ο πελάτης αποθηκεύσει ένα αρχείο στον χώρο αυτό η υπηρεσία δεδομένων θα ειδοποιήσει την SLA Service, από την οποία διαχειρίζεται, για τον χώρο στο αποθηκευτικό μέσο που θα δεσμεύσει το αρχείο και για τον ρυθμό μεταφοράς του αρχείου. Αν η SLA Service διαπιστώσει ότι ο πελάτης έχει υπερβεί κάποιον περιορισμό που είχε οριστεί στο SLA που είχε συμφωνήσει τότε πραγματοποιεί ενέργειες για να επαναφέρει την χρησιμοποίηση των πόρων στα πλαίσια που ορίζουν οι περιορισμοί του, δίνοντας οδηγία στην λειτουργική υπηρεσία να σταματήσει κάποιο αριθμό λειτουργιών του χρήστη, όπως για παράδειγμα να σταματήσει την δημιουργία νέων πόρων.

Οι υπηρεσίες καταγράφουν την χρησιμοποίηση των πόρων που κάνει ο πελάτης στους πόρους τους και την δηλώνουν στην SLA Service χρησιμοποιώντας μονάδες μέτρησης που ονομάζονται μετρικές (metrics). Οι μετρικές υποδεικνύουν στην SLA Service τις ποσότητες που πρέπει να μετρήσει. Μετρικές είναι για παράδειγμα ο χώρος σε κάποιο αποθηκευτικό μέσο, η ταχύτητα μεταφοράς δεδομένων, ο ρυθμός μετάδοσης δεδομένων, η CPU, εργασίες σε κάποια βάση δεδομένων κτλ. Κάθε μετρική περιγράφεται από ένα μοναδικό URI. Η SLA Service δεν καταλαβαίνει τι σημαίνει και τι αντιπροσωπεύει κάθε μετρική , απλά καταγράφει την χρησιμοποίηση του και την συγκρίνει με τους περιορισμούς που έχουν οριστεί στο SLA για την χρήση των συγκεκριμένων πόρων. Με αυτόν τον τρόπο η SLA Service μπορεί να διαχειρίζεται νέες υπηρεσίες που μπορεί να χρησιμοποιούν καινούριες μετρικές οι οποίες να περιγράφουν συγκεκριμένα κάποια λειτουργία ή κάποιο πόρο, χωρίς να αλλάζει κάτι στην λειτουργία της.
Παρακάτω εξηγούνται με λεπτομέρεια οι μετρικές καθώς και ο τρόπος που ορίζονται στην SLA Service του GRIA.

<u>Διαχείρηση της SLA υπηρεσίας από την υπηρεσία λογαριασμών.</u>

1) Οταν ο πελάτης προτείνει ένα νέο SLA η SLA Service μαζί με την υπηρεσία λογαριασμών ελέγχει αν η πίστωση του λογαριασμού του είναι ‘καλή’ άρα αν υπάρχει καλή σχέση εμπιστοσύνης με τον πελάτη.
2) Στο τέλος κάθε περιόδου χρέωσης η SLA Service καταθέτει ένα συνολικό ποσό χρέωσης για όλη την περίοδο χρήσης του πόρου στον λογαριασμό του πελάτη.
3) Αν κατά την χρέωση του λογαριασμού διαπιστώσει η SLA Service ότι ο λογαριασμός έχει διακοπεί προσωρινά (κατάσταση suspended) ή έχει κλείσει (κατάσταση closed) τότε διακόπτεται και το ίδιο το SLA.

    - Ο διαχειριστής (service manager) ρυθμίζει την SLA Service ώστε να εμπιστεύεται μία ή περισσότερες υπηρεσίες λογαριασμών.
    - Ο διαχειριστής καθορίζει την χωριτικότητα και τις ικανότητες των πόρων, μέσα στην SLA Service.

- Ο διαχειριστής δημιουργεί ένα ή περισσότερα SLA templates ανάλογα με το τι θέλει να προσφέρει η λειτουργική υπηρεσία στους πελάτες.
- Ο διαχειριστής ορίζει ποιοι πελάτες μπορούν να έχουν πρόσβαση σε κάθε SLA template.
- Ο διαχειριστής δημοσιεύει αυτά τα SLA templates
- Ο πελάτης αφού έχει δημιουργήσει έναν λογαριασμό σε κάποια από τις υπηρεσίες λογαριασμών τις οποίες εμπιστεύεται η υπηρεσία SLA, βλέπει την λίστα με όλα τα διαθέσιμα SLA templates.
- Ο πελάτης εξετάζει τα SLA templates και αν θέλει κάποιο το προτείνει στην SLA Service.
- Η SLA Service εξετάζει την πρόταση και ελέγχει τον λογαριασμό του πελάτη. Αν ο λογαριασμός έχει αρκετό υπόλοιπο για την δημιουργία νέων πόρων, τότε η πρόταση γίνεται αποδεκτή και δημιουργείται το SLA μεταξύ του παρόχου και του πελάτη.
- Ο πελάτης χρησιμοποιεί τις υπηρεσίες που του επιτρέπει το SLA του. Η χρησιμοποίηση των πόρων της υπηρεσίας ελέγχονται, περιορίζονται και χρεώνονται από την SLA Service.
- Ο διαχειριστής ελέγχει την χρησιμοποίηση των πόρων.
- Ο διαχειριστής μπορεί να διευθύνει την λειτουργική υπηρεσία.

Τι είναι οι μετρικές και πώς χρησιμοποιούνται:

Η υπηρεσία διαχείρισης SLA του GRIA έχει σχεδιαστεί έτσι ώστε να είναι πολύ ευέλικτη. Χρησιμοποίει πληροφορίες για χρησιμοποίηση από τις λειτουργικές υπηρεσίες όπως π.χ. υπηρεσίες εργασιών (job services) καταγράφει , περιορίζει ή/και χρεώνει την χρησιμοποίηση αυτών.

Οι μετρικές όπως αναφέρθηκε είναι μετρήσιμες ποσότητες που δείχνουν την χρησιμοποίηση ενός πόρου. Μέσω των μετρικών δηλώνεται η χρησιμοποίηση των πόρων στην SLA Service από τις λειτουργικές υπηρεσίες που τους παρέχουν, όπως για παράδειγμα η χρησιμοποίηση της CPU. Οι μετρικές παριστάνονται με URIs τα οποία είναι μοναδικά. Η SLA Service όπως αναφέραμε και παραπάνω δεν αντιλαμβάνεται την σημασία αυτών των URI και απλά δηλώνει την χρησιμοποίηση τους και ενεργεί ανάλογα με τους όρους του SLA που έχει συμφωνηθεί. Οι μετρικές χρησιμοποιούνται μέσα στο XML κείμενο του SLA στα στοιχεία <constraint> ώστε να προσδιορίζεται σε ποια μετρική δηλαδή σε ποιο πόρο αναφέρεται ο περιορισμός που θέλουμε να ισχύει και στο <pricing term> ώστε να προσδιορίζεται ο πόρος για την χρήση του οποίου θα γίνει η αντίστοιχη χρέωση. Η μετρική χρησιμοποιείται επίσης και στο αρχείο xml που προσδιορίζει την χωρητικότητα και τις δυνατότητες σε πόρους που παρέχει μία υπηρεσία.
Η μέτρηση της χρήσης των μετρικών γίνεται με δύο τρόπους:

- **Στιγμιαίες μετρήσεις των μετρικών (instantaneous measurements)**. Μετρήσεις της χρήσης του πόρου μία συγκεκριμένη στιγμή. Μετράει τον ρυθμό χρησιμοποίησης.
- **Μέτρηση της χρήσης της μετρικής καθ'όλη την διάρκεια χρήσης της υπηρεσίας (cumulative usage):** Είναι η άθροιση των στιγμιαίων μετρήσεων σε όλο το διάστημα που έχει χρησιμοποιηθεί η υπηρεσία έως εκείνη τη στιγμή. Αποτελεί την συνολική χρησιμοποίηση του πόρου.

Για κάποιες μετρικές όπως για παράδειγμα η μεταφορά δεδομένων, η στιγμιαία μέτρηση είναι ο ρυθμός μεταφοράς και μετριέται σε bytes/sec ενώ η συνολική χρησιμοποίηση μετριέται σε bytes χωρίς διάσταση χρόνου. Για άλλες μετρικές όπως η CPU η στιγμιαία μέτρηση είναι ο αριθμός των CPU που χρησιμοποιούνται αυτήν την στιγμή π.χ. 3 CPUs και η συνολική χρήση έχει την διάσταση του χρόνου αριθμός CPU*συνολικός χρόνος χρήσης τους = CPU.seconds π.χ. 180 CPU.seconds. Η SLA Service μπορεί να μετατρέπει τον έναν τρόπο μέτρησης στον άλλον.

Παράδειγμα:

Αν ένα μία εφαρμογή (job) τρέχει σε μία CPU για 5 δευτερόλεπτα τότε η υπηρεσία SLA θα ενημερωθεί ότι η στιγμιαία μέτρηση της CPU χρήσης πήγε στο 1 στην αρχή και πέντε λεπτά αργότερα πήγε στο 0. Η SLA Service μπορεί να αναφέρει ότι έχουν χρησιμοποιηθεί 1 CPU*5 min*60 sec/min = 300 CPU.seconds.
Αν μία υπηρεσία αναφέρει ότι έχει χρησιμοποιήσει 120 μονάδες ενός πόρου σε χρονική περίοδο 1 λεπτού η SLA Service μπορεί να αναφέρει ότι η μέση στιγμιαία μέτρηση (ρυθμός χρήσης) ήταν 120 μονάδες/60 sec = 2 units/sec.

Ορισμός των metrics στην SLA υπηρεσία:

Στην GRIA SLA Service υπάρχει ένας επιμελητής μετρικών (metric editor) όπου μπορούμε να ορίσουμε μία νέα μετρική για έναν πόρο που μας ενδιαφέρει ή να τροποποιήσουμε κάποια από τις ήδη υπάρχουσες που έχει ανακαλύψει η υπηρεσία SLA. Στον επιμελητή μπορούμε να καθορίσουμε τι μέτρηση θέλουμε να γίνεται δηλαδή στιγμιαία ή αθροιστική, τον τύπο της μετρικής αν θα είναι πόρος, δηλαδή θα χρησιμοποιείται για την μέτρηση ενός πραγματικού πόρου όπως CPU ή δραστηριότητα, δηλαδή θα χρησιμοποιείται για να δείχνει την χρησιμοποίηση κάποιας λειτουργίας όπως για παράδειγμα η δημιουργία ενός data-stager, ο τύπος της μονάδας μέτρησης αν θα είναι Δεκαδικός ή Δυαδικός που δείχνει αν θα πρέπει να διαιρέσουμε με το 1000 ή με το 1024 για να γίνει μετάβαση σε μεγαλύτερη μονάδα μέτρηση όταν έχουμε φτάσει σε μεγάλη ποσότητα χρησιμοποίησης.
Η SLA υπηρεσία μπορεί να δει τις καινούργιες μετρικές κατά τον ορισμό της χωρητικότητας (capacity) ή όταν φορτώνονται καινούρια SLA templates τα οποία περιέχουν μη ορισμένες μετρικές ή από τις αναφορές χρησιμοποίησης που λαμβάνει από τις λειτουργικές υπηρεσίες.

Διαχείρηση της χωρητικότητας (Capacity Management)

Ο πάροχος μπορεί να καθορίσει την χωρητικότητα δηλαδή το μέγεθος ή την ποσότητα των πόρων που θα παρέχει έτσι ώστε να μπορεί να την κατανείμει στους πελάτες έτσι ώστε να τηρούνται οι δεσμεύσεις για τους πόρους που ορίστηκαν στο SLA. Όταν όλη η διαθέσιμη χωρητικότητα έχει ανατεθεί, πρέπει να συμφωνηθούν νέα SLAs. Αυτό έχει κυρίως επίδραση σε μετρικές όπως χώρος σε αποθηκευτικό μέσο (disc space) όπου ο πάροχος έχει περιορισμένη ποσότητα πόρων και πρέπει να εγγυηθεί στους χρήστες για την ποσότητα των πόρων που μπορεί να τους αναθέσει.

Αν δεν οριστεί η χωρητικότητα θα μπορούν να ανατεθούν άπειρα SLAs και ο πάροχος δεν θα μπορεί να ικανοποιήσει τις δεσμεύσεις του για ανάθεση πόρων.

Υπάρχουν δύο τρόποι για να οριστεί η χωρητικότητα:

1) Φόρτωση ενός XML αρχείου που θα καθορίζει την χωρητικότητα.
2) Μέσω του αντίστοιχου επιμελητή στην ιστοσελίδα της υπηρεσίας SLA.

SLA Templates

Το SLA template ελέγχει ποιες υπηρεσίες μπορεί να χρησιμοποιήσει ο πελάτης, την ποσότητα των πόρων που θα μπορούν να χρησιμοποιήσουν σε αυτήν την υπηρεσία, πόσο θα χρεώνονται για αυτήν την χρήση και πόσο συχνά.

Υπάρχουν δύο τρόποι για να οριστεί ένα SLA template:

1) Φορτώνοντας ένα xml αρχείο
2) Με τον editor που υπάρχει στην ιστοσελίδα της SLA Service.

Παραδείγματα από SLAs φαίνονται στα τελευταία κεφάλαια όπου περιγράφεται η υλοποίηση.

Έλεγχος λειτουργίας ενός SLA (Monitoring)

Η υπηρεσία διαχείρισης SLA συλλέγει πληροφορίες για την χρησιμοποίηση των πόρων από τις λειτουργικές υπηρεσίες μέσω των μετρικών. Σε συγκεκριμένα χρονικά διαστήματα η υπηρεσία SLA συγκεντρώνει πληροφορίες από κάθε υπηρεσία για τους πόρους που έχουν χρησιμοποιηθεί. Καταχωρεί την χρησιμοποίηση για την αντίστοιχη ενέργεια, σε αντιδιαστολή με το SLA στο οποίο ανήκει η ενέργεια αυτή και σε σχέση με την υπηρεσία ως σύνολο. (Μπορούν να εμφανιστούν δεδομένα για την χρησιμοποίηση για όλα τα metrics για δραστηριότητες, SLAs και την υπηρεσία.).

Διαχείρηση ενός SLA (Managing)

Ένα SLA έχει τέσσερις καταστάσεις:

- *Ενεργό (active)* : Είναι η κανονική κατάσταση που έχει το SLA όταν δημιουργείται. Σε αυτήν την κατάσταση μπορεί να έλθει ένα SLA αφότου έχει διακοπεί από τον διαχειριστή. Όταν το SLA είναι ενεργό η υπηρεσία SLA μπορεί να επεξεργάζεται αιτήσεις από λειτουργικές υπηρεσίες ώστε να ξεκινήσει καινούριες ενέργειες σύμφωνα με τους περιοριστικούς όρους του SLA για την χρησιμοποίηση.
- *Ανασταλμένο (suspended)*: Ένα SLA έρχεται σε αυτήν την κατάσταση αν η υπηρεσία SLA διαπιστώσει ότι δεν είναι εφικτή η χρέωση στην υπηρεσία λογαριασμών για το SLA. Αυτό θα μπορούσε να συμβεί στο τέλος μιας περιόδου χρέωσης αν ο λογαριασμός του SLA είναι στην κατάσταση αναστελμένο ή κλειστό ή όταν ο διαχειριστής επιλέξει να το διακόψει (πατώντας το κουμπί Suspend της web page της SLA service). Όταν το SLA έχει διακοπεί δεν μπορούν να χρησιμοποιηθούν πόροι.

- *Έτοιμο να κλείσει (c)*: Κάποιο SLA έρχεται σε αυτήν την κατάσταση όταν είτε ο πελάτης επιλέξει να κλείσει το SLA του ή όταν ο διαχειριστής πατήσει το κουμπί "Close" από την διαδικτυακή εφαρμογή. Σε αυτήν την κατάσταση όλες οι δραστηριότητες που τρέχουν εκείνη τη στιγμή θα καταστραφούν και τα δεδομένα θα χαθούν. Το SLA παραμένει σε αυτήν την κατάσταση μέχρι να τελειώσουν όλες οι ενέργειες και να έχει σταλεί και ο τελικός λογαριασμός. Δεν μπορούν πλέον να ξεκινήσουν νέες δραστηριότητες.
- *Κλειστό (closed)*: Σε αυτήν την κατάσταση έχει σταλεί επιτυχώς ο τελικός λογαριασμός στην υπηρεσία λογαριασμών και δεν υπάρχουν άλλες τρέχουσες δραστηριότητες στο SLA. Δεν μπορούν να ξεκινήσουν νέες δραστηριότητες σε αυτήν την φάση.

2.4. Πακέτο Βασικών Εφαρμογών (Basic Application Services Package)

Το πακέτο αυτό παρέχει τις βασικές υπηρεσίες για την διαχείριση εργασιών (job) και δεδομένων (data). Αποτελείται από δύο επιμέρους υπηρεσίες.

2.4.1. **Υπηρεσία Δεδομένων (Data Service)**

Οι πόροι τους οποίους διαχειρίζεται η υπηρεσία δεδομένων ονομάζονται "data stagers". Ένα data stager είναι ένας αποθηκευτικός χώρος που μπορεί να περιέχει ένα μόνο αρχείο. Έχει ένα μοναδικό χαρακτηριστικό (identifier) και ένα σύστημα ελέγχου πρόσβασης ώστε να καθορίζεται ποιος μπορεί να διαβάσει ή να γράψει δεδομένα.
Επιτρέπει σε απομακρυσμένους χρήστες να δημιουργούν νέα data stagers, να φορτώνουν (upload) και να κατεβάζουν (download) αρχεία δεδομένων προς και από τον πάροχο της υπηρεσίας αλλά και να μεταφέρουν δεδομένα μεταξύ data stagers αλλά και μεταξύ Υπηρεσιών Δεδομένων που παρέχονται από διαφορετικούς παρόχους. Υποστηρίζει επίσης την διαχείριση δικαιωμάτων και κανόνων πρόσβασης για ανάγνωση και γράψιμο που θα ισχύουν για άλλους χρήστες και παρόχους υπηρεσιών.

2.4.2. **Υπηρεσία Επεξεργασίας (Job Service)**

Η υπηρεσία αυτή επιτρέπει σε απομακρυσμένους χρήστες να αρχίσουν , να ελέγξουν και να σταματήσουν εργασίες επεξεργασίας, που εκτελούνται από τον πάροχο της υπηρεσίας. Η υπηρεσία επεξεργασίας θα φέρει τα δεδομένα εισόδου από κάποια τοπική υπηρεσία δεδομένων και θα αποθηκεύσει τα δεδομένα εξόδου πάλι σε κάποια τοπική υπηρεσία δεδομένων. Μπορεί να ρυθμιστεί έτσι ώστε να υποστηρίζει πολλαπλές εφαρμογές οι οποίες επιλέγονται από τον πάροχο της υπηρεσίας.
Οι πόροι που διαχειρίζεται η υπηρεσία αποτελούν εργασίες (jobs) δηλαδή συγκεκριμένες εφαρμογές. Οι χρήστες της υπηρεσίας μπορούν να δημιουργήσουν καινούριες εργασίες, να φορτώσουν κάποια δεδομένα εισόδου, να αρχίσουν μία εργασία, να ελέγξουν την εξέλιξη του και να κατεβάσουν τα αποτελέσματα . Κάθε είσοδος και έξοδος μιας εργασίας είναι ουσιαστικά ένα data stager το οποίο το διαχειρίζεται η τοπική υπηρεσία δεδομένων. Με τον χαρακτηρισμό τοπική εννοούμε

ότι παρέχεται από τον ίδιο πάροχο. Οι χρήστες μπορούν επίσης να εκτελέσουν εργασίες οι οποίες παίρνουν δεδομένα ή αποθηκεύουν τα δεδομένα τους σε data stagers που διαχειρίζονται από Υπηρεσίες Δεδομένων που παρέχονται από άλλους απομακρυσμένους παρόχους.

Οι υπηρεσίες μπορούν να ρυθμιστούν έτσι ώστε να απαιτούν την ύπαρξη κάποιου λογαριασμού από τον χρήστη και την συμφωνία κάποιου SLA από το οποίο θα διαχειρίζονται ή να είναι ελεύθερες προς χρήση . Στην εργασία αυτή έχει υλοποιηθεί η πρώτη επιλογή.

ΚΕΦΑΛΑΙΟ 3

3. Υλοποίηση

3.1. Εγκατάσταση των πακέτων του GRIA

Οι υπηρεσίες του GRIA παρέχονται σε war (Web ARchive) αρχεία. Η διαδικασία που ακολουθείται για την εγκατάσταση, σύνδεση και διαμόρφωση τους είναι ίδια για όλες τις υπηρεσίες που χρησιμοποιήθηκαν. Εδώ θα περιγραφεί η παραπάνω διαδικασία για την εγκατάσταση του πακέτου διαχείρισης παρόχου υπηρεσιών στον υπολογιστή argugrid-server.netmode.ntua.gr.

ΠΕΡΙΒΑΛΛΟΝ ΥΛΟΠΟΙΗΣΗΣ

Η υλοποίηση πραγματοποιήθηκε σε δίκτυο τριών υπολογιστών:

dimitra.netmode.ntua.gr
niobe.netmode.ntua.gr
argugrid-server.netmode.ntua.gr

Η εγκατάσταση του GRIA και των υπηρεσιών του πραγματοποιήθηκε με τον ίδιο τρόπο και για τους τρεις υπολογιστές.

ΠΡΟΑΠΑΙΤΟΥΜΕΝΟ ΛΟΓΙΣΜΙΚΟ

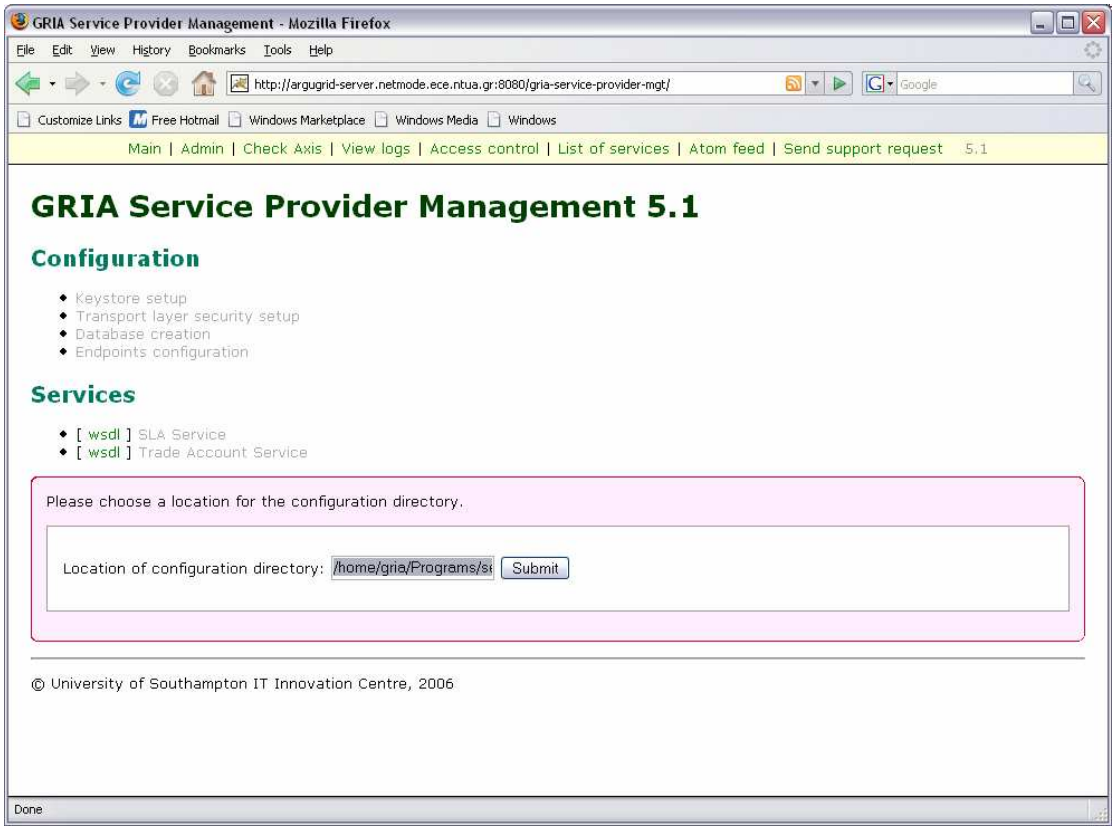
Java 1.5.0_12
Tomcat 5.5_25
Ελευθερώνουμε τις παρακάτω θύρες από τα firewalls:

- Tomcat, 8080 and και πρωτόκολλο μεταφοράς ορίζεται το TCP
- Tomcat (secure), 8443 και πρωτόκολλο μεταφοράς ορίζεται το TCP

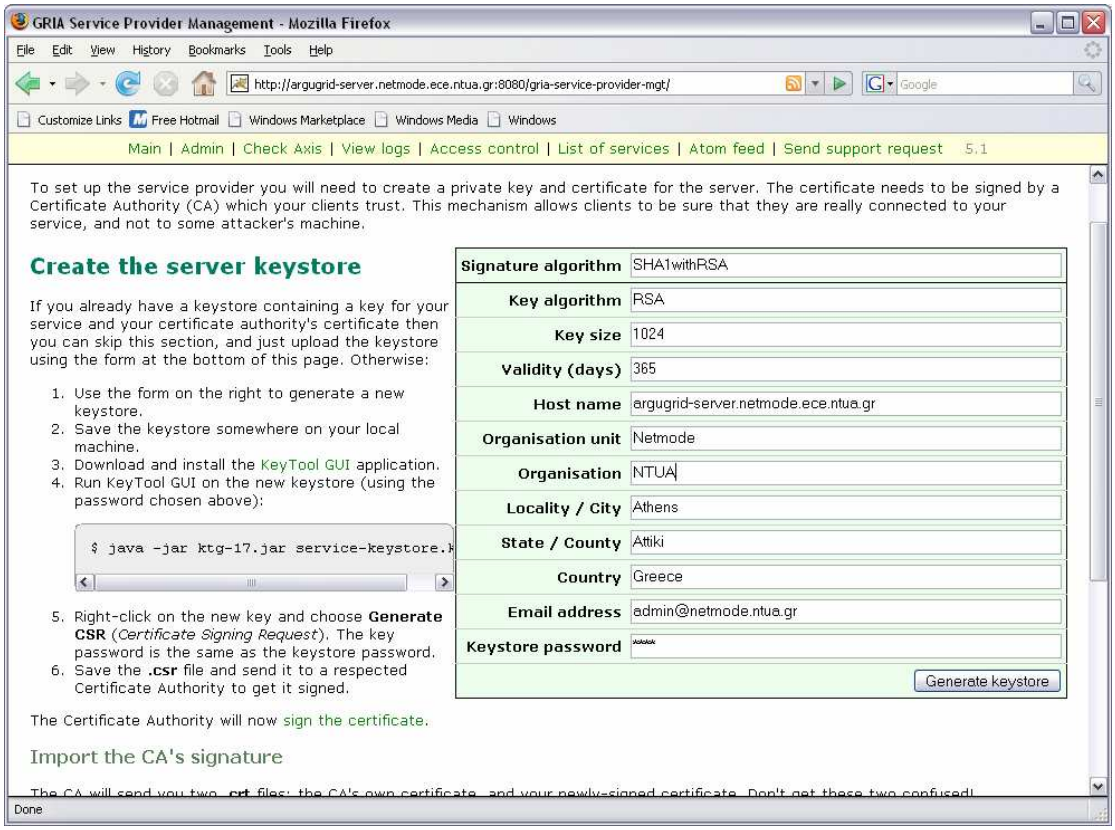
ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΔΙΑΔΙΚΑΣΙΑΣ

Αρχικά ανοίγουμε τον Tomcat και εισάγουμε το war αρχείο της υπηρεσίας. Στη συνέχεια πρέπει να ορίσουμε έναν έγκυρο χρήστη με ρόλο διαχειριστή για πρόσβαση στην υπηρεσία, το οποίο και πραγματοποιείται στο αρχείο tomcat-users.xml το οποίο βρίσκεται στο directory **/home/gria/Programs/apache-tomcat-5.5.23/conf/tomcat-users.xml**

Έπειτα επανεκκινούμε τον tomcat και ακολουθούμε τα βήματα που μας υποδεικνύονται για την πραγματοποίηση της ρύθμισης
Ορίζουμε την τοποθεσία του configuration file της υπηρεσίας. Στην συγκεκριμένη περίπτωση είναι το /home/gria/Programs/service-provider-mgt

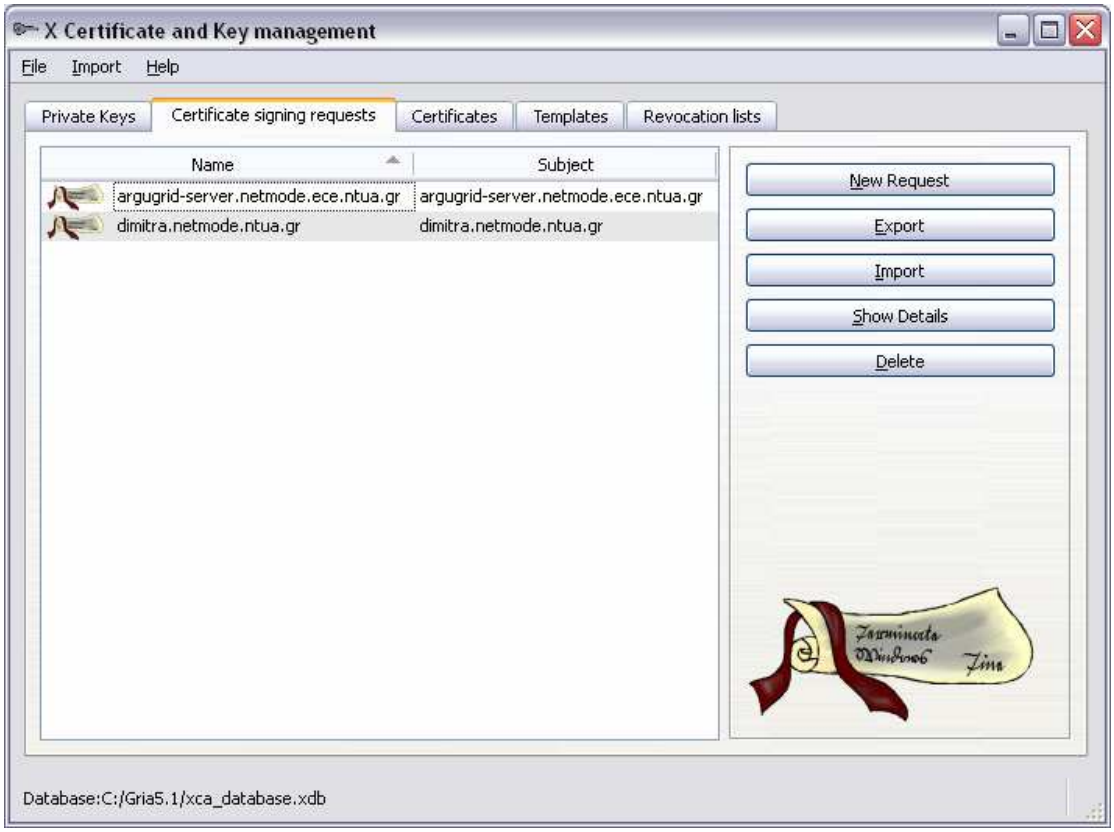


Ένα σημαντικό βήμα στην όλη διαδικασία είναι η δημιουργία έγκυρου κλειδιού και πιστοποιητικού για πρόσβαση στην υπηρεσία. Το GRIA παρέχει ένα εύχρηστο γραφικό περιβάλλον για την δημιουργία αρχικά ενός χώρου αποθήκευσης των κλειδιών (keystore) αλλά και για την δημιουργία κλειδιών και την υπογραφή τους από Έγκυρη Αρχή Πιστοποίησης (Certification Authority). Όπως φαίνεται και στο παρακάτω σχήμα αρχικά δημιουργείται ο χώρος αποθήκευσης στο συγκεκριμένο υπολογιστή. Στη συνέχεια με την εφαρμογή Key Tool Gui δημιουργείται το αντίστοιχο δημόσιο κλειδί και στέλνεται στην Αρχή Πιστοποίησης για υπογραφή (argugrid-request.csr)



Με την βοήθεια του προγράμματος xca δημιουργήθηκε η Έγκυρη Υπηρεσία Πιστοποίησης με το όνομα “NTUA” η οποία θα υπογράφει τα πιστοποιητικά που της στέλνονται.

Η αίτηση argugrid-request.csr στέλνεται στην Υπηρεσία Πιστοποίησης και αποθηκεύεται με το όνομα του υπολογιστή argugrid-server.netmode.ntua.gr που το δημιούργησε. Στην παρακάτω εικόνα φαίνεται η αίτηση που περιμένει να υπογραφεί από την Υπηρεσία Πιστοποίησης.



Η Αρχή Πιστοποίησης υπογράφει το αρχείο .csr οπότε το πιστοποιητικό είναι πλέον έγκυρο.

Η Αρχή Πιστοποίησης στέλνει στον υπολογιστή argugrid-server.netmode.ntua.gr από όπου έγινε η αίτηση τα δύο πιστοποιητικά: το NTUA.crt που είναι το πιστοποιητικό της ίδιας της Αρχής Πιστοποίησης και το argugrid-server.netmode.ece.ntua.gr.crt που είναι το νέο πιστοποιητικό που υπέγραψε.
Στον υπολογιστή argugrid-server.netmode.ntua.gr, στο keystore εισάγεται η NTUA.crt ως έγκυρη Αρχή Πιστοποίησης και στη συνέχεια
εισάγεται το πιστοποιητικό που υπογράφηκε από την Αρχή Πιστοποίησης.

Τώρα το keystore είναι έτοιμο για χρήση στην υπηρεσία Διαχείρισης Παρόχου Υπηρεσιών.

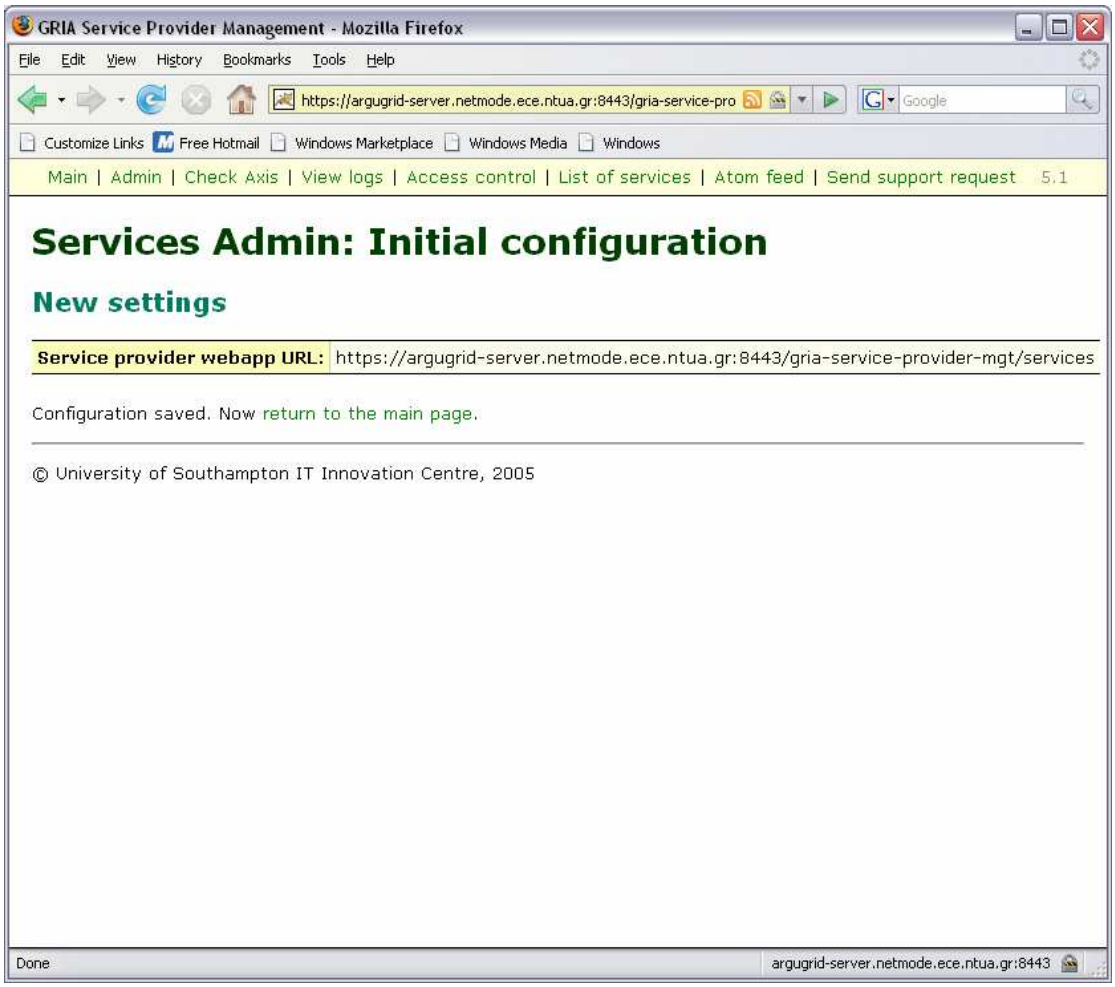
Αξίζει να σημειώσουμε ότι για τις υπόλοιπες υπηρεσίες που θα διαμορφωθούν στον υπολογιστή αυτό, δεν θα χρειαστεί η παραπάνω διαδικασία. Μπορεί να χρησιμοποιηθεί ο ίδιος χώρος αποθήκευσης κλειδιών σε όλες τις υπηρεσίες ώστε να υπάρχει πλήρης συμβατότητα μεταξύ των πιστοποιητικών τους και έτσι να μπορεί να έχει πρόσβαση η μία υπηρεσία στην άλλη. Χρειάζεται να επαναληφθεί μόνο το τελευταίο βήμα. Για να έχει πρόσβαση μια υπηρεσία σε υπηρεσία άλλου υπολογιστή θα πρέπει να στείλει το πιστοποιητικό της για υπογραφή και έγκριση από την Αρχή Πιστοποίησης του άλλου υπολογιστή δηλαδή να γίνει όλη η παραπάνω διαδικασία ώστε να θεωρείται ‘έμπιστη’ . Αυτό είναι απαραίτητο να γίνεται πριν την χρήση της υπηρεσίας SLA ώστε να έχει το δικαίωμα να διαχειρίζεται και να ελέγχει υπηρεσίες που ανήκουν σε άλλον υπολογιστή.

Ένα επιπλέον βήμα που απαιτείται για την διαμόρφωση της υπηρεσίας είναι η ενεργοποίηση των ασφαλών συνδέσεων (HTTPS Connections) ενεργοποιώντας το SSL στον Tomcat. Αυτό γίνεται διότι όλα τα SOAP messages υπογράφονται ώστε να αποφευχθεί η αλλοίωσή τους αλλά δεν κρυπτογραφούνται. Αντιθέτως η ίδια η σύνδεση θα πρέπει να είναι πάνω από μια κρυπτογραφημένη HTTPS σύνδεση εμποδίζοντας με τον τρόπο αυτόν τους επιτιθέμενους να δουν τα περιεχόμενα των μηνυμάτων.

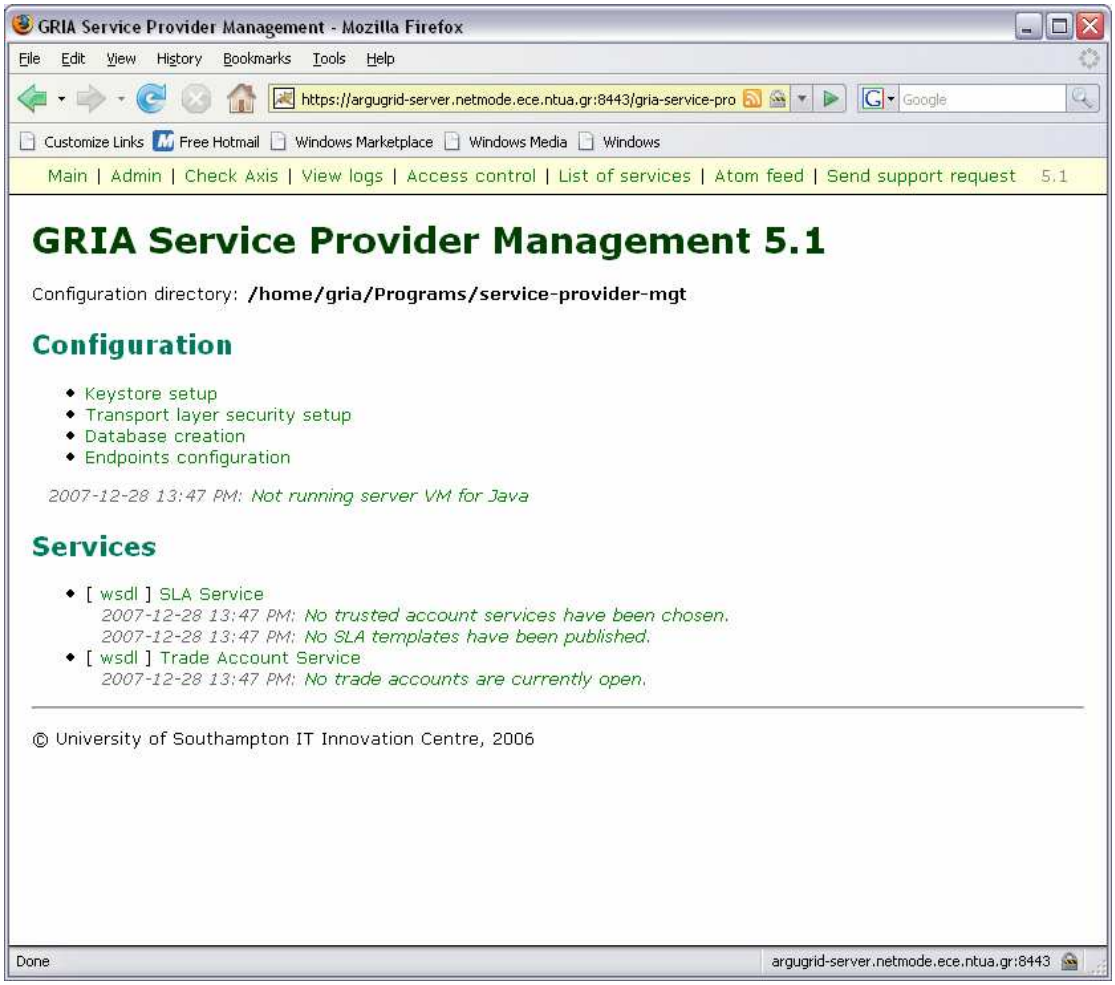
Επίσης το σύστημα απαιτεί πρόσβαση σε κάποια βάση δεδομένων για μόνιμη αποθήκευση.

Για καλύτερη απόδοση και αξιοπιστία πρέπει να χρησιμοποιηθεί μία ανεξάρτητη βάση δεδομένων. Στην δική μας υλοποίηση χρησιμοποιείται η Hypersonic. Μπορούν να χρησιμοποιηθούν και άλλες βάσεις δεδομένων αλλάζοντας το αρχείο **hibernate.properties** .

Τέλος χρειάζεται να οριστεί η endpoint address για την υπηρεσία μας δηλαδή το url με το οποίο η υπηρεσία θα είναι προσβάσιμη από τον φυλλομετρητή. Για την συγκεκριμένη υπηρεσία θα είναι το https://argugrid-server.netmode.ntua.gr:8443/gria-service-provider-mgt/services.



Η ρύθμιση της υπηρεσίας έχει ολοκληρωθεί και η υπηρεσία μπορεί να χρησιμοποιηθεί. Για πρώτη φορά η ιστοσελίδα της υπηρεσίας θα είναι:

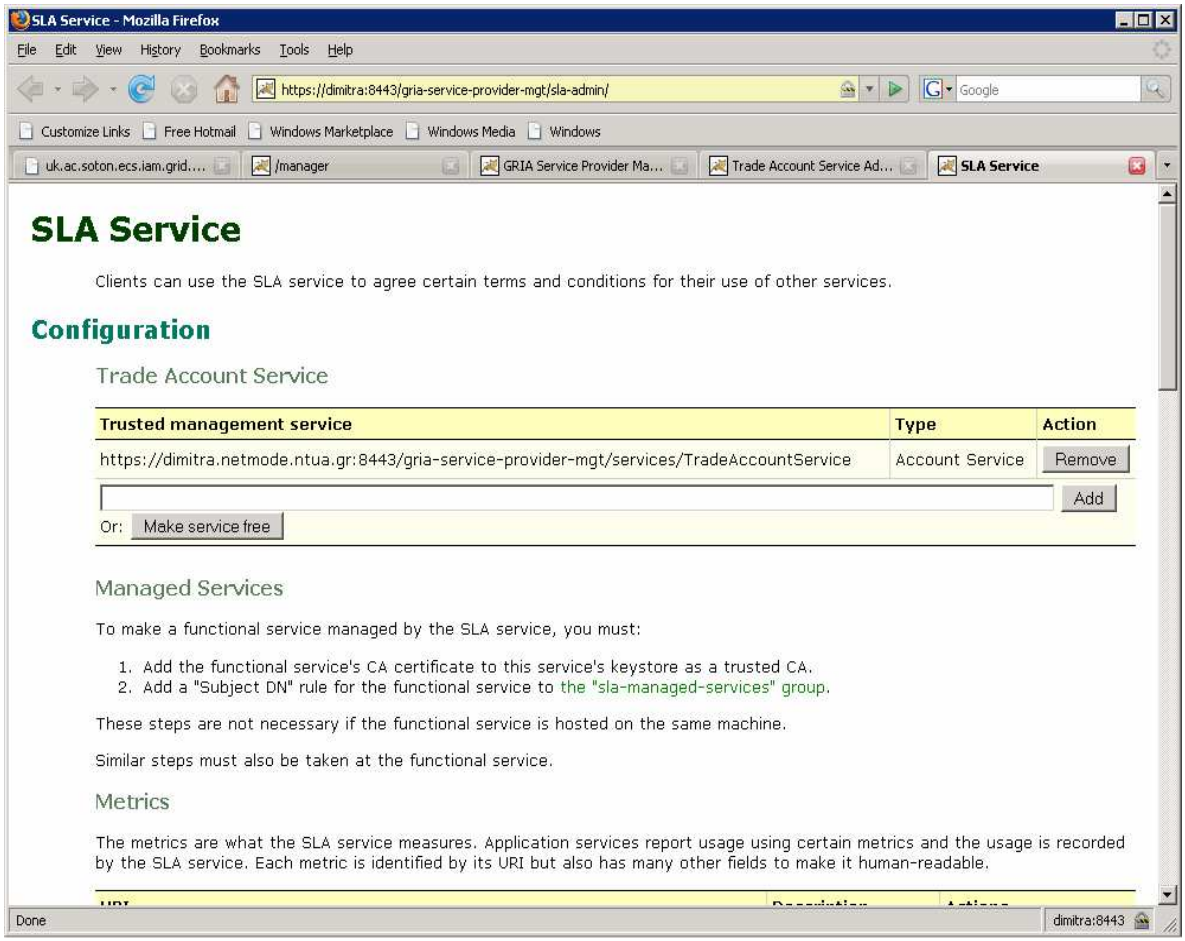

GRIA Service Provider Management - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://argugrid-server.netmode.ece.ntua.gr:8443/gria-service-pro
Google
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
Main | Admin | Check Axis | View logs | Access control | List of services | Atom feed | Send support request 5.1
GRIA Service Provider Management 5.1
Configuration directory: /home/gria/Programs/service-provider-mgt
Configuration
Keystore setup
Transport layer security setup
Database creation
Endpoints configuration
2007-12-28 13:47 PM: Not running server VM for Java
Services
[wsdl] SLA Service
2007-12-28 13:47 PM: No trusted account services have been chosen.
2007-12-28 13:47 PM: No SLA templates have been published.
[wsdl] Trade Account Service
2007-12-28 13:47 PM: No trade accounts are currently open.
© University of Southampton IT Innovation Centre, 2006
Done
argugrid-server.netmode.ece.ntua.gr:8443

3.2. *Χρήση της υπηρεσίας SLA και της υπηρεσίας Λογαριασμών ως διαχειριστής (manager)*

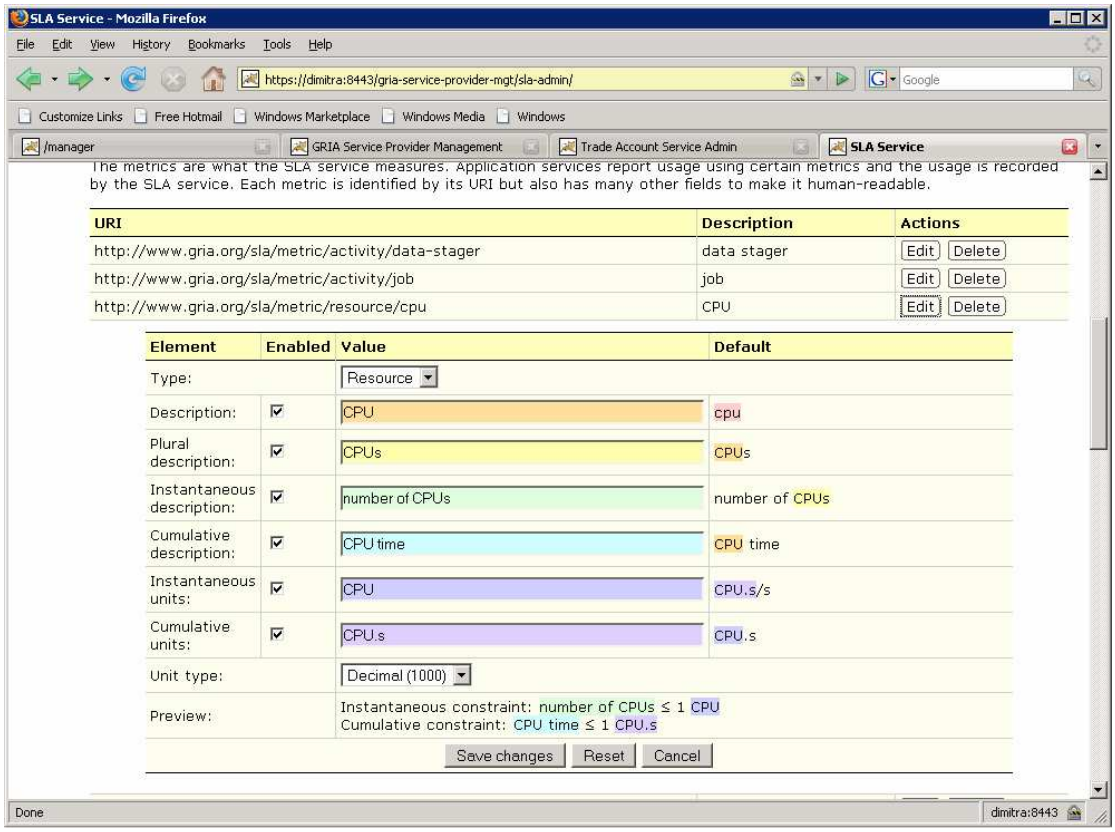
Η ρύθμιση (configuration) που κάνει ο διαχειριστής της υπηρεσίας

Στην παρακάτω ενότητα παρουσιάζεται η διαδικασία διαμόρφωσης και ρύθμισης των υπηρεσιών του πακέτου Διαχείρισης Παρόχου Υπηρεσιών., SLA Service και TradeAccount Service Οι εικόνες που παρουσιάζονται είναι από τον υπολογιστή dimitra.netmode.ntua.gr. Ωστόσο η ίδια διαδικασία ισχύει και για τις αντίστοιχες υπηρεσίες και των άλλων υπολογιστών του δικτύου.
Ρυθμίζουμε το νόμισμα για τις χρεώσεις σε EUR για ευρώ και την ακρίβεια σε 6 δεκαδικά ψηφία. Οποιαδήποτε συναλλαγή πραγματοποιείται πρέπει να χρησιμοποιεί αυτές τις ρυθμίσεις καθώς δεν γίνεται αυτόματη μετατροπή σε άλλα νομίσματα, οπότε και απορρίπτονται.

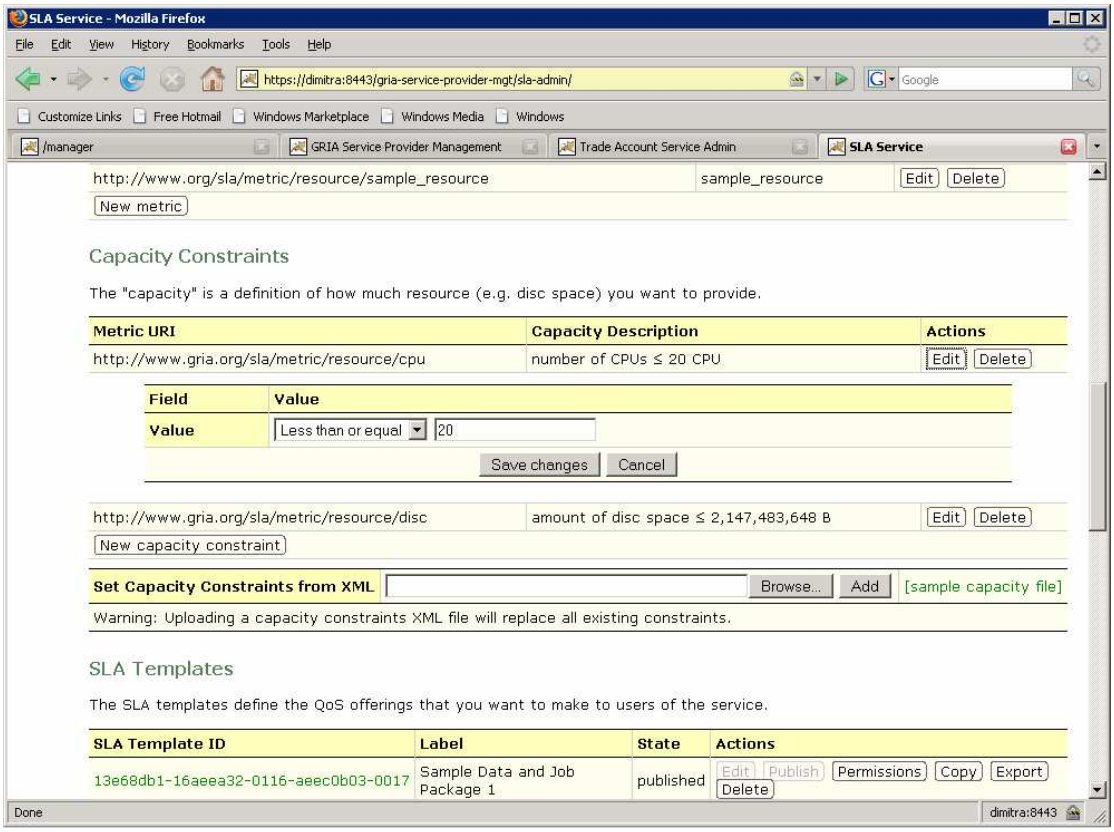
Προσθέτουμε την Trade Account Service στις υπηρεσίες τις οποίες θα εμπιστεύεται η SLA Service ώστε να μπορεί η SLA Service να χρεώνει για την χρήση των υπηρεσιών που απαιτούν SLA, τους λογαριασμούς των χρηστών. Με την επιλογή “Make service free” οι χρήστες που θα έχουν SLAs δεν θα χρεώνονται για την χρήση των υπηρεσιών. Αυτά φαίνονται στο παρακάτω σχήμα.



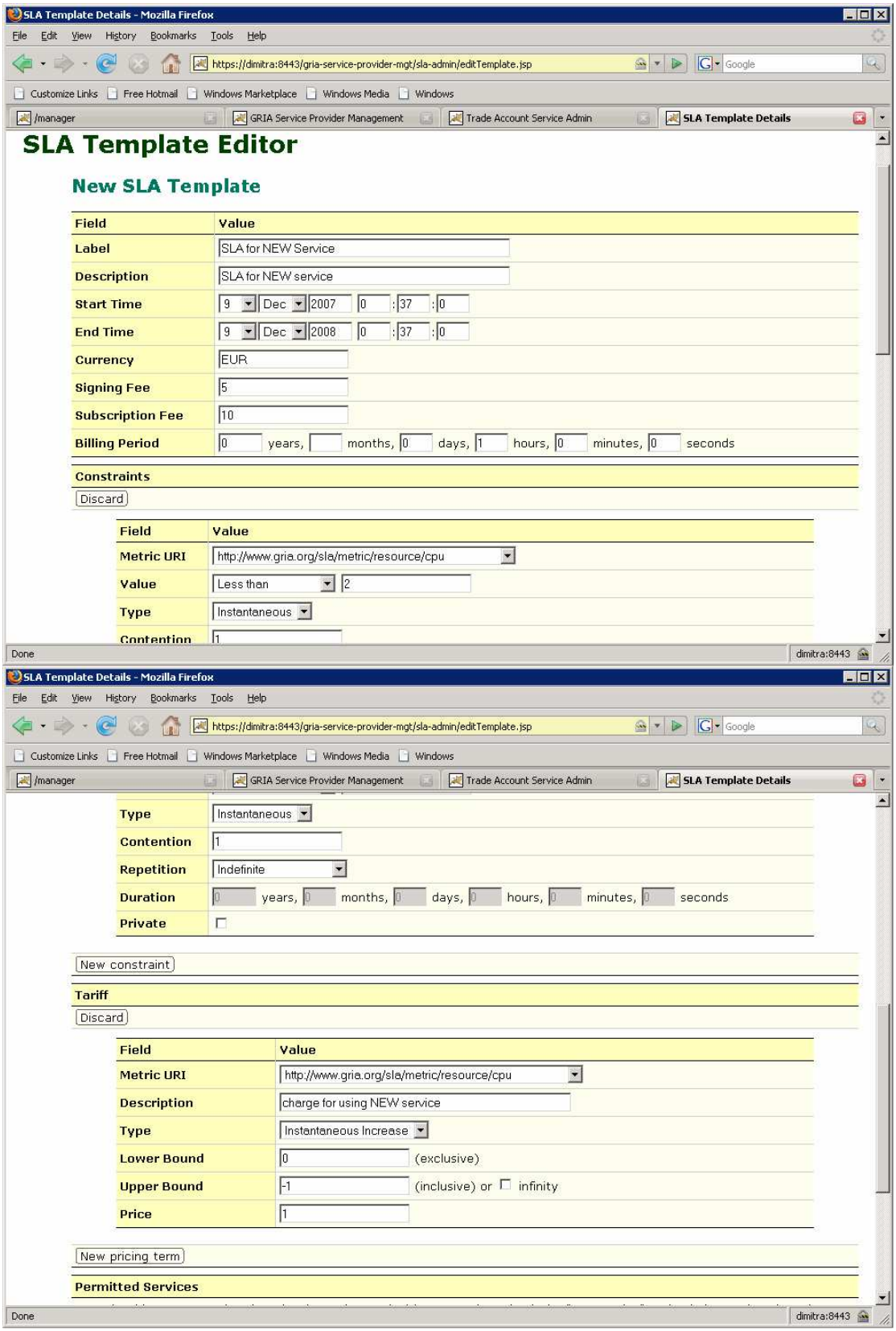
Στο παρακάτω σχήμα φαίνεται πώς χρησιμοποιείται ο επιμελητής μετρικών για την δημιουργία μιας μετρικής που μετράει την χρησιμοποίηση του πόρου CPU:

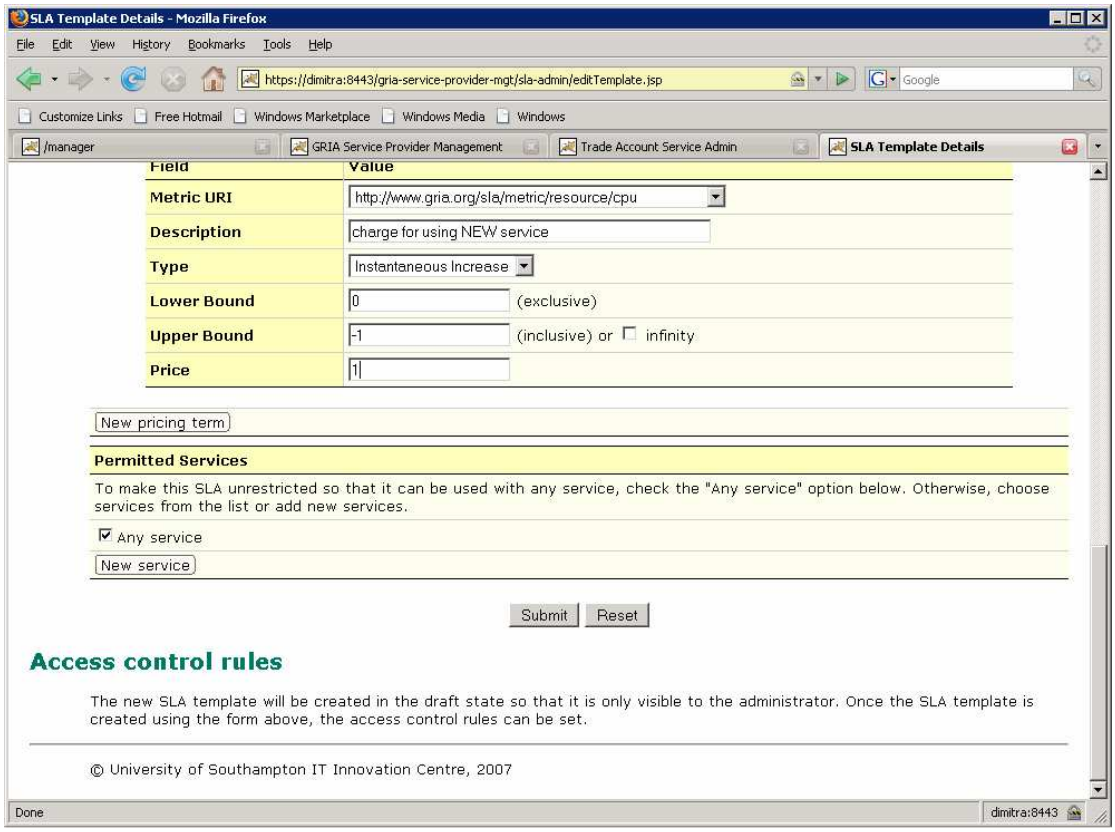


Στο επόμενο σχήμα φαίνεται πώς μπορούμε να ορίσουμε γραφικά την χωρητικότητα του συστήματος σε πόρους που αφορούν μονάδες CPU. Στην συγκεκριμένη περίπτωση ορίζουμε ότι το σύστημα μπορεί να διαθέσει συνολικά μέχρι και 20 μονάδες CPU:

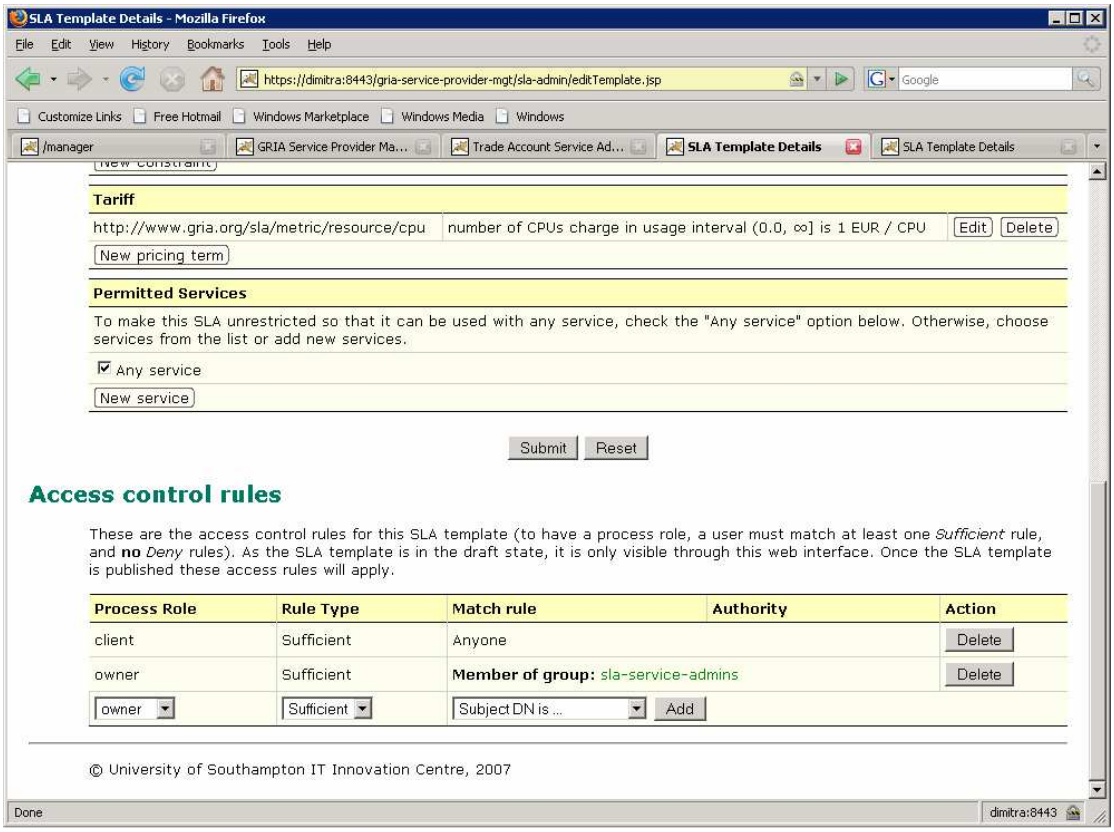


Στην συνέχεια φαίνεται πώς μπορούμε να ορίσουμε με γραφικό τρόπο τα SLA Templates δηλάδή τα xml αρχεία τα οποία περιγράφουν τι είδους πόροι μπορούν να διατεθούν για την εκτέλεση κάποιας συγκεκριμένης υπηρεσίας, τι πόσότητα από αυτούς τους πόρους μπορεί να χρησιμοποιήσει ένας πελάτης σε ένα συγκεκριμένο χρονικό διάστημα που ορίζεται, τι περιορισμοί ισχύουν για την χρησιμοποίηση αυτών των πόρων, και την χρέωση που υφίσταται ο χρήστης για την χρησιμοποίηση που μπορεί να κάνει, αν είναι επιθυμητό να υπάρχει χρέωση. Στις παρακάτω τρεις εικόνες φαίνεται η μορφή του SLA template.


SLA Template Details - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://dimitra:8443/gria-service-provider-mgt/sla-admin/editTemplate.jsp
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
/manager
GRIA Service Provider Management
Trade Account Service Admin
SLA Template Details
SLA Template Editor
New SLA Template
Field
Value
Label
SLA for NEW Service
Description
SLA for NEW service
Start Time
9 Dec 2007 0 : 37 : 0
End Time
9 Dec 2008 0 : 37 : 0
Currency
EUR
Signing Fee
5
Subscription Fee
10
Billing Period
0 years, 0 months, 0 days, 1 hours, 0 minutes, 0 seconds
Constraints
Discard
Field
Value
Metric URI
http://www.gria.org/sla/metric/resource/cpu
Value
Less than 2
Type
Instantaneous
Contention
1
Done
dimitra:8443
SLA Template Details - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://dimitra:8443/gria-service-provider-mgt/sla-admin/editTemplate.jsp
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
/manager
GRIA Service Provider Management
Trade Account Service Admin
SLA Template Details
Type
Instantaneous
Contention
1
Repetition
Indefinite
Duration
0 years, 0 months, 0 days, 0 hours, 0 minutes, 0 seconds
Private
New constraint
Tariff
Discard
Field
Value
Metric URI
http://www.gria.org/sla/metric/resource/cpu
Description
charge for using NEW service
Type
Instantaneous Increase
Lower Bound
0 (exclusive)
Upper Bound
-1 (inclusive) or infinity
Price
1
New pricing term
Permitted Services
Done
dimitra:8443



Το τελευταίο βήμα είναι να καθορίσουμε αν το SLA template μπορεί να είναι προσβάσιμο από όλους τους πελάτες ή αν θα υπάρχουν περιορισμοί και θα είναι ορατό σε πελάτες που ανήκουν σε συγκεκριμένες ομάδες. Αυτό σημαίνει να είναι ορατό από σταθμούς (υπολογιστές) που διαθέτουν συγκεκριμένα πιστοποιητικά. . Στην συγκεκριμένη περίπτωση ορίσαμε ότι το SLA template θα είναι ορατό σε όλους τους πελάτες όπως φαίνεται και στο παρακάτω σχήμα.



Το SLA template είναι σε πρόχειρη κατάσταση (draft state) και δεν είναι ακόμα ορατό στους πελάτες. Αφού δημοσιοποιηθεί μπορεί να είναι ορατό στους χρήστες και πλέον δεν μπορεί να αλλάξει.

3.3. Δημιουργία της Hello Service

Δημιουργήσαμε μία νέα διαδικτυακή υπηρεσία την Hello Service η οποία μπορεί να εκτελείται στο GRIA μεσισμικό πλέγματος και είναι συμβατή με τους κανόνες ασφαλείας και την διαχείριση στο GRIA. Για την δημιουργία της χρησιμοποιήθηκε το GRIA API 5.1 το οποίο παρέχεται από το GRIA και χρησιμοποιήθηκε και το Service Developer's Toolkit. Επίσης δημιουργήθηκε και ένας νέος client, μέσω της κλάση CallHelloservice.java, μέσω της οποίας μπορεί να καλείται η υπηρεσία.
Η Hello Service αποτελεί μια γενική υπηρεσία και παρέχει λειτουργίες για την δημιουργία και καταστροφή πόρων. Οι πόροι στην συγκεκριμένοι υλοποίηση δεν επιτελούν κάποια συγκεκριμένη λειτουργίαπαρά μόνο κάθε φορά που δημιουργείται κάθε νέος πόρος τυπώνεται η φράση "Hello world!".Μπορούμε να επεκτείνουμε την λειτουργικότητα της υπηρεσίας και ο πόρος που δημιουργεί να είναι περισσότερο εξειδικευμένος ανάλογα με το είδος των πόρων που πρέπει να διαχειρίζεται η υπηρεσία. Κάθε φορά που δημιουργείται ένα πόρος μπορούμε να ορίσουμε μέσω κατάλληλων μεθόδων και επιπλέον λειτουργίες να εκτελούνται.

Για την δημιουργία της Hello Service χρησιμοποιήθηκαν οι παρακάτω διεπαφές από την βιβλιοθήκη του GRIA:

- StorableInStateRepository: αντικείμενο το οποίο μπορεί να προστεθεί στο repository του client.
- Conversation : πληρεξούσιος σε έναν απομακρυσμένο πόρο στην πλευρά του πελάτη.
- RemoteService: πληρεξούσιος σε μία απομακρυσμένη υπηρεσία στην πλευρά του πελάτη.
- GridService: Λειτουργίες που επιστρέφουν τα πιστοποιητικά και τους πόρους τις υπηρεσίας.
- ManagedGridService: Υπηρεσία που διαχειρίζεται από έναν λογαριασμό ή μία υπηρεσία SLA.
- ReportAPI: Μια υπηρεσία η οποία μπορεί να στείλει αναφορές χρησιμοποίησης σε μία υπηρεσία SLA.
- WebAdmin: Μία υπηρεσία την οποία μπορεί να χειριστεί ένας διαχειριστής χρησιμοποιώντας μία διαδικτυακή διεπαφή.
- PolicyManagement: Ένας πόρος στον οποίο μπορούμε να διαχειριστούμε τους κανόνες ελέγχου πρόσβασης.
- WSResourceLifetime: Ένας πόρος που μπορεί να καταστραφεί σύμφωνα με τους όρους του WS-Resource Lifetime.
- ResourceMetadata: Ένας πόρος με ετικέτα που καθορίζεται από τον χρήστη και περιέχει μεταδεδομένα.

Πακέτο hellogria.common

Αποτελείται από δύο βασικές διεπαφές (java interfaces) :

HelloService.java

Περιλαμβάνει όλες τις SOAP λειτουργίες οι οποίες δεν παίρνουν κανένα περιεχόμενο (context), δηλαδή ενεργούν πάνω στην ίδια την υπηρεσία και όχι σε κάποιο συγκεκριμένο πόρο. Επεκτείνει την ManagedGridService οπότε ο διαχειριστής της υπηρεσίας (service administrator) μπορεί να προσθέτει, να αφαιρεί και να ρωτάει υπηρεσίες λογαριασμών που τις εμπιστεύεται η υπηρεσία. Αυτό σημαίνει ότι όταν ο χρήστης της υπηρεσίας πρέπει να πληρώσει κάποιο ποσο για την χρήση της υπηρεσίας θα πρέπει να έχει πρόσβαση σε λογαριασμό που ανήκει σε κάποια από αυτές τις υπηρεσίες. Επεκτείνει επίσης την ReportAPI οπότε επιτρέπει στην SLA υπηρεσία να λαμβάνει αναφορές για την χρησιμοποίηση των πόρων της υπηρεσίας από την ίδια την υπηρεσία. Επεκτείνει την GridService οπότε παρέχει μεθόδους για να λαμβάνει πιστοποιητικά για την ταυτότητα του χρήστη και για να ζητήται η δημιουργία πόρων. Επεκτείνει τέλος και την WebAdmin. Καθορίζεται ότι μπορεί να χρησιμοποιηθεί SLA. Ορίζεται ο τύπος της ίδιας της υπηρεσίας μέσου του μοναδικού URI:

HELLO_RESOURCE_TYPE =http://HELLOGRIA/2007/HelloServiceResourceType.

Η PBAC policy (πολιτική????) με αυτό το όνομα χρησιμοποιείται για να προστατεύει λειτουργίες πάνω στην υπηρεσία.

Ορίζεται η μέθοδος createHelloResource η οποία δημιουργεί τον νέο πόρο. Παίρνει ως παράμετρο ένα MatchPattern owner και όποιος ταιριάζει με τον κανόνα owner μπορεί να έχει πρόσβαση στον πόρο ως είναι κάτοχός του. Τυπικά αυτός ο κανόνας απλά ταιριάζει στην ταυτότητα του ατόμου που δημιούρησε τον πόρο και επιστρέφει το epr του δηλαδή το endpoint reference του. Όταν καλείται αυτή η μέθοδος μπορεί να απαιτείται ένας λογαριασμός ή η συμφωνία ενός SLA και ο ορισμός τους στο SOAP header, ανάλογα με το πώς έχει ρυθμιστεί η υπηρεσία.

HelloResource.java

Είναι η διεπαφή για την δημιουργία των αντικειμένων που θα αποτελούν τον πόρο που δημιουργήθηκε από την HelloService.

Περιλαμβάνει όλες τις SOAP λειτουργίες της υπηρεσίας οι οποίες παίρνουν την HelloResource ως το περιεχόμενο τους δηλαδή ενεργούν πάνω στο αντικείμενο HelloResource. Είναι η διεπαφή για την δημιουργία των αντικειμένων που θα αποτελούν τους πόρους. Κάθε μέθοδος που ορίζεται σε αυτό το interface είναι μία SOAP λειτουργία στην υπηρεσία HelloResource. Επίσης εδώ ορίζεται και η μέθοδος HelloWorld() η οποία καλείται κάθε φορά που δημιουργείται ο νέος πόρος και επιστρέφει ένα string “Hello World!”. Η μέθοδος αυτή ορίζεται επίσης και στο αρχείο όπου ορίζεται η PBAC πολιτική για τους πόρους. Εδώ πρέπει να οριστούν και οποιεσδήποτε άλλες μέθοδοι χρησιμοποιούνται από το αντικείμενο που αντιπροσωπεύει το νέο πόρο.Ορίζεται ο τύπος του πόρου που δημιουργείται από την HelloService μέσω του μοναδικού uri:

HELLO_RESOURCE_TYPE = http://HELLOGRIA/2007/HelloResourceType.

Η PBAC πολιτική με αυτό το όνομα χρησιμοποιείται για να προστατέψει λειτουργίες πάνω στην HelloService οι οποίες επικαλούνται ένα context με αυτόν τον τύπο. Όταν ανακαλύπτονται πόροι έχουν αυτόν τον τύπο στα μεταδεδομένα τους.

HelloResourceBean.java

Είναι ένα Java bean που αντιπροσωπεύει έναν πόρο. Οι νέοι πόροι δημιουργούνται με την μέθοδο createHelloResource που είναι μία SOAP λειτουργία. Αποθηκεύονται χρησιμοποιώντας την hibernate βάση δεδομένων.
Εδώ μπορούμε να ορίσουμε τις μεθόδους get και set για την υλοποίηση κάποιας λειτουργίας πάνω σε κάποιον πόρο που δημιουργείται. Τα αντικείμενα που επιστρέφονται από την μέθοδο get αποτελούν πεδία του αντικειμένου που αντιπροσωπεύει τον νέο πόρο και πρέπει να οριστούν επίσης στο αρχείο HelloResourceBean.hbm.xml ώστε η hibernate να τα αποθηκεύει στην βάση δεδομένων. Στην συγκεκριμένη υλοποίηση δεν υλοποιείται κάτι τέτοιο. Ορίζεται απλά η getResourceType η οποία επιστρέφει τον τύπο του νέου resource και ο οποίος έχει οριστεί στην διεπαφή HelloResource.

Πακέτο hellogria.impl

HelloServiceImpl.java

Η κλάση αυτή υλοποιεί την διεπαφή HelloService. Στο αρχείο implementationfactory.properties ορίζεται η αντιστοίχιση της διεπαφής με την κλάση:

```
hellogria.HelloService = hellogria.impl.HelloServiceImpl
```

Η κλάση κάνει extend το interface GridServiceLite.

Ορίζεται ότι ο χρήστης θα χρεώνεται με το ποσό των 12.50 ευρώ για την δημιουργία κάθε νέου πόρου. Αν δεν πληρωθεί το ποσό ο πόρος δεν δημιουργείται.

Ορίζεται η μονάδα μέτρησης (metric) για την μέτρηση του αριθμού των πόρων με το μοναδικό URI:

```
 static final Metric METRIC_HELLO_RESOURCES = new
Metric("http://SAMPLE/2006/metrics/sample-resource");
```

Θα χρησιμοποιείται από την υπηρεσία SLA για τον περιορισμό του αριθμού των πόρων που θα μπορούν να δημιουργούνται από έναν χρήστη.
Οι αναφορές για την χρησιμοποίηση των πόρων αποθηκεύονται εδώ:

```
ReportAPIHelper pullPoint = new ReportAPIHelper();
```

Ορίζεται η μέθοδος ensurePoliciesDeployed() η οποία καλείται από τον κώδικα του διαχειριστή όταν βλέπει την κεντρική σελίδα διαχείρησης. Εδώ ορίζονται οι PBAC policies και τα resources που χρειάζονται. Προστίθονται η policy hello-service-policy.xml για την προστασία των λειτουργιών στην διεπαφή της HelloService και η policy hello-policy.xml η οποία προστατεύει τις λειτουργίες που γίνονται πάνω στην διεπαφή HelloResource. Η ίδια η υπηρεσία προστίθεται ως πόρος. Ορίζεται η υλοποίηση της createHelloResource όπου γίνεται έλεγχος αν ο τρέχων χρήστης που καλεί την HelloService έχει δικαίωμα να δημιουργήσει νέο resource και επομένως αν θα έχει πρόσβαση σε αυτό, δημιουργείται πληρεξούσιος (proxy) στον πόρο που θα διαχειρίζεται την υπηρεσία και ο οποίος μπορεί να είναι ένας λογαριασμός ή κάποιο

SLA που έχει συμφωνηθεί από τον πελάτη, ελέγχει αν η υπηρεσία εμπιστεύεται τις αντίστοιχες υπηρεσίες, δημιουργείται το νέο αντικείμενο που αντιπροσωπεύει τον πόρο (Java bean object)

```
HelloResourceBean resource = new
HelloResourceBean(managingConversation, label);
```

Σώζεται με το hibernate το οποίο δίνει ένα ID για το resource το οποίο χρησιμοποιείται από παρακάτω μεθόδους.

```
addHibernateObject(resource);
String resourceID = resource.getResourceID();
```

Κάνει έλεγχο αν ο χρήστης έχει αρκετά χρήματα ή αν έχει περιθώριο από το SLA που έχει συμφωνήσει για την δημιουργία καινούριου πόρου.

```
checkCreationOK(resource, managingConversation);
```

Καταγράφει την δημιουργία του νέου πόρου και χρεώνει τον χρήστη για αυτή.

```
recordCreation(resource, managingConversation,
getCurrentUser().toID());
```

Σε αυτό το σημείο έχει ελεγθεί η εγκυρότητα του λογαριασμού και του SLA του χρήστη και ο χρήστης έχει χρεωθεί για την δημιουργία του καινούριου πόρου. Αν αποτύχει κάποια από τις παραπάνω λειτουργίες ως αυτό το σημείο το resource που έχει δημιουργηθεί καταστρέφεται και ενημερώνεται το SLA μηδενίζοντας την χρησιμοποίηση του resource. Ως αυτό το σημείο το resource ήταν κλειδωμένο («locked state») και οι οποιεσδήποτε αιτήσεις προς αυτό έμπαιναν σε ουρά αναμονής. Από εδώ και πέρα το resource είναι unlocked και μπορούν να εκτελεστούν διάφορες λειτουργίες πάνω σε αυτό.

Μετά υλοποιείται η μέθοδος checkCreationOK η οποία, αν ο πόρος που διαχειρίζεται το νέο resource είναι λογαριασμός, ελέγχει αν υπάρχει το όριο πίστωσης που έχει δωθεί για τον χρήστη που την κάλεσε ενώ αν πρόκειται για SLA προσθέτει τον περιορισμό να επιτρέπει το SLA την δημιουργία (μέχρι) ενός πόρου από τον χρήστη.
Σε αυτό το σημείο έχει δημιουργηθεί ο καινούριος πόρος και ενημερώνονται οι υπηρεσίες που διαχειρίζονται τους πόρους της HelloService. Συγκεκριμένα χρεώνεται ο λογαριασμός του χρήστη

```
account.bill(user, createResourceCharge, "EUR",
resource.getResourceID(), "Created sample resource");
```

Ενημερώνεται το SLA αν υπάρχει κάποια άλλη πληροφορία για χρησιμοποίηση.

Επίσης ορίζονται ενδείξεις (notices) από την υπηρεσία ως Atom Feed και παρουσιάζονται στην κεντρική σελίδα διαχείρησης. Το feed??? Χρησιμοποιείται για να αναφέρει στον διαχειριστή τυχόν προβλήματα κατά την ρύθμιση της υπηρεσίας (configuration).

```
public Document getAtomFeed(String atomFeed, String serviceBase) {
            Document doc = AtomUtils.getEmptyAtomFeed("Hello
Service", "Hello Service",atomFeed, serviceBase);
```

HelloResourceImpl.java

Είναι η υλοποίηση της διεπαφής HelloResource και επεκτείνει την GridServiceLite. Η αντιστοίχηση μεταξύ της διεπαφής και της κλάσης υλοποίησης ορίζεται στο αρχείο implementationfactory.properties

```
      hellogria.HelloResource = hellogria.impl.HelloResourceImpl
```

Η μέθοδος getResource παίρνει ένα HelloResourceBean το οποίο έχει αποθηκευτεί χρησιμοποιώντας hibernate.

Η μέθοδος getManagingConversation δημιουργεί έναν πληρεξούσιο του πελάτη (client proxy) στον διαχειριστικό πόρο ο οποίος όπως ειπώθηκε και προηγουμένως μπορεί να είναι ο λογαριασμός του χρήστη ή το SLA που έχει συμφωνήσει ο χρήστης για την χρήση της HelloService.

Η μέθοδος destroy ψάχνει τον πόρο που είναι HelloResourceBean και το καταστρέφει όταν έχει τελειώσει η χρησιμοποίηση του από τον χρήστη. Επίσης στέλνεται ειδοποίηση στον SLA διαχειριστή ότι η χρησιμοποίηση των πόρων της Hello Service έχει μηδενιστεί μέσω του metric για τον πόρο. Ειδοποιείται επίσης το PBAC και διαγράφεται ο πόρος από το hibernate. Ομοίως με πριν επιστρέφονται ειδοποιήσεις ως Atom Feed.

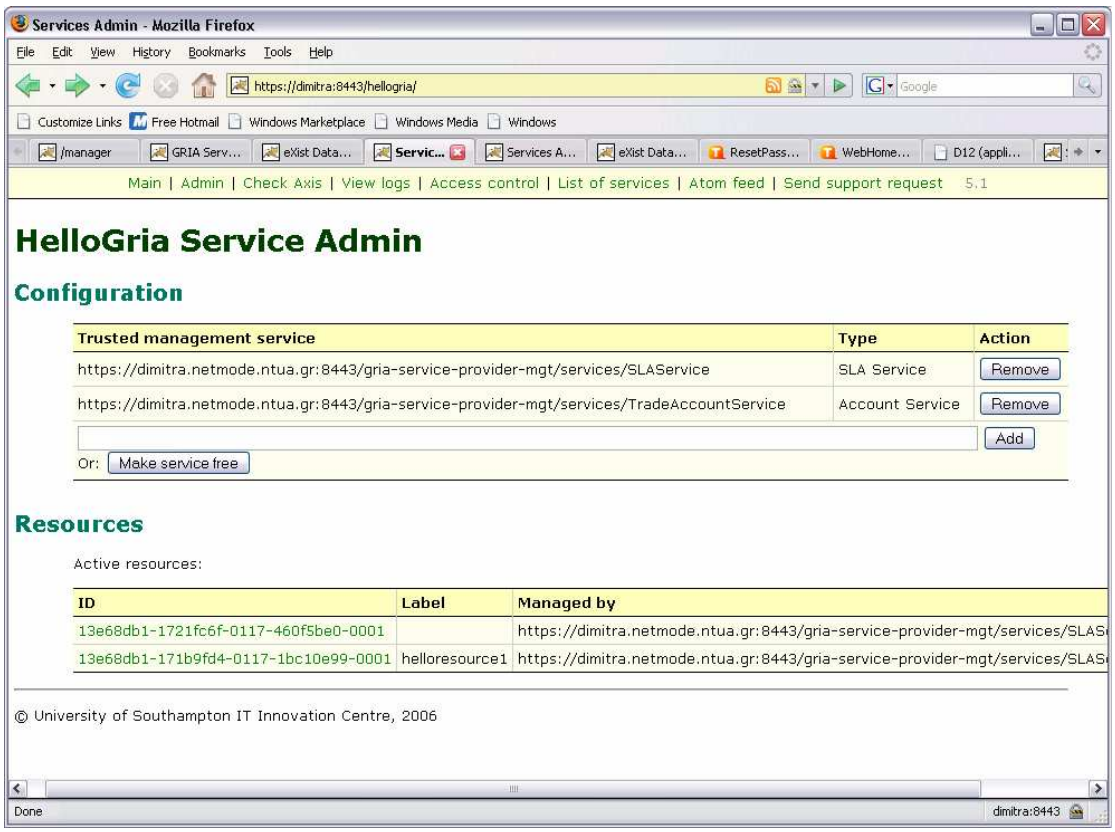
Στο τέλος της κλάσης υλοποιούνται οι μέθοδοι που θα εφαρμόζονται πάνω στο νέο πόρο που έχει δημιουργηθεί. Στην συγκεκριμένη υλοποίηση είναι η HelloWorld() η οποία δίνεται παρακάτω:

```
public String HelloWorld() throws RemoteException {
  String conversationID = getConversationFromContext();
  Session session = factory.openSession();

  try {
    HelloResourceBean resource = getResource(session,conversationID);

//Εδώ χρεώνεται ο χρήστης για την κλήση της μεθόδου

Conversation                  managingConversation                =
getManagingConversation(resource);
if (managingConversation instanceof TradeAccountConversation)
    {
      TradeAccountConversation  account  =  (TradeAccountConversation)
managingConversation;
```

```
 // the arguments to bill are: name of the user, the amount to
charge, // a message, sample resource's id

                account.bill(getCurrentUser().toID(),
                new BigDecimal(1.0), "EUR",
                conversationID,
                "Invoked HelloWorld()");
            }
                   Transaction tx = session.beginTransaction();
                   tx.commit();

    // Αποτέλεσμα της μεθόδου

                   return "Hello World!";
            } finally {session.close();
```

Αφού τελειώσει η υλοποίηση της υπηρεσίας, φτιάχνουμε τους φακέλους με τις κλάσεις, προσθέτουμε και τα απαραίτητα αρχεία όπως το pom.xml, configuration files και κάνουμε τις απαραίτητες αλλαγές στα υπόλοιπα απαραίτητα αρχεία.Φτιάχνουμε το hellogria.war αρχείο με την βοήθεια του πακέτου Apache Maven version 2.0.7 το οποίο κατεβάσαμε από το maven web-site [21] και το οποίο εγκαταστήσαμε στον φάκελο όπου είναι εγκατεστημένος ο Tomcat. Με την εντολή mvn install από το command prompt στο φάκελο όπου βρίσκεται όλο το πακέτο για την Hello Service, δημιουργούμε το war αρχείο το οποίο εισάγουμεy στον Tomcat.

Στην παρακάτω εικόνα φαίνεται η σελίδα της υπηρεσίας. Στο τμήμα 'Resources' φαίνονται οι πόροι της υπηρεσίας που έχουν δημιουργηθεί από κάποιον χρήστη.



Πακέτο hellogria.client

Οι παρακάτω κλάσεις χρησιμοποιήθηκαν για την υλοποίηση της πλευράς του χρήστη (client) για την υπηρεσία Hello service.

HelloConversation.java

Είναι διεπαφή (interface) για την υλοποίηση ενός πληρεξούσιου (proxy) σε πόρο που ανήκει σε απομακρυσμένη Hello Service. Επεκτείνει το SOAP interface HelloResource οπότε χρησιμοποιούνται οι ίδιες μέθοδοι για αυτό το αντικείμενο αλλά για την πλευρά του client. Μπορεί να υπάρχουν και κάποιες τοπικές διαχειριστικές μέθοδοι.

RemoteHelloService.java

Είναι interface και αποτελεί πληρεξούσιο (proxy) σε ένα instance (????) της υπηρεσίας. Ορίζεται η μέθοδος createHelloResource η οποία όμως εδώ επιστρέφει ένα HelloConversation δηλαδή έναν πληρεξούσιο σε απομακρυσμένο πόρο. Αποτελεί ουσιαστικά ένα "περιτύλιγμα" για την αντίστοιχη μέθοδο στην διεπαφή HelloService η οποία δημιουργεί ένα policy κάνοντας τον χρήστη που την καλεί τον κάτοχο του νέου πόρου. Παίρνει ως παράμετρους μία επιγραφή που περιγράφει τοπικά το conversation (????δηλαδή τον proxy) και το epr δηλαδή το url της υπηρεσίας που διαχειρίζεται τον πόρο της απομακρυσμένης υπηρεσίας.

HelloserviceHelpers.java

Η κλάση αυτή περιλαμβάνει μεθόδους για την RemoteHelloService οι οποίες δεν είναι απλές Web μέθοδοι????? Εδώ υλοποιείται η createHelloResource που ορίστηκε στην διεπαφή RemoteHelloService:

```
public HelloConversation createHelloResource(String description,
EndpointReferenceType billingInfo) throws RemoteException {
      proxy.setBillingInfo(billingInfo);
      HelloConversation helloconv = createHelloResource(description);
      proxy.setBillingInfo(null);
      return helloconv;

      }
```

Υλοποίηση της κλάσης που υλοποιεί την πλευρά του πελάτη (client) για την κλήση και χρήση της υπηρεσίας HelloService

Η υπηρεσία HelloService έχει οριστεί έτσι ώστε για την χρήση της να απαιτείται από τον χρήστη η ύπαρξη ενός λογαριασμού και η συμφωνία ενός SLA. Έχουμε ορίσει τις υπηρεσίες SLA Service και Trade Account Service ως Trusted υπηρεσίες δηλαδή υπηρεσίες που τις εμπιστεύεται η Hello Service. Αυτό το κάναμε με τον ρόλο του διαχειριστή της υπηρεσίας. Ο κώδικας της κλάσης αυτής παρέχεται στο παράρτημα.

Περιγραφή του κώδικα:

Σταθερές

Αρχικά ορίσαμε ως σταθερές τα epr της Hello Service και της SLA Service η οποία διαχειρίζεται την Hello Service.

Main

Στην πλευρά του client δημιουργείται ένα repository .Το repository είναι ένας χώρος όπου αποθηκεύονται πόροι και υπηρεσίες που δημιουργεί ή χρησιμοποιεί ο χρήστης. Δημιουργείται ως αντικείμενο της κλάσης MemoryStateRepository:

```
StateRepository repository = new MemoryStateRepository();
```

Τα αντικείμενα στο repository αντιπροσωπεύονται από το epr τους το οποίο περιέχει όλες τις πληροφορίες για αυτά, όπως uri διεύθυνση της υπηρεσίας από την οποία δημιουργήθηκαν και το χαρακτηριστικό ID τους που τους δόθηκε κατά την δημιουργία τους.

Δημιουργείται ένας πληρεξούσιος (proxy) στην Hello Service και προστίθεται στο repository.

Δημιουργείται ένας πληρεξούσιος (proxy) στην Trade Account Service και προστίθεται στο repository.

Αν ο χρήστης έχει ήδη έναν λογαριασμό σε αυτήν την υπηρεσία δίνει το χαρακτηριστικό του (ID) και αν όντως υπάρχει στην Trade Account Service προστίθεται ως πόρος στο repository. Αν δεν έχει λογαριασμό, δημιουργεί έναν νέο λογαριασμό. Δίνει τα στοιχεία του και με την μέθοδο openAccount ζητάει την δημιουργία νέου λογαριασμού. Ο λογαριασμός είναι στην κατάσταση pending-for-credit-checks και είναι ανενεργός. Στην πλευρά του παρόχου τώρα, αν ο χρήστης είναι έμπιστος και ανάλογα με τον βαθμό εμπιστοσύνης, ο διαχειριστής της Trade Account Service αποδέχεται την αίτηση για τον νέο λογαριασμό και ορίζει ένα όριο πίστωσης για τον χρήστη αυτό. Ο λογαριασμός περνάει σε κατάσταση open και πλέον μπορεί να χρησιμοποιηθεί. Στην πλευρά του client τώρα πάλι ο νέος λογαριασμός προστίθεται όπως και πριν ως πόρος στο repository.

Δημιουργείται ένας proxy στην SLA Service. Αν έχει ήδη συμφωνήσει SLA ο χρήστης για την Hello Service, δίνει το όνομα του SLA και αν όντως βρεθεί στην SLA Service προστίθεται στο repository ως πόρος χρησιμοποιώντας το epr του. Αν δεν έχει συμφωνήσει SLA πρέπει να συμφωνήσει ένα. Έτσι γίνεται αναζήτηση και παρατίθενται όλα τα SLA Templates τα οποία παρέχονται από τον πάροχο της υπηρεσίας μέσω της SLA Service, τα οποία ο χρήστης μπορεί να δει αναλυτικά αν επισκεφθεί την σελίδα της SLA service. Ο χρήστης μπορεί τοτε να επιλέξει αυτό που επιθυμεί δίνοντας το όνομα του ????αντι για το ονομα να δίνει το epr????και το

προτείνει στην SLA service. Του δίνει και κάποιο όνομα και μέσω της μεθόδου createSLA δημιουργείται το νέο SLA. Να σημειωθεί ότι η μέθοδος αυτή κάνει έλεγχο αν ο χρήστης δικαιούται την δημιουργία SLA ελέγχοντας τον λογαριασμό του και το υπόλοιπό του. Ομοίως με πριν το SLA που δημιουργήθηκε αποθηκεύεται στο repository.
Από αυτό το σημείο ο χρήστης μπορεί να προχωρήσει στην χρήση της Hello Service και να δημιουργήσει έναν καινούριο πόρο ανάλογα με το τι του επιτρέπει το υπόλοιπο του λογαριασμού του και η συμφωνία που έχει κάνει στο SLA. Δίνει ένα όνομα για τον νέο πόρο και στην συνέχεια καλείται η createHelloResource με παραμέτρους το όνομα που έδωσε οο χρήστης και το epr του SLA που έχει συμφωνήσει. Τέλος καλείται η μέθοδος που έχει οριστεί για αυτόν τον πόρο και τυπώνει στην οθόνη “Hello World!”. Να σημειώσουμε ότι θα μπορούσαν να είχαν οριστεί περισσότερες μέθοδοι και ο χρήστης να μπορεί να επιλέγει ποια ή ποιες θα ήθελε να χρησιμοποιήσει, αλλά αυτό δεν ήταν απαραίτητο στην περίπτωσή μας.

3.4. Χρήση του GRIA στην υλοποίηση

Οι υπηρεσίες που περιγράφηκαν παραπάνω θα αποτελέσουν τους πόρους που θα καταχωρηθούν στο registry και στην συνέχεια θα αναζητηθούν. Συγκεκριμένα:

1) Χρησιμοποιήθηκε η υπηρεσία Διαχείρισης SLA για την δημιουργία φορμών SLA (SLA templates).
2) Χρησιμοποιήθηκε η Υπηρεσία Λογαριασμών για την δημιουργία των λογαριασμών του χρήστη.
3) Χρησιμοποιήθηκαν επίσης η υπηρεσία Δεδομένων και η Υπηρεσία Επεξεργασίας.
4) Δημιουργήθηκε και χρησιμοποιήθηκε η HelloService. Η υπηρεσία αυτή όπως είπαμε και παραπάνω δημιουργεί έναν πόρο, το sample-resource, οποίος είναι γενικός και επιτελεί την απλή λειτουργία να τυπώνει το μήνυμα “Hello World!”. Μπορεί βεβαίως να εξειδικευθεί αλλά στα πλαίσια της διπλωματικής αυτής κάτι τετοιο δεν ήταν απαραίτητο. Η Hello Service απαιτεί την συμφωνία SLA πριν μπορέσει ο χρήστης να την χρησιμοποιήσει.
5) Δημιουργήθηκε επίσης ένας πελάτης με χρήση του API του GRIA ο οποίος δημιουργεί τον λογαριασμό, συμφωνεί το SLA και στη συνέχεια χρησιμοποιεί την υπηρεσία. Ο πελάτης διαδραματίζει ενεργό ρόλο στην υλοποιησή καθώς είναι αυτός που ορίζει τα κριτήρια για την εύρεση υπηρεσιών.

ΚΕΦΑΛΑΙΟ 4

4. Δημιουργία της Contextualized B2B Registry

4.1. Εισαγωγή

Θα χρησιμοποιήσουμε το Contextualised B2B στοιχείο για την δημιουργία της υπηρεσίας που θα παρέχει το μητρώο για την καταχώρηση των πόρων και όπου θα γίνεται η αναζήτηση (Registry Service). Η υπηρεσία αυτή θα επιτρέπει την contextualized αναζήτηση και τον εντοπισμό πόρων με χρήση μεταδεδομένων (xml αρχεία που προσδιορίζουν με κάποιο τρόπο τους πόρους και εξηγούνται παρακάτω αναλυτικά).

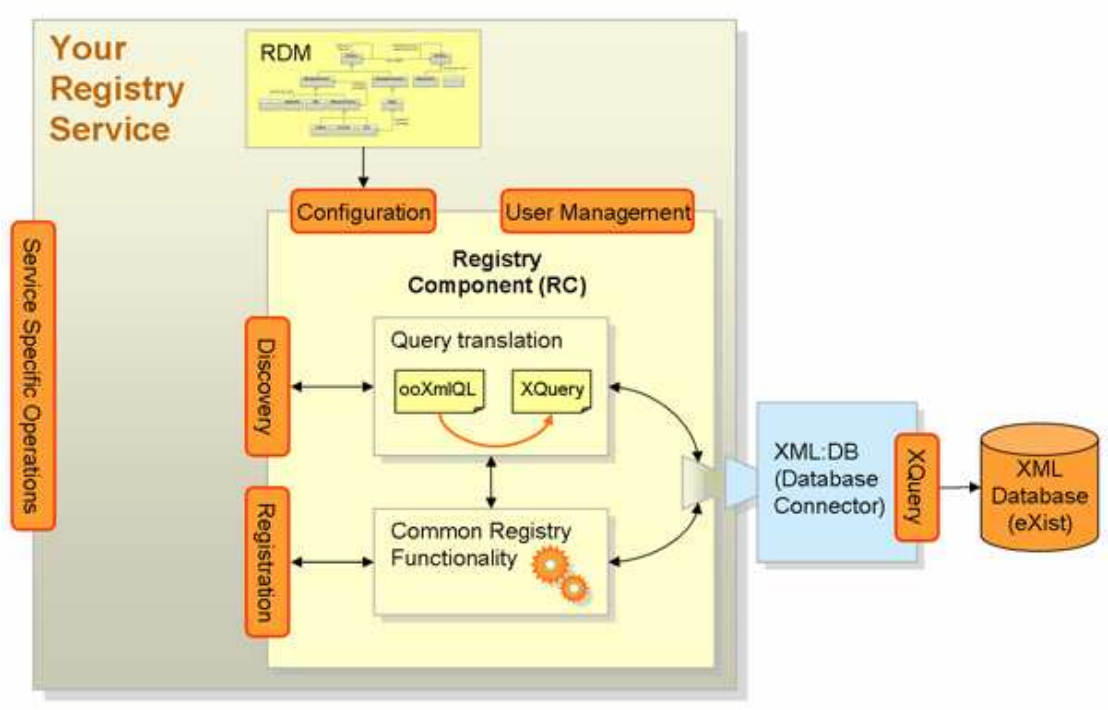
Η πληροφορία περιεχομένου ορίζεται μέσα στο Registry Domain Model (RDM) της Registry Service.

Το Registry Component (RC) παρέχει κοινή λειτουργικότητα η οποία κληρονομείται από όλα τα registries και την λειτουργία μετάφρασης ερωτημάτων που υποστηρίζει την αντιστοίχηση ooXmlQL ερωτημάτων στα αντίστοιχα XQuery ερωτήματα.

ooXmlQL είναι γλώσσα ερωτημάτων που σχεδιάστηκε για να υποστηρίζει και join queries και sub queries που θα βασίζονται στο υποκείμενο RDM όπως εξηγήται και παρακάτω. Η επιλογή και το φιλτράρισμα των statements (where πρόταση) είναι ανεξάρτητα από την γλώσσα που χρησιμοποιείται και μπορούν να οριστούν και με Xpath ή XQuery. Για την εκτέλεση τα ερωτήματα σε ooXmlQL θα μεταφραστούν στην γλώσσα που υποστηρίζει το RDM. Στην συγκεκριμένη υλοποίηση το RDM υποστηρίζει την XQuery.

4.2. Αρχιτεκτονική της Registry

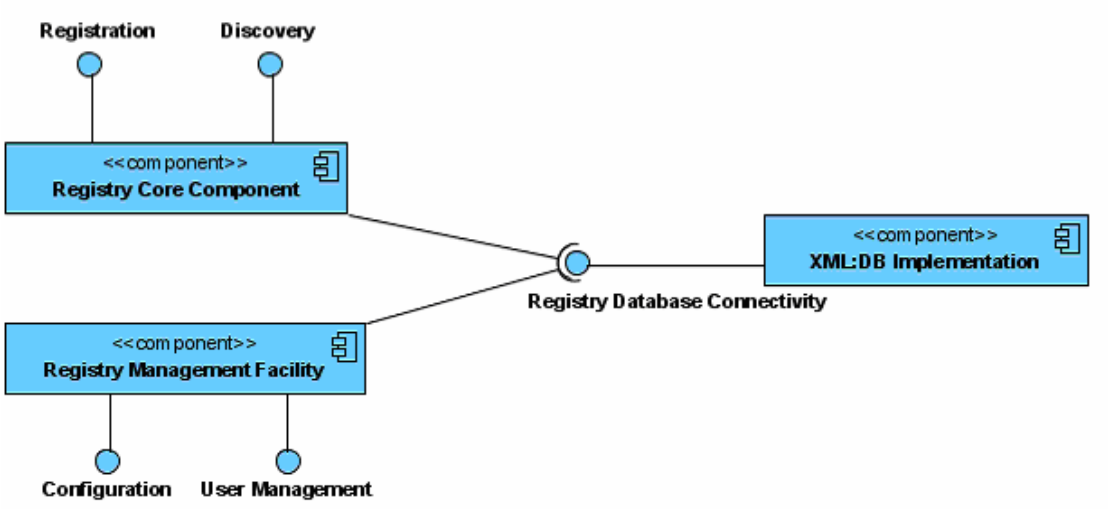
Η αρχιτεκτονική του φαίνεται στο παρακάτω σχήμα:



Εικόνα 5: Αρχιτεκτονική της Registry

Όπως φαίνεται και στο σχήμα το RC αποτελείται από τρία βασικά στρώματα:

- Registry Core Component (RCC)
- Registry Management Facility (RMF)
- Registry Database Connectivity (RDC)



Εικόνα 6: Τα βασικά στρώματα του Registry

Το RDC παρέχει ένα σύνολο από διεπαφές για πρόσβαση και αποθήκευση XML δεδομένων, για διαχείρηση των χρηστών και των κανόνων πρόσβασης στην υποκείμενη βάση δεδομένων.

Η τρέχουσα υλοποίηση στηρίζεται σε ένα XML:DB API και η Registry Service συνδέεται με την βάση δεδομένων exist.

Το RCC περιλαμβάνει διεπαφές για καταχώρηση και εύρεση XML αρχείων, ένα interface για το Registry Domain Model (RDM) καθώς και υλοποίηση αυτής της διεπαφής βασισμένη σε QWL και υλοποίηση της XML-based γλώσσας για δημιουργία ερωτημάτων ooXmlQL.

Το RDM ορίζει όλες τις έννοιες καθώς και τις σχέσεις μεταξύ αυτών που μπορούν να χρησιμοποιηθούν από καποιο registry. Τα XML αρχεία που αποθηκεύονται στο registry, μπορούν στη συνέχεια να ανακτηθούν με βάση αυτές τις έννοιες. Οι σχέσεις καθορίζουν τις εξαρτήσεις μεταξύ αυτών των εννοιών και μπορουν να χρησιμοποιηθούν στον σχηματισμό συνδεδεμένων ερωτήσεων και ημι-ερωτήσεων των βασικών ερωτήσεων για τον εντοπισμό διαφόρων πόρων στο registry. Μια ειδική σχέση is-a-relationship εισάγεται για να οριστούν ιεραρχίες από έννοιες. Οι έννοιες που είναι χαμηλότερα στην ιεραρχία κληρονομούν τις σχέσεις από τις έννοιες που είναι υψηλότερα στην ιεραρχία ως προς αυτά. Ως τώρα το RDM υποστηρίζει μόνο απλή κληρονομικότητα αλλά στο μέλλον θα μπορούσε να υπάρξει και πολλαπλή κληρονομικότητα και περιορισμοί στις σχέσεις που κληρονομούνται.

Το RMF κάνει την διαμόρφωση ολόκληρου του registry. Το τμήμα αυτό παρέχει διεπαφές για διαμόρφωση και για την διαχείρηση των χρηστών. Η διαχείρηση των χρηστών περιλαμβάνει δημιουργία και διαγραφή χρηστών και ομάδων στις οποίες μπορεί να ανήκουν οι χρήστες. Η διεπαφή της διαμόρφωσης επιτρέπει την αρχική ρύθμιση του RDM καθώς και την πρόσθεση νέων οντοτήτων και σχέσεων στη συνέχεια με δυναμικό τρόπο. Υποστηρίζει επίσης και την καταχώρηση προκαθορισμένων ερωτημάτων που μπορούν να εκτελεστούν απλά καλώντας τα με κάποιο όνομα μέσα από τη διεπαφή του RCC για ανακάλυψη πόρων. Στην παρούσα φάση μόνο προκαθορισμένα ερωτήματα σε γλώσσα XQuery μπορούν να αποθηκευτούν και να εκτελεστούν Στο μέλλον θα υποστηρίζονται και ερωτήματα και σε άλλες γλώσσες.

Παρακάτω δίνουμε τις διεπαφές που χρησιμοποιήσαμε στην υλοποίηση:
RCC

Registration

- registerEntity(): εκχώρηση ενός αντικειμένου κάτω από μια συγκεκριμένη οντότητα.
- deleteEntity(): διαγραφή ενός αντικειμένου από μια δοσμένη οντότητα.
- deleteAllEnties: διαγράφει όλα τα αντικείμενα στα οποία έχει πρόσβαση ο χρήστης.
- insertRelationship: δημιουργεί μία σχέση μεταξύ δύο οντοτήτων.
  Προϋπόθεση: η σχέση αυτή να είναι ορισμένη μέσα στο RDM μεταξύ των δύο οντοτήτων στα οποία ανήκουν τα δύο αντικείμενα. Αν η σχέση έχει μία αντίστροφη σχέση (owl:inverseOf) τότε θα προστεθεί αυτόματα και η αντίστροφη σχέση.

Discovery

- getRegistryDomainModel: επιστρέφει το RDM του registry.

- lookup: αναζητα ένα αντικείμενο από την οντότητα στο οποίο ανήκει και από το αναγνωριστικό της.
- lookupByConcept: αναζητά αντικείμενα με βάση μια συγκεκριμένη οντότητα.
- getIdentifier: επιτρέφει το αναγνωριστικό ενός αρχείου/οντότητας.
- query: ρωτά το registry χρησιμοποιώντας κλάση της Java (XmlQuery.java) αντιπροσωπεύοντας ooXmlQL ερωτήματα
- queryBySpecificQL: ρωτά το registry χρησιμποιώντας μία καθορισμένη γλώσσα ερωτημάτων. Αν η συγκεκριμένη γλώσσα δεν υποστηρίζεται από το συγκεκριμένο registry θα προκύψει εξαίρεση.
- executePredefinedQuery: εκτελεί ένα προκαθορισμένο ερώτημα το οποίο είναι αποθηκευμένο στο registry.

RMF

Configuration

- setRegistryDomainModel: αρχικοποιεί το RDM του registry και ρυθμίζει το registry ανάλογα.
- getRegistryDomainModel: επιστρέφει το RDM του registry.
- createConcept: δημιουργεί μια νέα οντότητα μέσα στο RDM και ενημερώνει το registry.
- insertBidirectionalRelationship: προσθέτει μία καινούρια σχέση ανάμεσα σε δύο οντότητες.(χρησιμοποιώντας owl:inverseOf).
- insertRelationship: προσθέτει μία καινούρια σχέση ανάμεσα σε δύο οντότητες.
- registerPredefinedQuery: καταχωρεί ένα καινούριο προκαθορισμένο ερώτημα με ένα συγκεκριμένο όνομα.
- setPredefinedQueries: καταχωρεί ένα σύνολο από προκαθορισμένα ερωτήματα σε XML μορφή προκαθορισμένων ερωτημάτων.

User Management:

- addUser: προσθέτει έναν νέο χρήστη στον πίνακα με τους χρήστες του registry.
- removeUser: διαγράφει έναν χρήστη από τον πίνακα με τους χρήστες του registry.
- addGroupToUser: προσθέτει μία νέα ομάδα στον πίνακα με τις ομάδες στις οποίες ανήκει ένας χρήστης.
- removeGroupFromUser: διαγράφει μία ομάδα από τον πίνακα με τις ομάδες στις οποίες ανήκει ένας χρήστης.
- listUsers: βάζει σε έναν πίνακα όλους τους χρήστες του registry.
- listGroups: βάζει σε έναν πίνακα όλες τις ομάδες του registry.

Διεπαφές που παρέχουν σταθερές για την σύνδεση με την βάση δεδομένων:

- PropertyConstants: γενικά αναγνωριστικά που χρησιμοποιούνται μέσα στο αρχείο registry.properties όπως για παράδειγμα ‘registry.conf.dir’ (package uk.ac.soton.itinnovation.registry.icomponent.util)

- QueryLanguageNS: namespaces των γλωσσών ερωτημάτων (package uk.ac.soton.itinnovation.registry.icomponent.namespace).
- XmlPropertyConstants: αναγνωριστικά που χρησιμοποιούνται μέσα στο αρχείο registry.properties και είναι ειδικά για μια υλοποίηση του registry βασισμένη σε XML (package uk.ac.soton.itinnovation.registry.component.xml.util)
- XmlDatabaseConstants: σταθερές που χρησιμποιούνται εσωτερικά όπως queries.xml για προκαθορισμένα ερωτήματα (uk.ac.soton.itinnovation.registry.component.xml.db).
- XmlPropertyConstantsXmlDb: σταθερές που χρησιμοποιούνται μέσα στην υλοποίηση του XML:DB connector του registry (package uk.ac.soton.itinnovation.registry.component.xml.db.xmldb).

4.3. Υλοποίηση της αναζήτησης

Σκοπός της εργασίας μας είναι η αναζήτηση υπηρεσιών και SLA templates τα οποία παρέχουν το ζητούμενο επίπεδο ποιότητας υπηρεσιών και τα οποία μπορούν να προταθούν από τους πελάτες στον πάροχο. Η αναζήτηση θα γίνεται σε ένα δίκτυο τριών υπολογιστών. Η κλάση αναζήτησης θα κάνει αναζήτηση και στους τρεις υπολογιστές και θα έχει ως έξοδο τα SLA templates που προσφέρουν τα ζητούμενα επίπεδα ποιότητας υπηρεσιών και τις υπηρεσίες που μπορούν να τα παρέχουν. Συγκεκριμένα για τις υπηρεσίες θα δίνεται στην έξοδο το Endpoint reference (epr) τους και τον υπολογιστή στον οποίο ανήκουν, δηλαδή το (epr) τους.

Χρησιμοποιήθηκε το δίκτυο των παρακάτω τριών υπολογιστών:
dimitra.netmode.ntua.gr
niobe.netmode.ntua.gr
argugrid-server.netmode.ntua.gr

Σε κάθε έναν από τους τρεις υπολογιστές έγινε εγκατάσταση ενός registry ενός χώρου δηλαδή όπου θα αποθηκεύονται οι πόροι . Αρχικά κάναμε εγκατάσταση του eXist στον Tomcat. Το eXist μας παρέχει μία βάση δεδομένων όπου και θα αποθηκεύεται το registry [19].

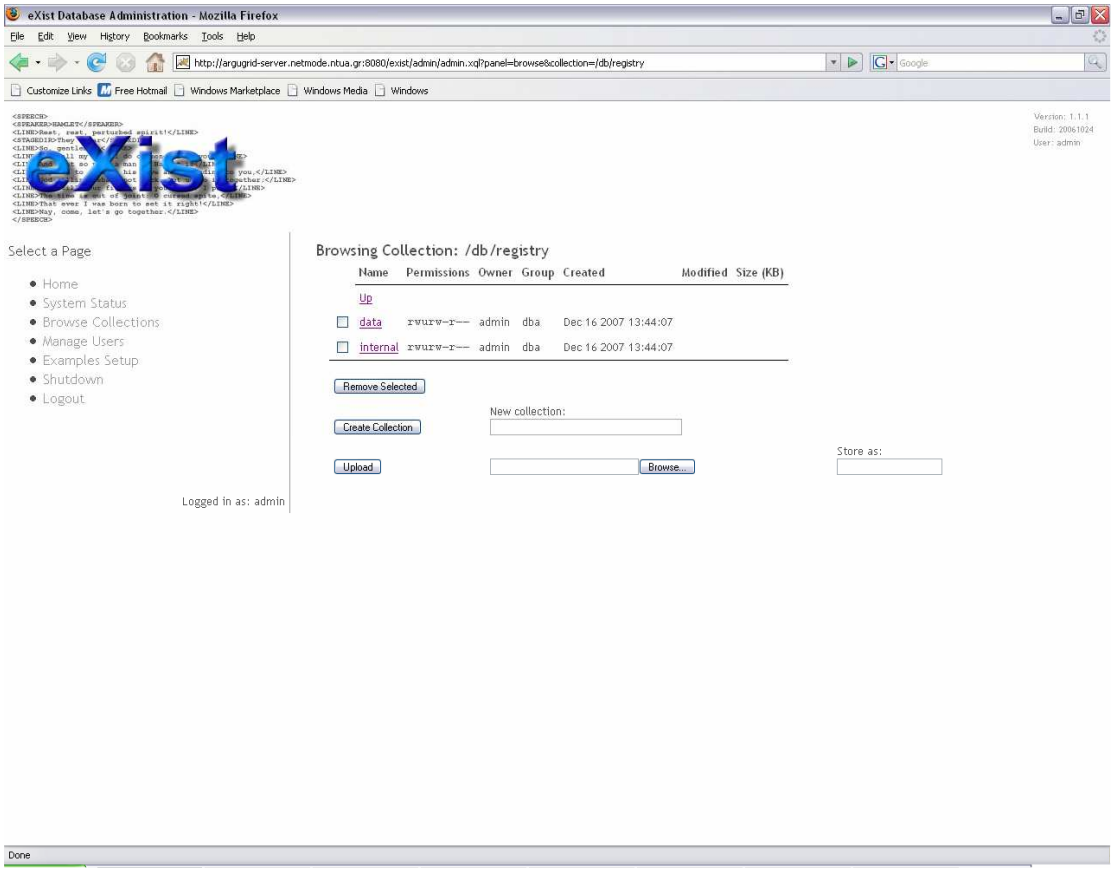
Στην συνέχεια δημιουργούμε ένα αρχείο registry.properties το οποίο περιέχει όλες τις απαραίτητες πληροφορίες για

1) τα στοιχεία του διαχειριστή της βάσης όπως το username του και το password,
2) Το namespace για το RegistryDomainModel (RDM) που χρησιμοποιείται στο registry,
3) την θέση της eXist βάσης δεδομένων δηλαδή το uri της,
4) το όνομα της βάσης στην XML βάση δεδομένων
5) τους κανόνες πρόσβασης στις οντοτήτες και τα αντικείμενα που έχουν καταχωρηθεί στην βάση.

Στο παράρτημα δίνονται τα τρία αρχεία registry.properties τα οποία χρησιμοποιήθηκαν για το registry κάθε υπολογιστή.

Στη συνέχεια δημιουργήθηκε το αρχείο implementationfactory.properties το οποίο δίνεται στο παράρτημα και είναι ίδιο και για τους τρεις υπολογιστές. Στο αρχείο αυτό καθορίζεται η κλάση η οποία υλοποιεί την σύνδεση με την βάση δεδομένων όπου θα αποθηκευτεί το registry. Στην περίπτωση μας αυτό στηρίζεται στο XML:DB που είναι API για XML βάσεις δεδομένων.

Στην συνέχεια δημιουργήθηκε ένας client οποίος πραγματοποιεί την εγκατάσταση της βάσης δεδομένων παίρνοντας το URI της βάσης με την οποία θα συνδεθεί διαβάζοντας το αρχείο registry.properties. Η μορφή της βάσης θα είναι:



Όπως βλέπουμε μέσα στο registry ορίζονται διάφορες συλλογές . Σε αυτήν την φάση οι συλλογές δεν περιέχουν τίποτα. Στην συλλογή data θα αποθηκευτούν οι οντότητες ενώ στην συλλογή internals θα αποθηκευτούν τα απαραίτητα αρχεία για την αρχικοποίηση και διαμόρφωση του registry.

Στην συνέχεια δημιουργείται ένας client ο οποίος θα πραγματοποιήσει τη διαμόρφωση και ρύθμιση του registry. Φορτώνεται στην βάση δεδομένων το RegistrationDomainModel.owl στο οποίο περιγράφονται όλα τα concepts και οι σχέσεις μεταξύ τους. Έτσι εγκαθίστανται στην βάση δεδομένων οι απαιτούμενες οντότητες και οι σχέσεις που τις συνδέουν.

Τώρα πλέον στην βάση θα υπάρχουν στην συλλογή data οι οντότητες οι οποίες ορίζονται στο αρχείο RegistryDomainModel.owl και το οποίο δίνεται στο

παράρτημα. Για την περιγραφή του χρησιμοποιείται η γλώσσα OWL (Web Ontology Language).

Σύντομη περιγραφή:
Στο στοιχείο owl:Class ορίζονται τα concepts που θα χρησιμοποιηθούν στο registry.
Στο στοιχείο rdfs:label ορίζεται το όνομα του concept.
Στο στοιχείο rdfs:subClassOf ορίζονται οι υπο-οντότητες τις οποία κληρονομούν όλες τις σχέσεις της οντότητας-πατέρα.
Στο στοιχείο owl:ObjectProperty ορίζεται μια σχέση μεταξύ δύο οντοτήτων.
Στο στοιχείο rdfs:label ορίζεται το όνομα της σχέσης αυτής.
Στο στοιχείο owl:inverseOf αναφέρεται η αντίστροφη σχέση αυτής που ορίστηκε η οποία βέβαια θα καθοριστεί πάλι παρακάτω σε ένα στοιχείο owl:ObjectProperty.
Στο στοιχείο rdfs:range ορίζεται η οντότητα από το οποίο ξεκινάει η σχέση και
Στο στοιχείο rdfs:domain η οντότητα στην οποία καταλήγει η σχέση.

Οι βασικές οντότητες από τις οποίες θα αποτελείται το registry είναι:
Reference: Καταχώρηση οντοτήτων που θα είναι xml αρχεία στα οποία θα καθορίζονται τα Endpoint References (epr) δηλαδή ουσιαστικά τα uri των διαφόρων υπηρεσιών που θα βρίσκονται στο registry.
Referencable: Όλες οι υπηρεσίες και οι πόροι που έχουν κάποιο epr.

Για να ορίσουμε δικές μας οντότητες καθώς και τις αντίστοιχες σχέσεις μεταξύ τους μπορούμε να κάνουμε δυναμική διαμόρφωση του RDM. Για αυτόν τον σκοπό δημιουργήσαμε έναν client στον οποίο με την βοήθεια της μεθόδου createConcept προσθέσαμε οντότητες και με την μέθοδο createBidrectionalRelationships τις αντίστοιχες σχέσεις μεταξύ τους.

Στην υλοποίηση μας θέλουμε να γίνεται αναζήτηση sla Templates με βάση κάποια κριτήρια που θα δίνει ο χρήστης και τα οποία αναλύονται περισσότερο παρακάτω καθώς και των υπηρεσιών που μπορούν να γίνουν διαχειριστούν από αυτά τα SLA templates. Για αυτόν τον σκοπό προστέθηκαν οι εξής έννοιες:

Έννοιες:

- SlaTemplate: εδώ θα είναι οι οντότητες που αποτελούν τα xml αρχεία που θα είναι sla templates
- DataService: wsdl αρχεία για υπηρεσίες της πλατφόρμας GRIA που προσφέρουν την δημιουργία πόρων τύπου data stagers δηλαδή αποθηκευτικών χώρων για την φόρτωση αρχείων ,εικόνων τα οποία μπορούν αργότερα να επεξεργαστούν από διάφορες εφαρμογές.

```
 if (!configuration.getRegistryDomainModel().containsConcept(
      "DataService")) {
// add concept as a sub concept of concept 'Reference'
configuration.createConcept("DataService", "ManagedResource");
 }
```

- JobService : wsdl αρχεία για υπηρεσίες της GRIA πλατφόρμας που παρέχουν πόρους (resources) jobs δηλαδή εφαρμογές (applications) που

δέχονται ως είσοδο κάποια δεδομένα τα επεξεργάζονται και δίνουν ως έξοδο κάποιο αποτέλεσμα.

```
if (!configuration.getRegistryDomainModel().containsConcept(
     "JobService")) {
 configuration.createConcept("JobService", "ManagedResource");
  }
```

- SlaManagedService: Concept που αντιπροσωπεύει τις υπηρεσίες τις οποίες μπορεί να διαχειριστεί κάποιο SLA και δεν ανήκουν στην κατηγορία των υπηρεσιών δεδομένων ή επεξεργασίας.

```
if (!configuration.getRegistryDomainModel().containsConcept(
     "SlaManagedService")) {
 configuration.createConcept("SlaManagedService",
 "ManagedResource");
   }
```

- Τα concepts RemoteJobService, RemoteDataService, RemoteHelloService Για την καταχώρηση xml αρχείων που περιγράφουν το epr των υπηρεσιών Job Service, Data Service, Hello Service.

```
String[] billingConcepts = {
"RemoteSLAService","RemoteDataService","RemoteJobService","RemoteHell
oService"};

for (String concept : billingConcepts) {
      if
(!configuration.getRegistryDomainModel().containsConcept(concept)) {
// add concept as a sub concept of concept 'Reference'
      configuration.createConcept(concept,"Reference");
                  }
}
```

Σχέσεις:

- Canbeamanaged: μεταξύ του SlaManagedService, DataService, JobService και του SlaTemplate
- Canmanage: αντίστροφη της παραπάνω σχέσης μεταξύ του SlaTemplate και του DataService, JobService, SlaManagedServices.

```
String nextfromConcept = "DataService";
String nexttoConcept = "SlaTemplate";
if (!configuration.getRegistryDomainModel().containsRelationship(
nextfromConcept, "canbemanaged", nexttoConcept)) {
System.out.println("add relationship...");
configuration.insertBidirectionalRelationship(nextfromConcept,
"canbemanaged","canmanage", nexttoConcept);
                  }
```

```
String nextfromConcept1 = "JobService";
String nexttoConcept1 = "SlaTemplate";

if (!configuration.getRegistryDomainModel().containsRelationship(
      nextfromConcept1, "canbemanaged", nexttoConcept1)) {
      System.out.println("add relationship...");
      configuration.insertBidirectionalRelationship(nextfromConcept1,
"canbemanaged","canmanage", nexttoConcept1);
                  }

String fromCon= "SlaManagedService";
String toCon= "SlaTemplate";

if (!configuration.getRegistryDomainModel().containsRelationship(
      fromCon, "canbemanaged", toCon)) {
System.out.println("add relationship...");
            configuration.insertBidirectionalRelationship(fromCon,
"canbemanaged","canmanage", toCon);
                  }
```

- has: σχέση μεταξύ των οντοτήτων DataService, JobService, SlaManagedService και των RemoteDataService, RemoteJobservice, RemoteHelloService που είναι υποοντότητες της Reference και όπου καταχωρούνται όλα τα xml αρχεία που παριστάνουν τα epr των υπηρεσιών. Η σχέση αυτή έχει οριστεί στην βασική ρύθμιση του RDM μοντέλου.

Η εξοδος της κλάσης είναι το καινούριο RDM το οποίο και παρουσιάζεται στο παράρτημα καθώς και ένα μήνυμα που ειδοποιεί ότι η ρύθμιση του RDM έγινε επιτυχώς.

Καταχώρηση των αρχείων (Registration)

Στην συνέχεια δημιουργήσαμε τον client που υλοποιεί την καταχώρηση των πόρων ως αρχεία xml στο registry . Δημιουργούμε ένα factory για το instantiate του registry.

```
RegistryFactory factory =
StartupRegistryFactory.createRegistryFactory(SupportedTechnology.XML)
;
```

Χρησιμοποιώντας την μέθοδο createRegistrationFacility() δημιουργείται το αντικείμενο registration τύπου Registration του οποίου χρησιμοποιούμε τις μεθόδους για την καταχώρηση των αρχείων.

```
Registration registration = factory.createRegistrationFacility();
```

Τα xml αρχεία ανήκουν σε κάποιο μονοπάτι (path) στον υπολογιστή στον οποίο ανήκουν. Ο client διαβάζει αυτό το μονοπάτι όπως για παράδειγμα:

```
classLoader.getResource("resources/dimitra/cpu/slaTemplate.xml").toURI()
classLoader.getResource("resources/dimitra/JobService.wsdl").toURI()
```

και καταχωρεί τα αρχεία στο registry χρησιμοποιώντας την κλάση registerEntity όπως για παράδειγμα εδώ όπου καταχωρείται το προηγούμενο SLA template υπό την οντότητα SlaTemplate και το JobService.wsdl υπό την οντότητα JobService.

```
String slaTemplate1 = registration.registerEntity("SlaTemplate",
uris[0], null);
String jobService = registration.registerEntity("JobService",
uris[2], "JobService.wsdl");
```

Δημιουργούνται οι απαιτούμενες σχέσεις μεταξύ των καταχωρημένων πόρων, όπως ορίστηκαν στο RDM όπως για παράδειγμα η δημιουργία της σχέσης canbemanaged μεταξύ του JobService.wsdl και του slaTemplate.xml που δόθηκε και παραπάνω:

```
registration.insertRelationship(jobService,"canbemanaged",
slaTemplate1);
```

Κάθε πόρος που καταχωρείται στο registry αντιστοιχίζεται σε κάποιο μοναδικό αναγνωριστικό. Αυτό το αναγνωριστικό απαιτείται για να δημιουργηθούν οι σχέσεις μεταξύ των πόρων.

Για την υλοποίηση της αναζήτησης καταχωρήσαμε τους εξής πόρους στους υπολογιστές:

Στον υπολογιστή dimitra.netmode.ntua.gr καταχωρήσαμε τις υπηρεσίες DataService.wsdl, JobService.wsdl και την HelloService.wsdl οι οποίες παρέχονται από αυτόν τον υπολογιστή , καθώς και τα xml αρχεία που περιγράφουν το epr τους δηλαδή το url τους και τα οποία είναι αντίστοιχα τα dataServiceEPR.xml, jobServiceEPR.xml, helloServiceEPR.xml. Προστέθηκαν επίσης τα sla templates τα οποία παρέχονται από την SLA Service αυτού του υπολογιστή και είναι κατάλληλα για αυτές τις υπηρεσίες:

Σε κάθε SLA template η περίοδος χρέωσης είναι μία ημέρα. Αυτό σημαίνει ότι οι περιορισμοί αναφέρονται σε αυτό το χρονικό διάστημα και ότι η χρέωση υπολογίζεται για αυτό το χρονικό διάστημα.

SLA templates:

1) Επιτρέπει την δημιουργία μέχρι πέντε job και η δημιουργία κάθε job χρεώνεται με 1.10 ευρώ
2) Επιτρέπει την δημιουργία μέχρι τρία job και η δημιουργία κάθε job χρεώνεται με 3 ευρώ.
3) Επιτρέπει την χρήση μέχρι 1000 CPU seconds και χρεώνεται 0,01 ευρώ για κάθε δευτερόλεπτο.
4) Επιτρέπει την χρήση μέχρι 1000 CPU seconds και χρεώνεται 0,05 ευρώ για κάθε δευτερόλεπτο.
5) Επιτρέπει την χρήση μέχρι 2000 CPU seconds και χρεώνεται 0,03 ευρώ για κάθε δευτερόλεπτο.
6) Επιτρέπει την δημιουργία μέχρι δύο data stagers και η δημιουργία κάθε νέου data stager χρεώνεται με 10 ευρώ.

7) Επιτρέπει την δημιουργία μέχρι 5 sample resources ανά ημέρα και η δημιουργία κάθε νεόυ resource χρεώνεται με 0,05 ευρώ ανά ημέρα.
8) Επιτρέπει την δημιουργία μέχρι 5 sample resources ανά ημέρα και η δημιουργία κάθε νεόυ resource χρεώνεται με 0,01 ευρώ ανά ημέρα.

Η JobService.wsdl μπορεί να διαχειριστεί από τα sla templates 1-5 οπότε συνδέονται οι αντίστοιχοι πόροι με την σχέση canbemanaged ενω η DataService.wsdl μόνο με το sla template 6. Η HelloService.wsdl μπορεί να διαχειριστεί μόνο από τα sla templates 7 έως 8 και συνδέεται μόνο με αυτά με την σχέση canbemanaged.

Στον υπολογιστή niobe.netmode.ntua.gr εγκαταστήσαμε τα wsdl αρχεία για την DataService η οποία παρέχεται από αυτόν τον υπολογιστή καθώς και το xml αρχείο που περιγράφει το epr της και έχει και κάποια μεταδεδομένα για την κλάση που χρησιμοποιείται για την υλοποίηση της υπηρεσίας στην πλευρά του client, υπό την οντότητα RemoteDataService. Το αρχείο αυτό συνδέεται με την DataService.wsdl με την σχέση has. Θεωρούμε ότι η Hello Service δεν παρέχεται από τον niobe.
Επίσης προσθέσαμε τα sla templates τα οποία περιγράφουν τα παρακάτω:

SLA templates:

1) Επιτρέπει την δημιουργία μέχρι δύο data stagers και η δημιουργία κάθε νέου data stager χρεώνεται με 0,10 ευρώ.
2) Επιτρέπει την δημιουργία μέχρι ενός data stager και η δημιουργία κάθε νέου data stager χρεώνεται με 0,10 ευρώ.
3) Επιτρέπει την δημιουργία μέχρι δύο data stagers και η δημιουργία κάθε νέου data stager χρεώνεται με 0,20 ευρώ.
4) Επιτρέπει την χρήση μέχρι 20971520 bytes από τον δίσκο κάθε μέρα. Η χρήση του δεν χρεώνεται.

Όλα τα sla templates συνδέονται με την DataService.wsdl με την σχέση canmanage.

Στον υπολογιστή argugrid-server.netmode.ntua.gr προσθέσαμε την DataService.wsdl και την JobService.wsdl και τα xml αρχεία τα οποία παρέχουν το epr τους και τα οποία είναι αντίστοιχα τα dataServiceEPR.xml, jobServiceEPR.xml. Προσθέσαμε επίσης τα Sla Templates τα οποία προσφέρουν τα εξής:

1) Επιτρέπει την δημιουργία μέχρι 3 job ανά ημέρα και η χρέωση κάθε job ανά ημέρα είναι 1.10 ευρώ.
2) Επιτρέπει την χρήση μέχρι 1000 CPU δευτερολέπτων και κάθε δευτερόλεπτο που χρησιμοποιείται χρεώνεται με 0.02 ευρώ.
3) Επιτρέπει την χρήση μέχρι 5 data stagers ανά ημέρα και η δημιουργία κάθε νέου data stager χρεώνεται με 0.30 ευρώ.

Η JobService.wsdl συνδέεται με την σχέση “canbemanaged” με τα sla templates (1) και (2) ενώ η DataService.wsdl με το template (3). Θεωρούμε ότι η Hello Service δεν παρέχεται από τον argugrid-server.

Τους παραπάνω clients μπορεί να τους τρέξει μόνο ο διαχειριστής του registry και της βάσης δεδομένων. Ο περιορισμός αυτός γίνεται μέσω ενός κωδικού που πρέπει να δώσει ο διαχειριστής.

Στην συνέχεια υλοποιήθηκε η κλάση DiscoveryClient για την υλοποίηση της αναζήτησης. Όταν κάποιος χρήστης ή και ο διαχειριστής της βάσης θέλει να κάνει κάποια αναζήτηση σε αυτήν τρέχει αυτόν τον client. Δημιουργούμε πάλι ένα factory για το instantiation της κλάσης Discovery και δημιουργούμε το αντικείμενο discovery του οποίου οι μέθοδοι χρησιμποιούνται για την αναζήτηση στο registry.

```
RegistryFactory<XMLResource> factory = (RegistryFactory<XMLResource>)
StartupRegistryFactory.createRegistryFactory(SupportedTechnology.XML)
;
Discovery<XMLResource> discovery =
factory.createRegistryDiscoveryFacility();
```

Γενικά μπορούν να υλοποιηθούν τα παρακάτω είδη αναζήτησης:

- Αναζήτηση με βάση το αναγνωριστικό του πόρου.
- Αναζήτηση με βάση κάποιο προκαθορισμένο ερώτημα το οποίο θα έχει εισαχθεί στην βάση κατά την διάρκεια της διαμόρφωσης της (configuration) .
- Αναζήτηση με χρήση των queries σε ooXmlQL.

Στην συγκεκριμένη υλοποίηση χρησιμοποιήθηκαν μόνο ooXmlQL ερωτήματα επειδή παρείχαν μεγαλύτερη ευελιξία στην υποβολή περιορισμών ανάλογα με τα κριτήρια με βάση τα οποία θέλει ο χρήστης να κάνει την αναζήτηση.

ΚΕΦΑΛΑΙΟ 5

5. Παραδείγματα και αποτελέσματα

5.1. Ερωτήματα

Ο χρήστης μπορεί να δώσει σε μορφή αρχείου το ερώτημα που θέλει να θέσει στην βάση δεδομένων και το αποτέλεσμα της αναζήτησης παρέχεται πάλι σε ένα αρχείο εξόδου. Οι τοποθεσίες και των δύο αρχείων καθορίζονται από τον χρήστη, όπως για παράδειγμα φαίνεται παρακάτω:

```
Please give the path of the file with the query:
C:\dimitra\query1.txt
Please give the path of the output file:

C:\dimitra\query1_out.txt
...
```

Ερώτημα 1

Αναζήτηση του template το οποίο επιτρέπει την δημιουργία μέχρι τριών jobs ανά ημέρα και χρεώνει τον χρήστη για την δημιουργία κάθε νέου job 3 ευρώ την ημέρα, καθώς και το epr της Job Service η οποία μπορεί να διαχειριστεί από αυτό το template.

```
Please give the path of the file with the query:
C:\dimitra\query1.txt

Please give the path of the output file:
C:\dimitra\query1_out.txt
```

Τα αποτελέσματα της αναζήτησης και στους τρεις υπολογιστές θα είναι:

```
The given query is:

<query>
  <select>$template $epr $refable</select>
   <declare name="ns1">http://it-
   innovation.soton.ac.uk/2005/grid</declare>
   <declare
   name="addressing">http://www.w3.org/2005/08/addressing</declare>\
      <from as="$epr"> <class name="Reference"/>
       <join on="holdBy" as="$refAble" class="JobService">
         <join on="canbemanaged" as="$template" class="SlaTemplate"/>
       </join>
      </from>\n
      <where>
      $template//constraints//constraint//metric//description="job
      and $template//constraints//constraint//bound="LE" and
      $template//constraints//constraint//limit="3" and
      $template//pricingTerms//pricingTerm//price="3.00"
      </where>
  </query>
```

```
Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:

0.ID: [bab961b8.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
 <Address>https://dimitra:8443/gria-basic-app-
     services/services/JobService
  </Address>
 <Metadata>
   <ns2:type xmlns:ns2="http://it-
    innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.
    client.RemoteJobService
   </ns2:type>
   <ns3:default xmlns:ns3="http://it-
     innovation.soton.ac.uk/2005/grid">https://dimitra:8443/gria-
     basic-app-services/services/JobService#13e68db1-16afeee9-0116
     -aff40458-0002
    </ns3:default>
   </Metadata>
</EndpointReference>

 1.ID: [6d5f163d.xml]
 <slaTemplate>
    <label>SLA template for Job Service</label>
    <description>SLA for Job Service limited to 3 new jobs per day
      and 3.00 EUR per job
    </description>

<!-- The billing period defines how often the aggregated usage will
be charged to the user's account
-->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

<!-- The signing fee defines how much the user will be charged
     upfront on agreeing the SLA
-->
<!-- The subscription fee defines how much the user will be charged
     at the end of each billing period
-->
    <signingFee>10.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
<!-- The currency applies to all the prices listed in the SLA
     template
  -->
    <currency>EUR</currency>
<!-- The startTime and endTime define the validity period of the SLA
     template
 -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
```

```
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

<!-- permittedServices is a list of which services the client may use
     with this SLA.
 -->
  <permittedServices>
     <permittedService>
       <url>https://dimitra:8443/gria-basic-app-
              services/services/JobService
       </url>
      </permittedService>
    </permittedServices>

<!-- constraints is a mandatory list of constraint elements.-->
    <constraints>
<!-- This constraint limits the user to using up to 3 jobs any one
      time -->
        <constraint type="INSTANTANEOUS">
            <metric type="ACTIVITY">
               <uri>http://www.gria.org/sla/metric/activity/job</uri>
                <description>
                    <description>job</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>job</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>3</limit>
            <contention>1.0</contention>
            <repeating>false</repeating>
        </constraint>
    </constraints>
    <pricingTerms>
      <!-- Every time a job starts, the instantaneous measurement of
       the number of jobs goes up by one (and then down by one when
       it finishes). A pricingTerm of type INSTANTANEOUS_INCREASE
       calculates the increase of the instantaneous measurement over
       the billing period, ignoring decreases.  The instantaneous
       increase is then multiplied by the price accordingly. -->

        <pricingTerm type="INSTANTANEOUS_INCREASE">
            <description>creation charge</description>
            <lowerBound>0</lowerBound>
            <upperBound>-1</upperBound>
            <price>3.00</price>
            <metric type="ACTIVITY">
               <uri>http://www.gria.org/sla/metric/activity/job</uri>
                <description>
                    <description>job</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>job</instantaneous>
                </units>
            </metric>
```

```
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

Results Found
Number of results is: 2

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0

 Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc


Outputs:
No results found!
Number of results is: 0
```

Στην έξοδο αρχικά εμφανίζεται το id ου ζητούμενου πόρου. Το id χαρακτηρίζει τον πόρο στο registry όπου είναι αποθηκευμένο, είναι μοναδικό και αντιστοιχίζεται στον πόρο κατά την αποθήκευσή του στο registry. Στην συνέχεια εμφανίζεται το περιεχόμενό του. Εμφανίζεται το epr της ζητούμενης υπηρεσίας, εδώ της ζητούμενης Job Service, και αποκάτω το ζητούμενο SLA template. Έχουν τονιστεί τα στοιχεία που ικανοποιούν τους περιορισμούς του ερωτήματος. Στο ερώτημα 1 παρατηρούμε ότι το ζητούμενο SLA template παρέχεται μόνο στον host dimitra.netmode.ntua.gr άρα ο χρήστης θα πρέπει να χρησιμοποιήσει την Job Service που παρέχεται από αυτόν τον πάροχο και να συμφωνήσει το συγκεκριμένο SLA template με την SLA υπηρεσία αυτού του παρόχου.

Ερώτημα 2

Αναζήτηση του template το οποίο επιτρέπει την δημιουργία μέχρι δύο data stagers την ημέρα και χρεώνει 0.10 ευρώ για την δημιουργία κάθε νέου data stager και του epr της Data Service την οποία μπορεί να διαχειριστεί αυτό το template.

Τα αποτελέσματα της αναζήτησης είναι:

```
The given query is:
<query>
 <select>$template $epr</select>
   <declare name="ns1">http://it-
   innovation.soton.ac.uk/2005/grid</declare>
   <declare
   name="addressing">http://www.w3.org/2005/08/addressing</declare>
   <from as="$epr"> <class name="Reference"/>
     <join on="holdBy" as="$refAble" class="DataService">
       <join on="canbemanaged" as="$template" class="SlaTemplate"/>
     </join>
   </from>
```

```
   <where>
   $template//constraints//constraint//metric//description="data
   stager" and $template//constraints//constraint//bound="LE"
   and $template//constraints//constraint//limit="2" and
       $template//pricingTerms//pricingTerm//price="0.10"
   </where>
</query>

Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:

0.ID: [abfba779.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
   <Address>https://dimitra:8443/gria-basic-app-
        services/services/DataService
   </Address>
   <Metadata>
     <ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.
client.RemoteDataService
    </ns2:type>
    <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">https://dimitra:8443/gria-basic-
app-services/services/DataService#13e68db1-16afeee9-0116-aff40458-
0003
    </ns3:default>
  </Metadata>
</EndpointReference>

 1.ID: [2181dce4.xml]
 <slaTemplate>
    <label>SLA Template for Data Service</label>
    <description>
      SLA Template for Data Service Limited to 2 data stagers and
      0.10 EUR per data stager
</description>
<!-- The billing period is mandatory and defines how often the
aggregated usage will be charged to the user's account -->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

<!-- The signing fee is mandatory and defines how much the user will
be charged upfront on agreeing the SLA -->
<!-- The subscription fee is mandatory and defines how much the user
will be charged at the end of each billing period -->
    <signingFee>20.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
      <!-- The currency applies to all the prices listed in the SLA
        template -->
    <currency>EUR</currency>
 <!-- The startTime and endTime define the validity period of the SLA
       template -->
    <startTime>
        <year>2000</year>
```

```
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

      <!-- permittedServices is a list of which services the client
       may use with this SLA. -->
    <permittedServices>
       <permittedService>
         <url>https://dimitra:8443/gria-basic-app-
             services/services/DataService
          </url>
        </permittedService>
    </permittedServices>

<!-- constraints is a mandatory list of constraint elements  -->
    <constraints>
            <!-- This constraint limits the user to using up to 2
                 data-stagers any one time -->
        <constraint type="INSTANTANEOUS">
            <metric type="RESOURCE">
              <uri>http://www.gria.org/sla/metric/activity/data-
                   stager
              </uri>
                <description>
                    <description>data stager</description>
                    <instantaneous>data-stager</instantaneous>
                </description>
                <units type="DECIMAL">
                    <instantaneous>DATA-STAGER</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>2</limit>
            <contention>1.0</contention>
            <repeating>false</repeating>
        </constraint>
    </constraints>
    <pricingTerms>
<!-- This pricing term specifies that there is a charge of 0.10 EUR
every time a data-stager is created -->
        <pricingTerm type="INSTANTANEOUS_INCREASE">
            <description>creation charge</description>
            <lowerBound>0</lowerBound>
            <upperBound>-1</upperBound>
            <price>0.10</price>
            <metric type="ACTIVITY">
                 <uri>http://www.gria.org/sla/metric/activity/data-
                  stager
                 </uri>
                   <description>
                      <description>data stager</description>
                    </description>
                   <units type="DECIMAL">
                     <instantaneous>data-stager</instantaneous>
                   </units>
```

```
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

Results Found
Number of results is: 2

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0

Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
0.ID: [bdf029b.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
   <Address>https://niobe:8443/gria-basic-app-
      services/services/DataService</Address>
    <Metadata>
       <ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.clie
nt.RemoteDataService</ns2:type>
        <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">https://niobe:8443/gria-basic-app-
services/services/DataService#13e68db1-16afeee9-0116-aff40458-
0003</ns3:default>
    </Metadata>
</EndpointReference>



1.ID: [ea345c28.xml]
 <slaTemplate>
    <label>SLA Template for Data Service</label>
    <description>SLA Template for Data Service Limited to 2 data
     stagers and 0.10 EUR per data stager.</description>

      <!-- The billing period is mandatory and defines how often the
       aggregated usage will be charged to the user's account -->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

<!-- The signing fee is mandatory and defines how much the user will
be charged upfront on agreeing the SLA -->
<!-- The subscription fee is mandatory and defines how much the user
will be charged at the end of each billing period -->
    <signingFee>20.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
<!-- The currency applies to all the prices listed in the SLA
template -->
    <currency>EUR</currency>
```

```
<!-- The startTime and endTime define the validity period of the SLA
template -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

<!-- permittedServices is a list of which services the client may use
with this SLA-->
    <permittedServices>
        <permittedService>
           <url>https://niobe:8443/gria-basic-app-
              services/services/DataService</url>
        </permittedService>
    </permittedServices>

<!-- constraints is a mandatory list of constraint elements. -->
    <constraints>
      <!-- This constraint limits the user to using up to 2
           data-stagers any one time -->
        <constraint type="INSTANTANEOUS">
          <metric type="RESOURCE">
            <uri>http://www.gria.org/sla/metric/activity/
                 data-stager
            </uri>
                <description>
                    <description>data stager</description>
                    <instantaneous>data-stager</instantaneous>
                </description>
                <units type="DECIMAL">
                    <instantaneous>DATA-STAGER</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>2</limit>
            <contention>1.0</contention>
            <repeating>false</repeating>
        </constraint>
    </constraints>
    <pricingTerms>
<!-- This pricing term specifies that there is a charge of 0.10 EUR
every time a data-stager is created -->
        <pricingTerm type="INSTANTANEOUS_INCREASE">
            <description>creation charge</description>
            <lowerBound>0</lowerBound>
            <upperBound>-1</upperBound>
            <price>0.10</price>
            <metric type="ACTIVITY">
              <uri>http://www.gria.org/sla/metric/activity/
                   data-stager
               </uri>
                <description>
                    <description>data stager</description>
```

```
                </description>
                <units type="DECIMAL">
                    <instantaneous>data-stager</instantaneous>
                </units>
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

Results Found
Number of results is: 2
```

Στο ερώτημα 2 παρατηρούμε ότι το ζητούμενο SLA template παρέχεται και από τον host dimitra.netmode.ntua.gr αλλά και από τον host niobe.netmode.ntua.gr οπότε ο χρήστης μπορεί να επιλέξει όποιον υπολογιστή επιθυμεί για να χρησιμοποιήσει την αντίστοιχη Data Service.

Ερώτημα 4

Αναζήτηση του tempate το οποίο επιτρέπει την χρήση μέχρι 1000 CPU seconds ανά ημέρα και χρεώνει 0.05 ευρώ το κάθε second, καθώς και το epr της Job Service την οποία μπορεί να διαχειριστεί αυτό το template.

Τα αποτελέσματα της αναζήτησης είναι:

```
The given query is:
<query>
<select>$template $epr</select>
<declare name="ns1">http://it-
innovation.soton.ac.uk/2005/grid</declare>
<declare
name="addressing">http://www.w3.org/2005/08/addressing</declare>
<from as="$epr">
 <class name="Reference"/>
 <join on="holdBy" as="$refAble" class="JobService">
<join on="canbemanaged" as="$template" class="SlaTemplate"/>
</join>
</from>
<where>
$template//constraints//constraint//metric//description="CPU" and
$template//constraints//constraint//limit="1000.0" and
$template//pricingTerms//pricingTerm//price="0.05"
</where>
</query>


Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:

0.ID: [bab961b8.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
    <Address>https://dimitra:8443/gria-basic-app-
      services/services/JobService</Address>
    <Metadata>
```

```
      <ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.clie
nt.RemoteJobService</ns2:type>
      <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">https://dimitra:8443/gria-basic-
app-services/services/JobService#13e68db1-16afeee9-0116-aff40458-
0002</ns3:default>
    </Metadata>
</EndpointReference>

 1.ID: [ed6f7e1c.xml]
<slaTemplate>
    <label>Sla Template for Job Service</label>
    <description>Limited to 1000 CPU.s/day and 0.05 EUR/CPU
      second</description>
      <!-- The billing period defines how often the aggregated usage
        will be charged to the user's account -->
      <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>
        <!-- The signing fee defines how much the user will be
        charged upfront on agreeing the SLA -->
      <!-- The subscription fee defines how much the user will be
        charged at the end of each billing period -->
    <signingFee>20.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
      <!-- The currency applies to all the prices listed in the SLA
       template -->
    <currency>EUR</currency>

<!-- The startTime and endTime define the validity period of the SLA
template -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

<!-- permittedServices is a list of which services the client may use
with this SLA.-->
    <permittedServices>
        <permittedService>
         <url>https://dimitra:8443/gria-basic-app-
services/services/JobService</url>
        </permittedService>
    </permittedServices>

      <!-- constraints is a mandatory list of constraint elements.
       It may be empty.  -->
    <constraints>
```

```
<!-- This constraint limits the user to using at most 1000
CPU.seconds each day -->
        <constraint type="CUMULATIVE">
            <metric type="RESOURCE">
               <uri>http://www.gria.org/sla/metric/resource/cpu</uri>
                <description>
                    <description>CPU</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>CPU</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>1000.0</limit>
            <contention>1.0</contention>
            <!-- This constraint is repeating.  It constrains the
            usage over a period of time defined by the duration -->
            <repeating>true</repeating>
      <!-- The duration of a repeating constraint would normally be
       the same as the billing period -->
            <duration>
                <years>0</years>
                <months>0</months>
                <days>1</days>
                <hours>0</hours>
                <minutes>0</minutes>
                <seconds>0</seconds>
            </duration>
        </constraint>
    </constraints>
    <pricingTerms>
<!-- A pricing term may be either of type CUMULATIVE or
INSTANTANEOUS_INCREASE -->
<!-- This pricing terms specifies that CPU usage is charged at 0.05
EUR per second -->
        <pricingTerm type="CUMULATIVE">
            <description>standard rate</description>
            <lowerBound>0</lowerBound>
                  <!-- An upperBound of -1 means +infinity -->
            <upperBound>-1</upperBound>
            <price>0.05</price>
            <metric type="RESOURCE">
               <uri>http://www.gria.org/sla/metric/resource/cpu</uri>
                <description>
                    <description>CPU</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>CPU</instantaneous>
                </units>
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

Results Found
Number of results is: 2

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc
Outputs:
```

```
No results found!
Number of results is: 0

 Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0
```

Παρατηρούμε ότι και σε αυτό το ερώτημα όπως και στο πρώτο το ζητούμενο template το έχει μόνο ο host dimitra.netmode.ntua.gr.

Ερώτημα 5

Αναζήτηση του tempate το οποίο επιτρέπει την χρήση μέχρι 1000 CPU seconds ανά ημέρα και χρεώνει 0.02 ευρώ το κάθε second, καθώς και το epr της Job Service την οποία μπορεί να διαχειριστεί αυτό το template.

```
The given query is:
 <query>
   <select>$template $epr</select>
   <declare name="ns1">http://it-
      innovation.soton.ac.uk/2005/grid</declare>
   <declare>
     name="addressing">http://www.w3.org/2005/08/addressing
   </declare>
   <from as="$epr">
    <class name="Reference"/>
      <join on="holdBy" as="$refAble" class="JobService">
          <join on="canbemanaged" as="$template"
              class="SlaTemplate"/>
     </join>
  </from>
  <where>
   $template//constraints//constraint//metric//description="CPU" and
   $template//constraints//constraint//limit="1000.0" and
    $template//pricingTerms//pricingTerm//price="0.02"
  </where>
</query>

Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
0.ID: [9494301b.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
    <Address>https://argugrid-server.netmode.ece.ntua.gr:8443/gria-
basic-app-services/services/JobService</Address>
    <Metadata>
        <ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.clie
nt.RemoteJobService</ns2:type>
```

```
        <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">https://argugrid-
server.netmode.ece.ntua.gr:8443/gria-basic-app-
services/services/JobService#13e68db1-16afeee9-0116-aff40458-
0002</ns3:default>
    </Metadata>
</EndpointReference>

 1.ID: [3e364084.xml]
<slaTemplate>
    <label>Sla Template for Job Service</label>
    <description>Limited to  1000 CPU.s/day and 0.02 EUR per
     job</description>
<!-- The billing period is mandatory and defines how often the
aggregated usage will be charged to the user's account -->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

<!-- The signing fee is mandatory and defines how much the user will
be charged upfront on agreeing the SLA -->
<!-- The subscription fee is mandatory and defines how much the user
will be charged at the end of each billing period -->
    <signingFee>20.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
<!-- The currency applies to all the prices listed in the SLA
template -->
    <currency>EUR</currency>

<!-- The startTime and endTime define the validity period of the SLA
template -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

<!-- permittedServices is a list of which services the client may use
with this SLA.  If the list is empty then all services are permitted
-->
    <permittedServices>
        <permittedService>
            <url>https://argugrid-server.netmode.ntua.gr:8443/gria
                 -basic-app-services/services/HelloService</url>
        </permittedService>
    </permittedServices>

<!-- constraints is a mandatory list of constraint elements. -->
    <constraints>
<!-- This constraint limits the user to using at most 1000
CPU.seconds each day -->
```

```
        <constraint type="CUMULATIVE">
            <metric type="RESOURCE">
               <uri>http://www.gria.org/sla/metric/resource/cpu</uri>
                <description>
                    <description>CPU</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>CPU</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>1000.0</limit>
            <contention>1.0</contention>
<!-- This constraint is repeating.  It constrains the usage over a
period of time defined by the duration -->
            <repeating>true</repeating>
<!-- The duration of a repeating constraint would normally be the
same as the billing period -->
            <duration>
                <years>0</years>
                <months>0</months>
                <days>1</days>
                <hours>0</hours>
                <minutes>0</minutes>
                <seconds>0</seconds>
            </duration>
        </constraint>
    </constraints>

<!-- pricingTerms is a mandatory list of pricingTerm elements.  It
may be empty.  -->
    <pricingTerms>
<!-- A pricing term may be either of type CUMULATIVE or
INSTANTANEOUS_INCREASE -->
<!-- This pricing terms specifies that CPU usage is charged at 0.02
EUR per second -->
        <pricingTerm type="CUMULATIVE">
            <description>standard rate</description>
            <lowerBound>0</lowerBound>
                  <!-- An upperBound of -1 means +infinity -->
            <upperBound>-1</upperBound>
            <price>0.02</price>
            <metric type="RESOURCE">
               <uri>http://www.gria.org/sla/metric/resource/cpu</uri>
                <description>
                    <description>CPU</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>CPU</instantaneous>
                </units>
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

Results Found
Number of results is: 2

Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc
```

```
Outputs:
No results found!
Number of results is: 0
```

Στο ερώτημα 5 το ζητούμενο SLA template παρέχεται μόνο από τον host argugrid-server.netmode.ntua.gr.

Ερώτημα 6:

Αναζητήται το SLA template εκείνο το οποίο επιτρέπει την χρήση μέχρι 2000 CPU δευτερολέπτων ανά ημέρα και χρεώνει την χρήση κάθε δευτερολέπτου CPU με 0.01 ευρώ και ζητήται η αντίστοιχη Job Service που μπορεί να διαχειριστεί από αυτό το SLA template.

```
 The given query is:
<query>
  <select>$template $epr</select>
    <declare name="ns1">http://it-
    innovation.soton.ac.uk/2005/grid</declare>
    <declare
     name="addressing">http://www.w3.org/2005/08/addressing</declare>
      <from as="$epr">
         <class name="Reference"/>
           <join on="holdBy" as="$refAble" class="JobService">
             <join on="canbemanaged" as="$template"
                class="SlaTemplate"/>
           </join>
      </from>
      <where>
      $template//constraints//constraint//metric//description="CPU"
      and $template//constraints//constraint//limit="2000.0" and
      $template//pricingTerms//pricingTerm//price="0.01"
      </where>
</query>

Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0

 Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0

Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0
```

Παρατηρούμε ότι το ζητούμενο SLA template στο ερώτημα 6 δεν παρέχεται από κανέναν host.

Ερώτημα 7

Ζητήται το SLA template το οποίο επιτρέπει την δημιουργία μέχρι τριών job ανά ημέρα και η δημιουργία κάθε νέου job χρεώνεται με 0.01 ευρώ ανά ημέρα.

```
The given query is:
<query>
 <select>$template $epr</select>
  <declare name="ns1">http://it-
   innovation.soton.ac.uk/2005/grid</declare>
  <declare
   name="addressing">http://www.w3.org/2005/08/addressing</declare>
  <from as="$epr">
    <class name="Reference"/>
       <join on="holdBy" as="$refAble" class="JobService">
      <join on="canbemanaged" as="$template" class="SlaTemplate"/>
      </join>
    </from>
    <where>
     $template//constraints//constraint//metric//description="job"
     and $template//constraints//constraint//bound="LE" and
     $template//constraints//constraint//limit="3" and
     $template//pricingTerms//pricingTerm//price="1.10"
   </where>
 </query>
```

```
Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0

 Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
0.ID: [9494301b.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
    <Address>https://argugrid-server.netmode.ece.ntua.gr:8443/gria-
basic-app-services/services/JobService</Address>
    <Metadata>
        <ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.clie
nt.RemoteJobService</ns2:type>
        <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">https://argugrid-
server.netmode.ece.ntua.gr:8443/gria-basic-app-
services/services/JobService#13e68db1-16afeee9-0116-aff40458-
0002</ns3:default>
    </Metadata>
</EndpointReference>
```

```
 1.ID: [e43dac62.xml]
<slaTemplate>
    <label>Sla template for Job Service</label>
    <description>Sla for Job Service limited to 3 new jobs per day
     and 1.10 EUR per job</description>

<!-- The billing period defines how often the aggregated usage will
be charged to the user's account -->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

<!-- The signing fee defines how much the user will be charged
upfront on agreeing the SLA -->
<!-- The subscription fee defines how much the user will be charged
at the end of each billing period -->
    <signingFee>10.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
<!-- The currency applies to all the prices listed in the SLA
template -->
    <currency>EUR</currency>

<!-- The startTime and endTime define the validity period of the SLA
template -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

<!-- permittedServices is a list of which services the client may use
with this SLA.  If the list is empty then all services are permitted
-->
    <permittedServices>
        <permittedService>
 <url>https://argugrid-server.netmode.ece.ntua.gr:8443/gria-basic-
app-services/services/JobService</url>
        </permittedService>
    </permittedServices>

<!-- constraints is a mandatory list of constraint elements-->
    <constraints>
<!-- This constraint limits the user to using up to 3 jobs any one
time -->
        <constraint type="INSTANTANEOUS">
            <metric type="ACTIVITY">

<uri>http://www.gria.org/sla/metric/activity/job</uri>
                <description>
                    <description>job</description>
                </description>
```

```
                <units type="DECIMAL">
                    <instantaneous>job</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>3</limit>
            <contention>1.0</contention>
            <repeating>false</repeating>
        </constraint>
    </constraints>
    <pricingTerms>
      <!-- Every time a job starts, the instantaneous measurement of
      the number of jobs goes up by one (and then down by one when
      it finishes).  A pricingTerm of type INSTANTANEOUS_INCREASE
      calculates the increase of the instantaneous measurement over
      the billing period, ignoring decreases.  The instantaneous
      increase is then multiplied by the price accordingly. -->
        <pricingTerm type="INSTANTANEOUS_INCREASE">
            <description>creation charge</description>
            <lowerBound>0</lowerBound>
            <upperBound>-1</upperBound>
            <price>1.10</price>
            <metric type="ACTIVITY">
               <uri>http://www.gria.org/sla/metric/activity/job</uri>
                <description>
                    <description>job</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>job</instantaneous>
                </units>
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

 Results Found
Number of results is: 2

 Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc
 Outputs:
No results found!
Number of results is: 0
```

Το ζητούμενο SLA Template παρέχεται μόνο από τον argugrid.

Ερώτημα 7

Ζητήται το SLA template το οποίο επιτρέπει την δημιουργία μέχρι 5 πόρων της υπηρεσίας Hello Service οι οποίοι ονομάζονται sample resource και χρεώνει την δημιουργία κάθε πόρου με 0.01 ευρώ.

```
The given query is:
<query>
  <select>$template $epr</select>
  <declare name="ns1">http://it-
innovation.soton.ac.uk/2005/grid</declare>
```

```
  <declare
name="addressing">http://www.w3.org/2005/08/addressing</declare>
  <from as="$epr">
      <class name="Reference"/>
        <join on="holdBy" as="$refAble" class="SlaManagedService">
            <join on="canbemanaged" as="$template"
class="SlaTemplate"/>
       </join>
  </from>
  <where>
   $template//constraints//constraint//metric//description="sample-
resource" and
   $template//constraints//constraint//bound="LE" and
   $template//constraints//constraint//limit="5" and
   $template//pricingTerms//pricingTerm//price="0.01"
  </where>
</query>


Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
0.ID: [e206a906.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">

<Address>https://dimitra.netmode.ntua.gr:8443/hellogria/services/Hell
oService</Address>
    <Metadata>
        <ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">hellogria.client.RemoteHelloService
</ns2:type>
        <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">https://dimitra.netmode.ntua.gr:844
3/hellogria/services/HelloService#13e68db1-171b9fd4-0117-1bc10e99-
0001</ns3:default>
    </Metadata>
</EndpointReference>

 1.ID: [e1b6cfe2.xml]
<slaTemplate>
    <label>SLA Template for Hellogria Service</label>
    <description>SLA Template for Hellogria Service Limited to 5
     resources 0.01 EUR per resource.</description>
<!-- The billing period is mandatory and defines how often the
aggregated usage will be charged to the user's account -->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

<!-- The signing fee is mandatory and defines how much the user will
be charged upfront on agreeing the SLA -->
<!-- The subscription fee is mandatory and defines how much the user
will be charged at the end of each billing period -->
    <signingFee>20.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
```

```
<!-- The currency applies to all the prices listed in the SLA
template -->
    <currency>EUR</currency>

<!-- The startTime and endTime define the validity period of the SLA
template -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

<!-- permittedServices is a list of which services the client may use
with this SLA.  -->
<permittedServices>

<permittedService>
https://dimitra.netmode.ntua.gr:8443/hellogria/services/HelloService
</permittedService>
 </permittedServices>
     <!-- constraints is a mandatory list of constraint elements -->
    <constraints>
<!-- This constraint limits the user to using up to 5 sample-
resources any one time -->
        <constraint type="INSTANTANEOUS">
            <metric type="RESOURCE">
                <uri>http://SAMPLE/2006/metrics/sample-resource</uri>
                <description>
                    <description>sample-resource</description>
                    <instantaneous>sample-resource</instantaneous>
                </description>
                <units type="DECIMAL">
                    <instantaneous>sample-resource</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>5</limit>
            <contention>1.0</contention>
            <repeating>false</repeating>
        </constraint>
    </constraints>
    <pricingTerms>
<!-- This pricing term specifies that there is a charge of 0.05 EUR
every time a sample-resource is created -->
        <pricingTerm type="INSTANTANEOUS_INCREASE">
            <description>creation charge</description>
            <lowerBound>0</lowerBound>
            <upperBound>-1</upperBound>
            <price>0.01</price>
            <metric type="ACTIVITY">
                <uri>http://SAMPLE/2006/metrics/sample-resource</uri>
                <description>
                    <description>sample-resource</description>
                </description>
                <units type="DECIMAL">
```

```
                    <instantaneous>sample-resource</instantaneous>
                </units>
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

 Results Found
Number of results is: 2

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0


Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc

Outputs:
No results found!
Number of results is: 0
```

Κάθε φορά που γίνεται σύνδεση με κάποια βάση δεδομένων εμφανίζονται τα παρακάτω στην οθόνη του χρήστη :

niobe:

```
Using configDir = C:/Documents and Settings/Dimitra
Kollia/workspace/Registry/src/niobe

Login...
User: admin password: admin

Sindesi me tin vasi (uri):
xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc

Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc
```

dimitra:

```
Using configDir = C:/Documents and Settings/Dimitra
Kollia/workspace/Registry/src/resources

Login...
User: admin password: admin
Sindesi me tin vasi (uri):
xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc
```

argugrid:

```
Using configDir = C:/Documents and Settings/Dimitra
Kollia/workspace/Registry/src/argugrid
```

```
Login...
User: admin password: admin
Sindesi me tin vasi (uri): xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc
```

5.2. Συμφωνία του SLA

Ο χρήστης μπορεί στην συνέχεια να χρησιμοποιήσει τον client CallHelloService που παρουσιαστηκε προηγουμένως CallHelloService για να συμφωνήσει ένα SLA με αυτό το template επέλεξε και για την υπηρεσία που επέλεξε.

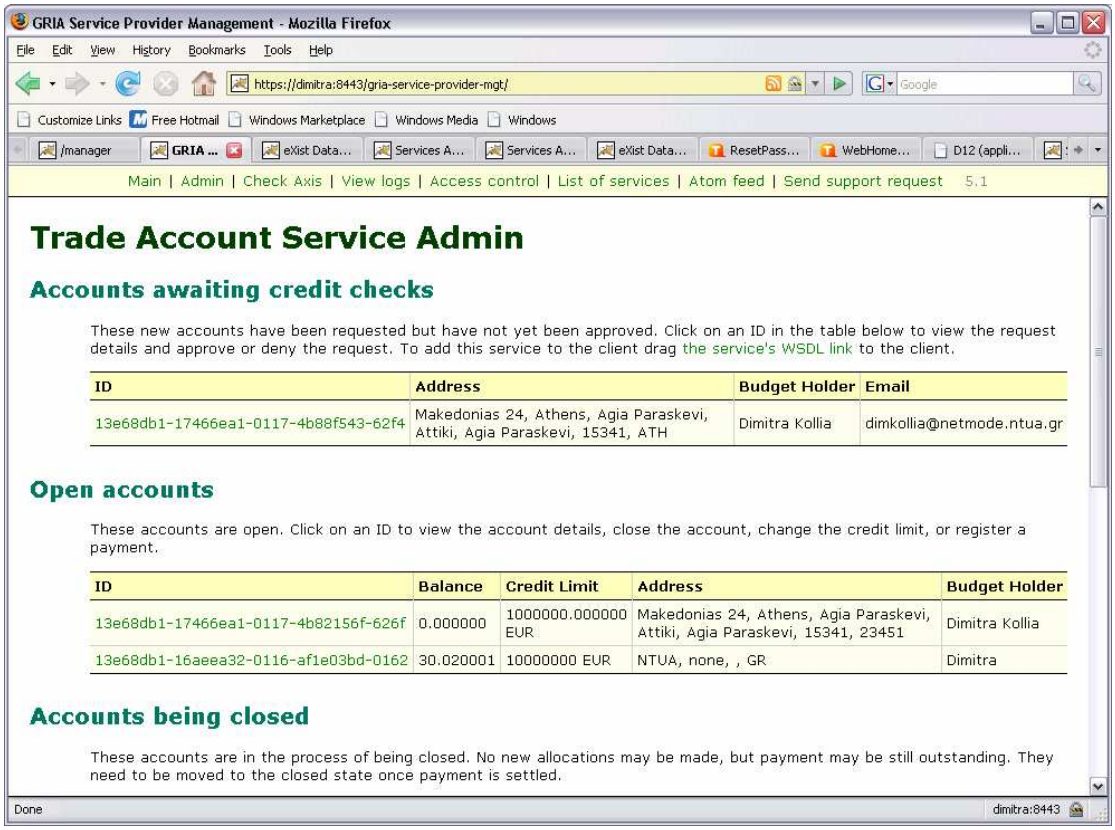
Αποτελέσματα

Υλοποιώντας τα παραπάνω προκύπτουν τα παρακάτω αποτελέσματα:

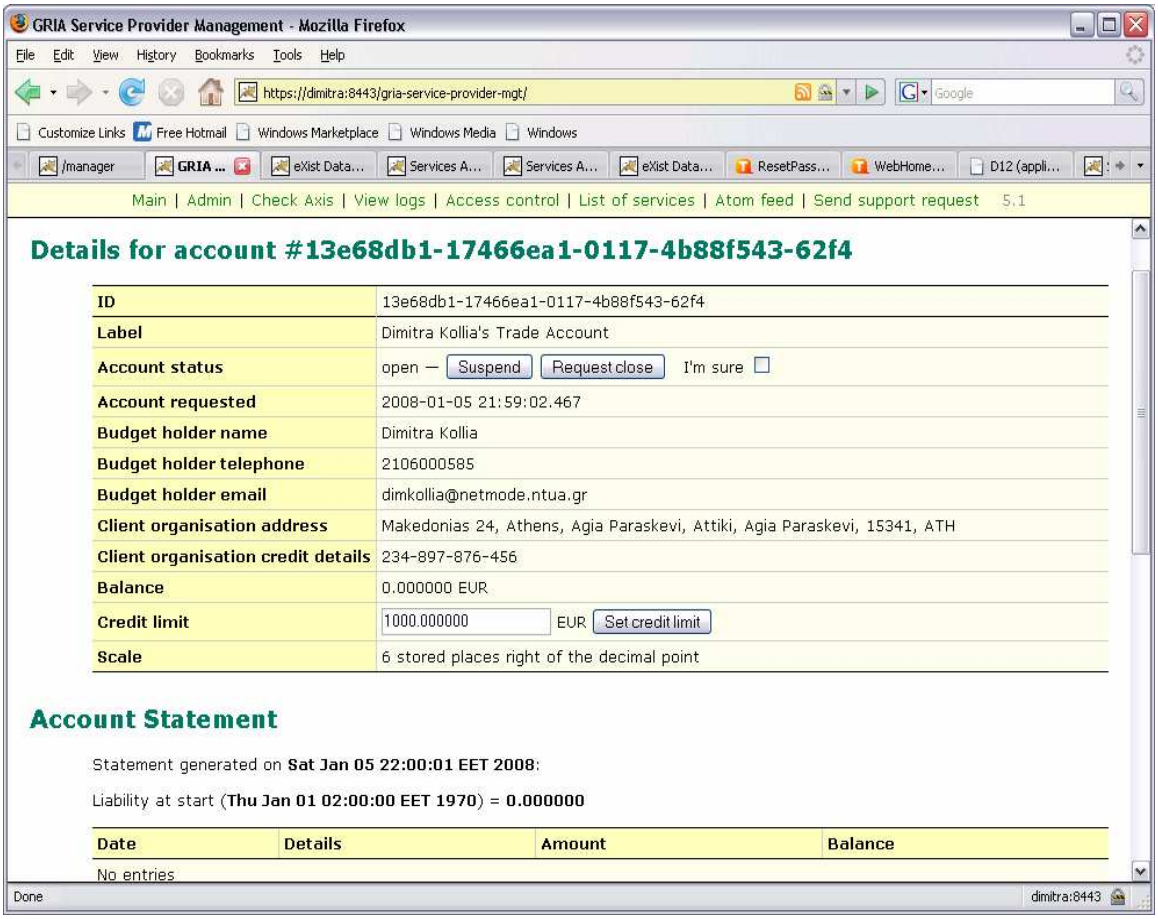
Ο χρήστης ανοίγει καινούριο λογαριασμό:

```
- Loading keystore 'C:\GRIDHELLOSAMPLE\config\service-keystore.ks'
Do you want to open a new account? (y/n)

Dwse ta stoixeia tou account:
Holder Name: Dimitra Kollia
Holder Telephone: 2106000585
Holder e-mail: dimkollia@netmode.ntua.gr
Client-Organisation Address:
City: Athens
Country Code: ATH
Postal Code: 15341
Region: Agia Paraskevi
State: Attiki
Street: Makedonias 24
Town: Agia Paraskevi
Credit details: 234-897-876-456
Label: Dimitra Kollia's Trade Account
```

Ο λογαριασμός έχει σταλεί στην Trade Account υπηρεσία και είναι στην κατάσταση “awaiting credit checks” :



Ο διαχειριστής της υπηρεσίας λογαριασμών έχει αποδεχτεί την αίτηση για την δημιουργία νέου λογαριασμού και ορίζει το όριο πίστωσης το οποίο προσδιορίζει και τον βαθμό εμπιστοσύνης του προς τον χρήστη ίσο με 1000 ευρώ. Ο λογαριασμός είναι στην κατάσταση open και μπορεί πλέον να χρησιμοποιηθεί για την συμφωνία SLAs. Η μορφή του λογαριασμού, όπως είναι στην υπηρεσία λογαριασμών,φαίνεται στην παρακάτω εικόνα:




```
.... Αναμονή μέχρι ο λογαριασμός να τεθεί σε κατάσταση open από την
Trade Account Service …….
- Invoking TradeAccountService.getAccountStatus#13e68db1-17466ea1-
0117-4b88f543-62f4 (0 arguments 0 headers)
- Verification successful for URI "#id-31480948"
```

Ο λογαριασμός είναι σε κατάσταση open και μπορεί να χρησιμοποιηθεί.

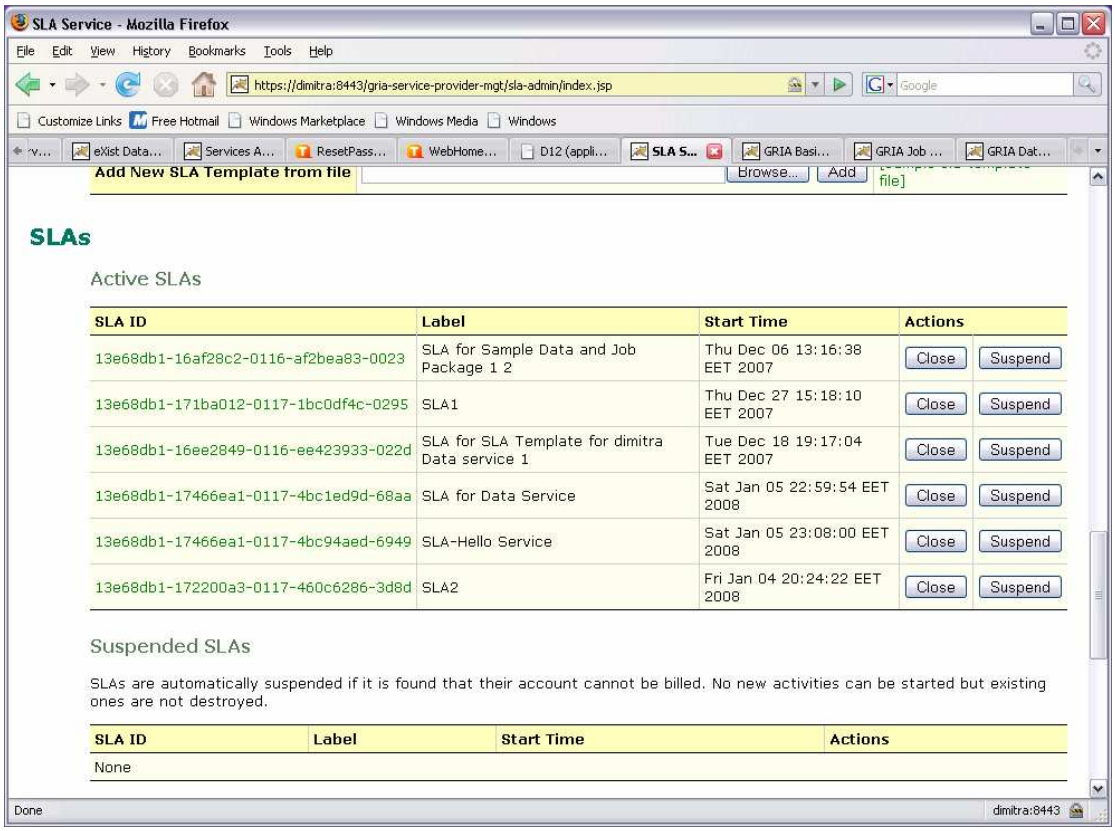
Συμφωνία νέου SLA(συνέχεια)

```
Do you want to create a new SLA?(y/n)
y
answer is :y
- Fetching WSDL from https://dimitra.netmode.ntua.gr:8443/gria-
service-provider-mgt/services/SLAService?wsdl
- Invoking SLAService.getResources (0 arguments 0 headers)
- Verification successful for URI "#id-27223228"

Τα SLA templates που παρέχονται από τον χρήστη:
Searching for SLA's templates to use...

Sample Data and Job Package 1
SLA Template for Hellogria Service-helloresource1
SLA Template for Data Service
SLA for NEW Service Package

Give the name of the sla-template you want:
SLA Template for Hellogria Service-helloresource1

 epr του SLA template που θα χρησιμοποιηθεί:
Address: https://dimitra.netmode.ntua.gr:8443/gria-service-provider-
mgt/services/SLAService
Reference property[0]:
<itinnov:ConversationID xmlns:itinnov="http://it-
innovation.soton.ac.uk/2005/grid">13e68db1-17466ea1-0117-4bb33731-
672e</itinnov:ConversationID>
Metadata [0]:
<ns52:title xmlns:ns52="http://purl.org/dc/elements/1.1/">SLA
Template for Hellogria Service-helloresource1</ns52:title>
Metadata [1]:
<ns53:serviceresourcetype xmlns:ns53="http://it-
innovation.soton.ac.uk/2005/grid">http://www.it-
innovation.soton.ac.uk/grid/resource/sla-
template</ns53:serviceresourcetype>
Metadata [2]:
<ns54:status xmlns:ns54="http://it-
innovation.soton.ac.uk/2005/grid">published</ns54:status>

- Invoking SLAService.getSLATemplate#13e68db1-17466ea1-0117-4bb33731-
672e (0 arguments 0 headers)
- Verification successful for URI "#id-31513350"

You chose the SLA-template: SLA Template for Hellogria Service-
helloresource1

Give a name for the new SLA:
SLA-Hello Service
- Invoking SLAService.getTrustedAccountServices (0 arguments 0
headers)
- Verification successful for URI "#id-28244442"
- Billing information not set by user.
- Using Trade Account :Dimitra Kollia's Trade Account

Δημιουργία του SLA
- Invoking SLAService.createSLA (3 arguments 1 header)
- Verification successful for URI "#id-3564969"
New SLA Created 'SLA-Hello Service'
```

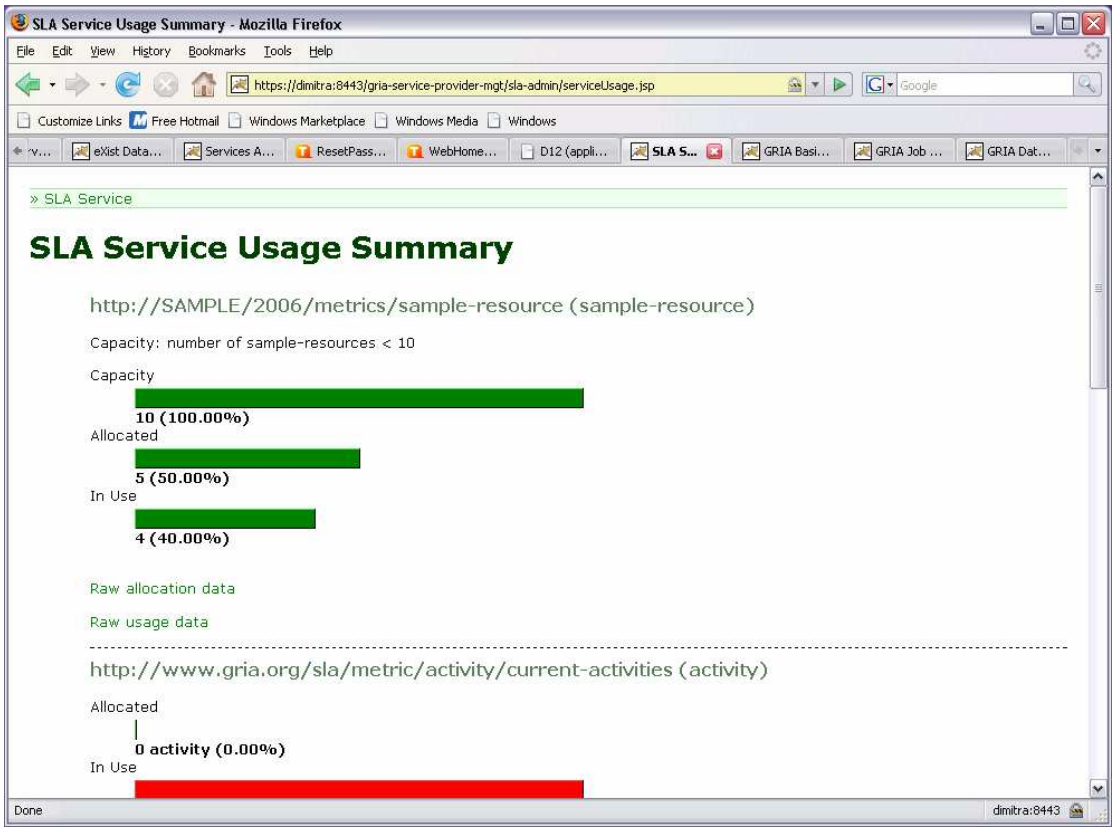
Ο χρήστης μπορεί πλέον να χρησιμοποιήσει την Hello Service με το SLA που συμφώνησε:

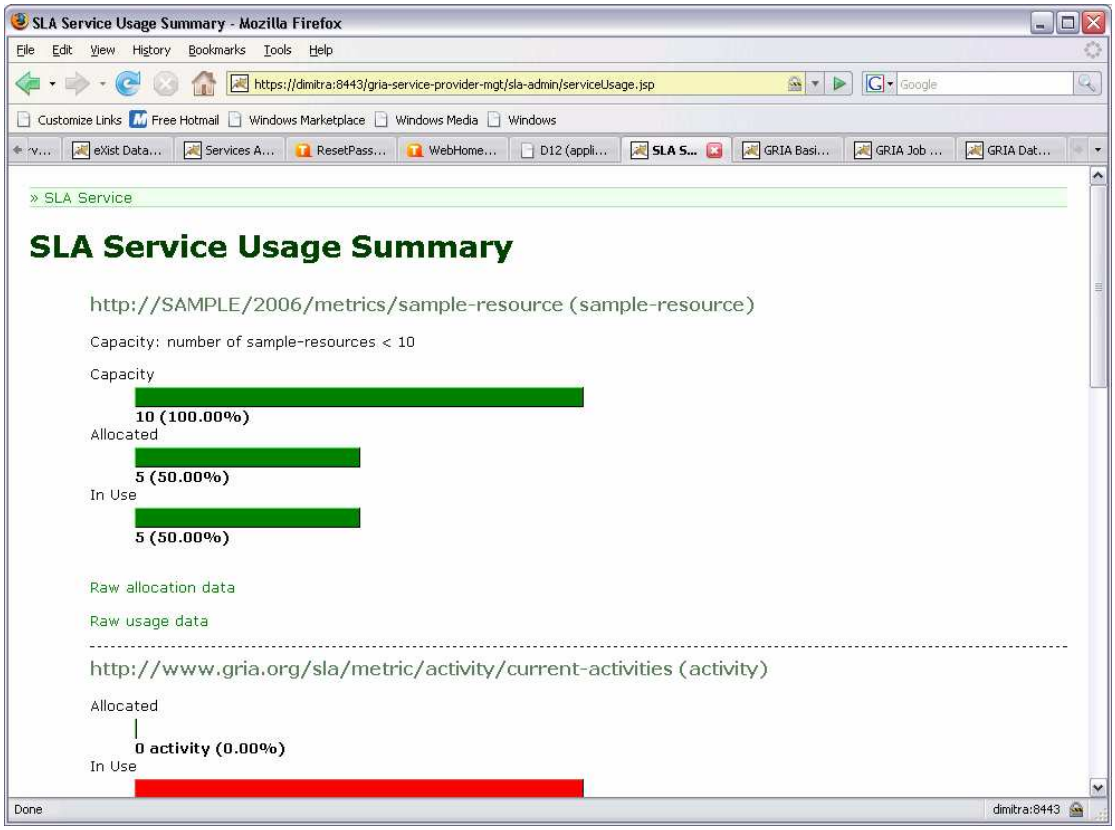
```
Δημιουργία του νέου resource στην Hello Service
Give a name for the new resource: helloresource
helloresource
- Fetching WSDL from
https://dimitra.netmode.ntua.gr:8443/hellogria/services/HelloService?
wsdl
-- Invoking HelloService.getTrustedAccountServices (0 arguments 0
headers)
- Verification successful for URI "#id-25133301"
- Using billing epr https://dimitra.netmode.ntua.gr:8443/gria-
service-provider-mgt/services/SLAService
- Invoking HelloService.createHelloResource (2 arguments 1 header)
- Verification successful for URI "#id-32618511"
- Fetching WSDL from
https://dimitra.netmode.ntua.gr:8443/hellogria/services/HelloResource
?wsdl
- Invoking HelloResource.HelloWorld#13e68db1-17466ece-0117-4bcb1491-
0002 (0 arguments 0 headers)
- Verification successful for URI "#id-4182591"
Hello World!
```

Στην εικόνα φαίνεται το SLA που συμφωνήθηκε. Είναι το προτελευταίο SLA:



Στην παρακάτω εικόνα φαίνεται η χρησιμοποίηση του sample-resource όπως φαίνεται από την πλευρά του παρόχου. Στην πρώτη ράβδο φαίνεται η συνολική δυνατότητα του παρόχου της Hello Service να παρέχει σε όλους τους χρήστες συνολικά μέχρι 10 πόρους sample-resource. Στην δεύτερη ράβδο φαίνεται ο αριθμός των πόρων που έχουν δεσμευθεί σύμφωνα με το SLA που έχει συμφωνηθεί. Εδώ έχει συμφωνηθεί ένα SLA που επιτρέπει μέχρι πέντε πόρους σε κάθε χρήστη οπότε έχουν δεσμευθεί πέντε πόροι. Η τρίτη ράβδος δείχνει πόσοι πόροι sample resource έχουν δημιουργηθεί και επομένως καταναλωθεί. Παρατηρούμε ότι επειδή κλήθηκε τέσσερις φορές η Hello Service δημιουργήθηκαν 4 πόροι.



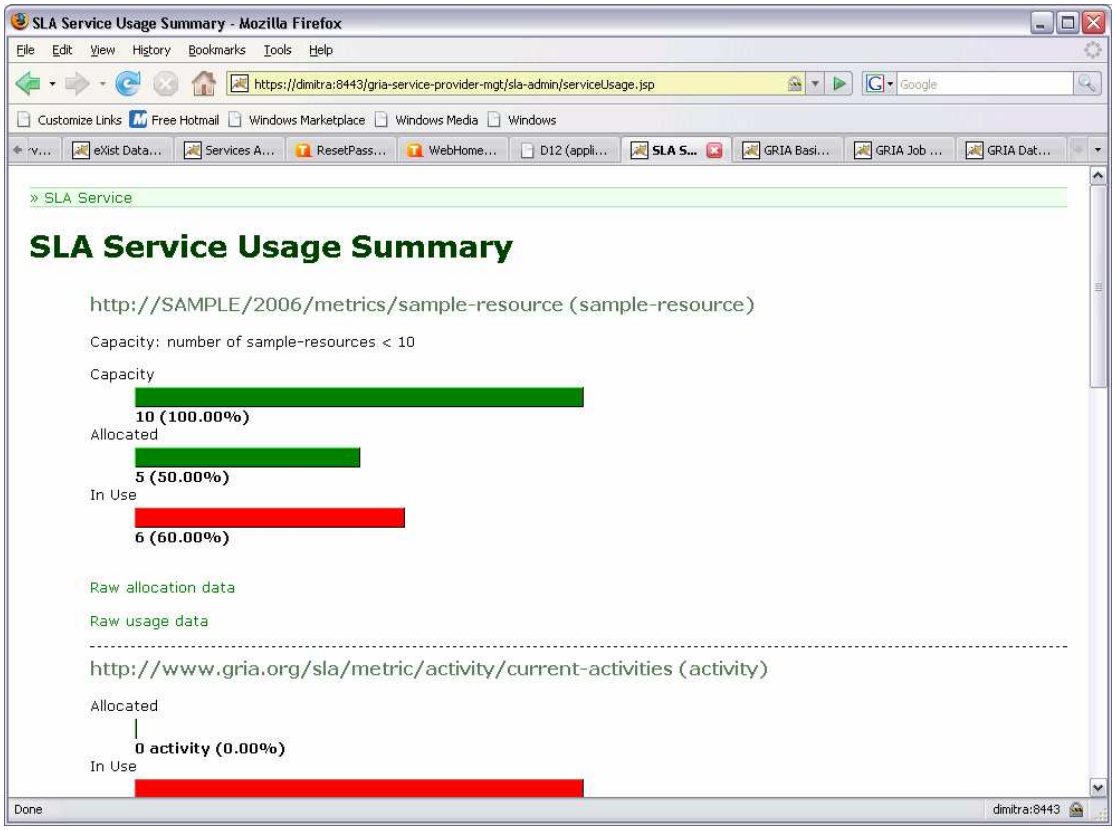

SLA Service Usage Summary - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://dimitra:8443/gria-service-provider-mgt/sla-admin/serviceUsage.jsp
Customize Links Free Hotmail Windows Marketplace Windows Media Windows
» SLA Service
SLA Service Usage Summary
http://SAMPLE/2006/metrics/sample-resource (sample-resource)
Capacity: number of sample-resources < 10
Capacity
10 (100.00%)
Allocated
5 (50.00%)
In Use
5 (50.00%)
Raw allocation data
Raw usage data
http://www.gria.org/sla/metric/activity/current-activities (activity)
Allocated
0 activity (0.00%)
In Use
Done
dimitra:8443

Αν ο ίδιος χρήστης να δημιουργήσει και άλλα δύο sample resources χρησιμοποιώντας τον ίδιο λογαριασμό και το ίδιο SLA:

```
Do you want to open a new account? (y/n)
n
Give the name for your account:
Dimitra Kollia's Trade Account
- Fetching WSDL from https://dimitra.netmode.ntua.gr:8443/gria-
service-provider-mgt/services/TradeAccountService?wsdl
- Invoking TradeAccountService.getResources (0 arguments 0 headers)
- Verification successful for URI "#id-6673186"
The label of the account is: Dimitra Kollia's Trade Account
Dimitra Kollia's Trade Account

Address: https://dimitra.netmode.ntua.gr:8443/gria-service-provider-
mgt/services/TradeAccountService
Reference property[0]:
<itinnov:ConversationID xmlns:itinnov="http://it-
innovation.soton.ac.uk/2005/grid">13e68db1-17466ea1-0117-4b88f543-
62f4</itinnov:ConversationID>
Metadata [0]:
<ns4:title xmlns:ns4="http://purl.org/dc/elements/1.1/">Dimitra
Kollia's Trade Account</ns4:title>
Metadata [1]:
<ns5:serviceresourcetype xmlns:ns5="http://it-
innovation.soton.ac.uk/2005/grid">http://www.it-
innovation.soton.ac.uk/grid/resource/trade-
account</ns5:serviceresourcetype>
Metadata [2]:
<ns6:status xmlns:ns6="http://it-
innovation.soton.ac.uk/2005/grid">open</ns6:status>
Metadata [3]:
<ns1:type xmlns:ns1="http://it-
innovation.soton.ac.uk/2005/grid">uk.ac.soton.ecs.iam.grid.comms.clie
nt.TradeAccountConversation</ns1:type>
Metadata [4]:
<ns1:parent xmlns:ns1="http://it-
innovation.soton.ac.uk/2005/grid">https://dimitra.netmode.ntua.gr:844
3/gria-service-provider-mgt/services/TradeAccountService</ns1:parent>

Do you want to create a new SLA?(y/n)
n
answer is :n
- Fetching WSDL from https://dimitra.netmode.ntua.gr:8443/gria-
service-provider-mgt/services/SLAService?wsdl
- Invoking SLAService.getResources (0 arguments 0 headers)
- Verification successful for URI "#id-19783010"

Searching for SLA's to use...

Type resource is http://www.it-
innovation.soton.ac.uk/grid/resource/sla

Class name is: uk.ac.soton.ecs.iam.grid.comms.client.SLAConversation


Label for the SLA is: SLA for Sample Data and Job Package 1 2
Label for the SLA is: SLA1
Label for the SLA is: SLA for SLA Template for dimitra Data service 1
Label for the SLA is: SLA for Data Service
```

```
Label for the SLA is: SLA-Hello Service
Label for the SLA is: SLA2
Label for the SLA is: SLA for SLA Template for dimitra Data service 2

Give the name for the SLA you want:
SLA-Hello Service

SLA Found 'SLA-Hello Service'

Give a name for the new resource:
helloresource_Dimitra Kollia2
- Fetching WSDL from
https://dimitra.netmode.ntua.gr:8443/hellogria/services/HelloService?
wsdl
- Invoking HelloService.getTrustedAccountServices (0 arguments 0
headers)
- Verification successful for URI "#id-19014742"
- Using billing epr https://dimitra.netmode.ntua.gr:8443/gria-
service-provider-mgt/services/SLAService
- Invoking HelloService.createHelloResource (2 arguments 1 header)
- Verification successful for URI "#id-28312319"
- Fetching WSDL from
https://dimitra.netmode.ntua.gr:8443/hellogria/services/HelloResource
?wsdl
- Invoking HelloResource.HelloWorld#13e68db1-17466ece-0117-4bda3b5a-
0004 (0 arguments 0 headers)
- Verification successful for URI "#id-17622485"
Hello World!
```

Παρατηρούμε ότι ξεπέρασε το όριο που είχε συμφωνηθεί στο SLA. Δημιουργήθηκε όμως ο νέος πόρος επειδή το σύστημα διέθετε ελεύθερους πόρους και δεν απαιτούνταν από άλλα SLAs. Όμως θα υπάρξουν κυρώσεις προς τον χρήστη. Όταν τελειώσουν οι διαθέσιμοι πόροι δεν μας επιτρέπει την δημιουργία νέου πόρου.



5.3. Use Case από το project Argugrid

Θα χρησιμοποιηθεί ένα σενάριο από την περίπτωση παρατήρησης της γης (Earth Observation).
Στο σενάριο που θα υλοποιηθεί θέλουμε να επιλέξουμε την υπηρεσία που εντοπίζει πετρελαιοκηλίδες έπειτα από παρατήρηση και επεξεργασία φωτογραφιών που έχουν παρθεί από δορυφόρους. Επομένως ο πόρος που θα παρέχει η υπηρεσία θα είναι η δορυφορική φωτογραφία. Η ποιότητα της φωτογραφίας θεωρούμε ότι χαρακτηρίζεται από τα pixels της φωτογραφίας. Η φωτογραφία πρέπει να έχει οπωσδήποτε ανάλυση 10 Mpixels ώστε να είναι πιο ακριβής ο εντοπισμός της πετρελαιοκηλίδας. Αυτό θα αποτελέσει και το κριτήριο αναζήτησης.

Για την υλοποίηση του σεναρίου θεωρούμε ότι και οι τρεις υπολογιστές dimitra.netmode.ntua.gr, niobe.netmode.ntua.gr, argugrid-server.netmode.ntua.gr παρέχουν τη ζητούμενη υπηρεσία αλλά με διαφορετικό επίπεδο ποιότητας δηλαδή παρέχουν φωτογραφίες με διαφορετική ανάλυση. Αυτή την παροχή τους την «διαφημίζουν» στα αντίστοιχα SLA templates που δημοσιοποιούν στην υπηρεσία

SLA που παρέχεται από τον υπολογιστή από τον οποίο παρέχονται κα αυτές. Να σημειωθεί ότι σε κάθε SLA υπηρεσία έχει δημιουργηθεί η μετρική με uri http://www.EO.com/sla/metric/resource/image/resolution για την μέτρηση από την υπηρεσία SLA αυτού του πόρου.

Συγκεκριμένα:

Στον dimitra.netmode.ntua.gr οι φωτογραφίες παρέχονται με ανάλυση 10 Mpixels ή μικρότερη.

Στον niobe.netmode.ntua.gr οι φωτογραφίες παρέχονται με ανάλυση 5 pixels ή μικρότερη.

Στον argugrid-server.netmode.ntua.gr οι φωτογραφίες παρέχονται με ανάλυση 7 Mpixels ή μικρότερη.

Για την υλοποίηση της αναζήτησης ο διαχειριστής του Registry προσθέτει τις wsdl περιγραφές των υπηρεσιών υπό την οντότητα “SlaManagedService” και τα xml αρχεία με τα Sla templates υπό την οντότητα SlaTemplate ενώ καταχωρεί και τις αντίστοιχες σχέσεις μεταξύ τους ακριβώς όπως και στην υλοποίηση των προηγούμενων σεναρίων.

Το αποτέλεσμα είναι αυτό που περιμέναμε:

```
The given query is:

<query>
  <select>$template $epr</select>
  <declare name="ns1">http://it-
innovation.soton.ac.uk/2005/grid</declare>
  <declare
name="addressing">http://www.w3.org/2005/08/addressing</declare>
  <from as="$epr">
      <class name="Reference"/>
        <join on="holdBy" as="$refAble" class="SlaManagedService">
            <join on="canbemanaged" as="$template"
class="SlaTemplate"/>
       </join>
  </from>
  <where>
   $template//constraints//constraint//metric//description="image-
resolution" and
   $template//constraints//constraint//bound="LE" and
   $template//constraints//constraint//limit="10" and
   $template//pricingTerms//pricingTerm//price="0.01"
  </where>
</query>
```

```
Searching in xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc
Outputs:
0.ID: [7c333852.xml]
<EndpointReference xmlns="http://www.w3.org/2005/08/addressing">

<Address>https://dimitra.netmode.ntua.gr:8443/EO/services/SatelliteIm
ageService</Address>
    <Metadata>
<ns2:type xmlns:ns2="http://it-
innovation.soton.ac.uk/2005/grid">hellogria.client.RemoteHelloService
</ns2:type>
  <ns3:default xmlns:ns3="http://it-
innovation.soton.ac.uk/2005/grid">
https://dimitra.netmode.ntua.gr:8443/EO/services/SatelliteImageServic
e#14d65db2-171b9df5-0118-1cb0023g-0004</ns3:default>
    </Metadata>
</EndpointReference>

 1.ID: [c88a1b14.xml]
<slaTemplate>
    <label>SLA Template for image service</label>
    <description>SLA Template for image Service Limited to 10 Mpixels
and 0.01 EUR per pixel.</description>

      <!-- The billing period is mandatory and defines how often the
aggregated usage will be charged to the user's account -->
    <billingPeriod>
        <years>0</years>
        <months>0</months>
        <days>1</days>
        <hours>0</hours>
        <minutes>0</minutes>
        <seconds>0</seconds>
    </billingPeriod>

      <!-- The signing fee is mandatory and defines how much the user
will be charged upfront on agreeing the SLA -->
      <!-- The subscription fee is mandatory and defines how much the
user will be charged at the end of each billing period -->
    <signingFee>20.00</signingFee>
    <subscriptionFee>10.00</subscriptionFee>
      <!-- The currency applies to all the prices listed in the SLA
template -->
    <currency>EUR</currency>

      <!-- The startTime and endTime define the validity period of
the SLA template -->
    <startTime>
        <year>2000</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </startTime>
    <endTime>
        <year>2010</year>
        <month>1</month>
        <dayOfMonth>1</dayOfMonth>
    </endTime>

      <!-- permittedServices is a list of which services the client
may use with this SLA.  →
```

```
     <!-- constraints is a mandatory list of constraint elements.  --
>
    <constraints>
            <!-- This constraint limits the user to using up to 10
Mpixels any one time -->
        <constraint type="CUMULATIVE">
            <metric type="RESOURCE">

<uri>http://www.EO.com/sla/metric/resource/image/resolution</uri>
                <description>
                    <description>image-resolution</description>
                    <instantaneous>image-resolution</instantaneous>
                </description>
                <units type="DECIMAL">
                    <instantaneous>pixels</instantaneous>
                </units>
            </metric>
            <bound>LE</bound>
            <private>false</private>
            <limit>10</limit>
            <contention>1.0</contention>
            <repeating>false</repeating>
        </constraint>
    </constraints>
    <pricingTerms>
<!-- This pricing term specifies that there is a charge of 0.01 EUR
every time a sample-resource is created -->
        <pricingTerm type="CUMULATIVE">
            <description>pixel charge</description>
            <lowerBound>0</lowerBound>
            <upperBound>-1</upperBound>
            <price>0.00</price>
            <metric type="ACTIVITY">

<uri>http://www.EO.com/sla/metric/resource/image/resolution</uri>
                <description>
                    <description>image-resolution</description>
                </description>
                <units type="DECIMAL">
                    <instantaneous>image-resolution</instantaneous>
                </units>
            </metric>
        </pricingTerm>
    </pricingTerms>
</slaTemplate>

 Results Found
Number of results is: 2



Searching in xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc
Outputs:
No results found!
Number of results is: 0

Searching in xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc
```

```
Outputs:
Outputs:
No results found!
Number of results is: 0
```

Μόνο η υπηρεσία που παρέχεται από τον dimitra.netmode.ntua.gr μπορεί να παρέχει φωτογραφίες με ανάλυση 10 Mpixels.

ΠΑΡΑΡΤΗΜΑ

registry.properties

```
#########################################################################
####
#
# Registry properties specification
#
#########################################################################
####

# database administrator for installing and configuration of the
registry database
db.admin.user=admin
db.admin.password=admin
db.admin.group=dba

# namespace of registry domain model
# e.g. 'http://www.gria.org/registrydomainmodel.owl'
registry.domain.model.ns=http://www.gria.org/RegistryDomainModel.owl

# XML database URI
# e.g. 'xmldb:exist://localhost:8080/exist/xmlrpc'
xml.db.uri=xmldb:exist://dimitra.netmode.ntua.gr:8080/exist/xmlrpc

# database name in the XML database
registry.database.name=tutorial

# Registration default policy. Used during registration.
# Usage: r|w|u, e.g. for an owner who should have read,
# write and update access: xml.db.perm.owner=rwu

# entities
# owner
xml.db.perm.entity.owner=rwu
# group
xml.db.perm.entity.group=r
# other
xml.db.perm.entity.other=r

# concepts
# owner
xml.db.perm.concept.owner=rwu
# group
xml.db.perm.concept.group=rw
# other
xml.db.perm.concept.other=r


#########################################################################
####
#
# Registry properties specification
#
#########################################################################
####

# database administrator for installing and configuration of the
registry database
```

```
db.admin.user=admin

db.admin.password=admin
db.admin.group=dba

# namespace of registry domain model
# e.g. 'http://www.gria.org/registrydomainmodel.owl'
registry.domain.model.ns=http://www.gria.org/RegistryDomainModel.owl

# XML database URI
# e.g. 'xmldb:exist://localhost:8080/exist/xmlrpc'
xml.db.uri=xmldb:exist://argugrid-
server.netmode.ntua.gr:8080/exist/xmlrpc

# database name in the XML database
registry.database.name=tutorial

# Registration default policy. Used during registration.
# Usage: r|w|u, e.g. for an owner who should have read,
 write and update access: xml.db.perm.owner=rwu

# entities
# owner
xml.db.perm.entity.owner=rwu
# group
xml.db.perm.entity.group=r
# other
xml.db.perm.entity.other=

# concepts
# owner
xml.db.perm.concept.owner=rwu
# group
xml.db.perm.concept.group=rw
# other
xml.db.perm.concept.other=


#
##########################################################################
####
#
# Registry properties specification
#
##########################################################################
####

# database administrator for installing and configuration of the
registry database
db.admin.user=admin

db.admin.password=admin
db.admin.group=dba

# namespace of registry domain model
# e.g. 'http://www.gria.org/registrydomainmodel.owl'
registry.domain.model.ns=http://www.gria.org/RegistryDomainModel.owl

# XML database URI
# e.g. 'xmldb:exist://localhost:8080/exist/xmlrpc'
xml.db.uri=xmldb:exist://niobe.netmode.ntua.gr:8080/exist/xmlrpc
```

```
# database name in the XML database
registry.database.name=tutorial

# Registration default policy. Used during registration.
# Usage: r|w|u, e.g. for an owner who should have read,
 write and update access: xml.db.perm.owner=rwu

# entities
# owner
xml.db.perm.entity.owner=rwu
# group
xml.db.perm.entity.group=r
# other
xml.db.perm.entity.other=

# concepts
# owner
xml.db.perm.concept.owner=rwu
# group
xml.db.perm.concept.group=rw
# other
xml.db.perm.concept.other=
```

implementationfactory.properties

```
# Communication channel to registry database
uk.ac.soton.itinnovation.registry.component.xml.db.XmlDbAccessManager
=
uk.ac.soton.itinnovation.registry.component.xml.db.xmldb.XmlDbAccessM
anagerXmlDb
#uk.ac.soton.itinnovation.registry.component.xml.db.XmlDbAccessManage
r =
uk.ac.soton.itinnovation.registry.component.xml.db.soap.XmlDbAccessMa
nagerSOAP
```

RegistryDomainModel.owl

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#">
  <owl:Ontology
rdf:about="http://www.gria.org/RegistryDomainModel.owl">
    <rdfs:comment>Registry Domain Model for GRIA Client Management.
(C) 2006, University of Southampton IT Innovation
Center</rdfs:comment>
    <owl:versionInfo>$Id: RegistryDomainModel.owl,v 1.0 2006/10/31
Uwe Radetzki$</owl:versionInfo>
  </owl:Ontology>
  <rdfs:Datatype
rdf:about="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:Datatype
rdf:about="http://www.w3.org/2001/XMLSchema#anyType"/>
```

```
  <rdfs:Datatype
rdf:about="http://www.w3.org/2001/XMLSchema#anyURI"/>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Reference">
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Reference could be an EPR</rdfs:comment>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Reference</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Registry">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Registry</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagedResourc
e"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SlaTemplate</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Resource"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaManagedReso
urce">
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagedResourc
e"/>
    </rdfs:subClassOf>
    <rdfs:label>SlaManagedResource</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaManagedServ
ice">
    <rdfs:label>SlaManagedService</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagedResourc
e"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagedResourc
e">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ManagedResource</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Resource"/>
```

```
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#RemoteDataServ
ice">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:label>RemoteDataService</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#DataService">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagedReso
urce"/>
    <rdfs:label>DataService</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Resource">
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Resources are created by services</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble"
/>
    </rdfs:subClassOf>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Resource</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#TradeAccountCo
nversation">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:label>TradeAccountConversation</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#JobService">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagedReso
urce"/>
    <rdfs:label>JobService</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#RemoteHelloSer
vice">
    <rdfs:label>RemoteHelloService</rdfs:label>
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Service">
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagerResourc
e"/>
    </rdfs:subClassOf>
```

```
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Service</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SLAConversatio
n">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:label>SLAConversation</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagerResourc
e">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ManagerResource</rdfs:label>
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagedReso
urce"/>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#RemoteSLAServi
ce">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:label>RemoteSLAService</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Sla">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Sla</rdfs:label>
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagerReso
urce"/>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#MegaAccount">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagerReso
urce"/>
    <rdfs:label>MegaAccount</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Account">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagerReso
urce"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Account</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble"
>
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >Reference-able elements have a unique reference
(EPR)</rdfs:comment>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ReferenceAble</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#RemoteJobServi
ce">
    <rdfs:subClassOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:label>RemoteJobService</rdfs:label>
  </owl:Class>
  <owl:Class
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Usage">
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Usage reports on SLAs</rdfs:comment>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Usage</rdfs:label>
  </owl:Class>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaManagedReso
urce-canbemanaged-SlaTemplate">
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-SlaManagedResource"/>
    </owl:inverseOf>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaManagedR
esource"/>
    <rdfs:label>canbemanaged</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble-
containedIn-Registry">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >containedIn</rdfs:label>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Registry"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Registry-
contains-ReferenceAble"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Reference-
holdBy-ReferenceAble">
```

```
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >holdBy</rdfs:label>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble-
has-Reference"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#JobService-
canbemanaged-SlaTemplate">
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-JobService"/>
    </owl:inverseOf>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#JobService"
/>
    <rdfs:label>canbemanaged</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-SlaManagedService">
    <rdfs:label>canmanage</rdfs:label>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaManagedS
ervice"/>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaManagedServ
ice-canbemanaged-SlaTemplate"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#DataService-
canbemanaged-SlaTemplate">
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-DataService"/>
    </owl:inverseOf>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#DataService
"/>
```

```
    <rdfs:label>canbemanaged</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Sla-hasUsage-
Usage">
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Usage"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Sla"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasUsage</rdfs:label>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Usage-
isUsageOf-Sla"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Usage-
isUsageOf-Sla">
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Sla"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >isUsageOf</rdfs:label>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Usage"/>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Sla-
hasUsage-Usage"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Reference-
hasReferenceChild-Reference">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasReferenceChild</rdfs:label>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Reference-
hasReferenceParent-Reference"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-JobService">
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#JobService-
canbemanaged-SlaTemplate"/>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#JobService"
/>
```

```
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:label>canmanage</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Sla-
derivedFrom-SlaTemplate">
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Sla"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >derivedFrom</rdfs:label>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
derives-Sla"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Account-owned-
MegaAccount">
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#MegaAccount-
owner-Account"/>
    </owl:inverseOf>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#MegaAccount
"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Account"/>
    <rdfs:label>owned</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-SlaManagedResource">
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaManagedR
esource-canbemanaged-SlaTemplate"/>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaManagedR
esource"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:label>canmanage</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble-
has-Reference">
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >has</rdfs:label>
```

```
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference-
holdBy-ReferenceAble"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
derives-Sla">
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Sla"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >derives</rdfs:label>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Sla-
derivedFrom-SlaTemplate"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble-
hasParent-ReferenceAble">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasParent</rdfs:label>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble-
hasChild-ReferenceAble"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Reference-
hasReferenceParent-Reference">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasReferenceParent</rdfs:label>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference"/
>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Reference-
hasReferenceChild-Reference"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate-
canmanage-DataService">
```

```
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#DataService
-canbemanaged-SlaTemplate"/>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#DataService
"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <rdfs:label>canmanage</rdfs:label>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#Registry-
contains-ReferenceAble">
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Registry"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >contains</rdfs:label>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le-containedIn-Registry"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagerResourc
e-manages-ManagedResource">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >manages</rdfs:label>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagedReso
urce"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagerReso
urce"/>
    <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagedResourc
e-managedBy-ManagerResource"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#SlaManagedServ
ice-canbemanaged-SlaTemplate">
    <rdfs:label>canbemanaged</rdfs:label>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaManagedS
ervice"/>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
"/>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#SlaTemplate
-canmanage-SlaManagedService"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ManagedResourc
e-managedBy-ManagerResource">
```

```
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagerReso
urce"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagedReso
urce"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >managedBy</rdfs:label>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ManagerReso
urce-manages-ManagedResource"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#ReferenceAble-
hasChild-ReferenceAble">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasChild</rdfs:label>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le"/>
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#ReferenceAb
le-hasParent-ReferenceAble"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty
rdf:about="http://www.gria.org/RegistryDomainModel.owl#MegaAccount-
owner-Account">
    <owl:inverseOf
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Account-
owned-MegaAccount"/>
    <rdfs:range
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#Account"/>
    <rdfs:domain
rdf:resource="http://www.gria.org/RegistryDomainModel.owl#MegaAccount
"/>
    <rdfs:label>owner</rdfs:label>
  </owl:ObjectProperty>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#minCardinality"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#onProperty"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#subPropertyOf"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#label"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#subClassOf"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#samePropertyAs"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#comment"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#versionInfo"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#range"/>
```

```
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#domain"/>
  <rdf:Property
rdf:about="http://www.owl.org/2001/03/owl+oil#cardinality"/>
</rdf:RDF>
```

Client CallHelloService.java

```
package hellogria.client;

import java.net.URL;
import java.lang.Object;
import uk.ac.soton.itinnovation.grid.service.types.AddressType;
import java.util.GregorianCalendar;
//import com.ebay.soap.eBLBaseComponents;
import org.apache.axis.utils.XMLUtils;
import org.w3c.dom.Document;
import java.io.*;
import uk.ac.soton.ecs.iam.grid.comms.client.StateRepository;
import
uk.ac.soton.ecs.iam.grid.client.staterepos.MemoryStateRepository;
//import uk.ac.soton.ecs.iam.grid.comms.client.RemoteHelloService;
import uk.ac.soton.itinnovation.grid.types.ConversationID;
import java.io.File;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;



import uk.ac.soton.ecs.iam.grid.comms.client.DataConversation;
import uk.ac.soton.ecs.iam.grid.comms.client.JobConversation;
import uk.ac.soton.ecs.iam.grid.comms.client.SLAConversation;
import uk.ac.soton.ecs.iam.grid.comms.client.SLATemplateConversation;
import
uk.ac.soton.ecs.iam.grid.comms.client.TradeAccountConversation;
import uk.ac.soton.ecs.iam.grid.comms.client.RemoteJobService;
import uk.ac.soton.ecs.iam.grid.comms.client.RemoteSLAService;
import
uk.ac.soton.ecs.iam.grid.comms.client.RemoteTradeAccountService;
import uk.ac.soton.itinnovation.grid.service.types.JobStatus;
import uk.ac.soton.itinnovation.grid.service.types.SLAProposal;
import uk.ac.soton.itinnovation.grid.service.types.SLATemplate;
import org.apache.axis.message.addressing.EndpointReferenceType;

public class CallHelloService {

      private static String JOB_SERVICE_ENDPOINT =
"https://dimitra.netmode.ntua.gr:8443/hellogria/services/HelloService
";
      /** SLA Service Endpoint */
      private static final String SLA_SERVICE_ENDPOINT =
"https://dimitra.netmode.ntua.gr:8443/gria-service-provider-
mgt/services/SLAService";

      private static BufferedReader stdin = new BufferedReader(
                  new InputStreamReader(System.in));

      public static void main(String[] args) throws Exception {

            //Proxy stin HelloService
```

```
            StateRepository repository = new MemoryStateRepository();
            RemoteHelloService helloservice = (RemoteHelloService)
repository
                        .getOrCreateObject(RemoteHelloService.class,
ConversationID
                                    .getEPR(JOB_SERVICE_ENDPOINT));


            //create a proxy to the RemoteTradeAccount

            EndpointReferenceType tradeAccountServiceEPR =

      ConversationID.getEPR("https://dimitra.netmode.ntua.gr:8443/gri
a-service-provider-mgt/services/TradeAccountService");
            RemoteTradeAccountService tradeAccountService =
                  (RemoteTradeAccountService)
repository.getOrCreateObject(RemoteTradeAccountService.class,tradeAcc
ountServiceEPR);
            //EndpointReferenceType
accountepr=tradeAccountService.getEndpointRef();
            TradeAccountConversation tradeAccount=null;
            System.out.println("Do you want to open a new account?
(y/n)");
            String ans=stdin.readLine();

            if (ans.equals("y")){


            //Open a new account
            System.out.println("Dwse ta stoixeia tou account: ");
            System.out.print("Holder Name: ");
            String accountname = stdin.readLine();
            //System.out.println(accountname);

            System.out.print("Holder Telephone: ");
            String accountphone = stdin.readLine();
            //System.out.println(accountphone);

            System.out.print("Holder e-mail: ");
            String accountmail = stdin.readLine();
            //System.out.println(accountmail);

            System.out.println("Client-Organisation Address: ");
            AddressType accountaddress= new AddressType();
            System.out.print("City: ");
            String city=stdin.readLine();
            System.out.print("Country Code: ");
            String countryCode=stdin.readLine();
            System.out.print("Postal Code: ");
            String postalCode=stdin.readLine();
            System.out.print("Region: ");
            String region=stdin.readLine();
            System.out.print("State: ");
            String state=stdin.readLine();
            System.out.print("Street: ");
            String street=stdin.readLine();
            System.out.print("Town: ");
            String town=stdin.readLine();

            accountaddress.setCity(city);
            accountaddress.setCountryCode(countryCode);
```

```
            accountaddress.setPostalCode(postalCode);
            accountaddress.setRegion(region);
            accountaddress.setState(state);
            accountaddress.setStreet(street);
            accountaddress.setTown(town);
            //System.out.println(accountaddress);

            System.out.print("Credit details: ");
            String accountcreditdetails= stdin.readLine();
            //System.out.println(accountcreditdetails);

            System.out.print("Label: ");
            String accountlabel= stdin.readLine();
            //System.out.println(accountlabel);

            //zitame tin dimiourgia enos neou account

      //int i=0;
            TradeAccountConversation tradeAccount2 =
tradeAccountService.openAccount(accountname,accountphone,accountmail,
accountaddress,accountcreditdetails, accountlabel);
            //String accountstatus=tradeAccount.getAccountStatus();
            String accountstatus=null;
      do{
            accountstatus = tradeAccount2.getAccountStatus();
      }while(!(accountstatus.equals("open")));

 tradeAccount=tradeAccount2;
            }
            else {
                  TradeAccountConversation tradeAccount1=null;
                  System.out.println("Give the name for your account:
");
                  String givenaccount=stdin.readLine();
                  EndpointReferenceType[] accounteprs =
tradeAccountService.getResources();
                  //EndpointReferenceType tradeAccountEPR = null;
                  for (EndpointReferenceType epr : accounteprs){
                        String
labelaccount=ConversationID.getLabel(epr);
                        System.out.println("The label of the account
is: "+labelaccount);
                        if(labelaccount.equals(givenaccount)){
                              tradeAccount1 =
repository.getOrCreateObject(TradeAccountConversation.class,epr);
                              EndpointReferenceType
tradeaccount1epr=tradeAccount1.getEndpointRef();
                              System.out.println(tradeAccount1);
                        break;
                        }

                  }
            tradeAccount=tradeAccount1;
            EndpointReferenceType
tradeaccountepr=tradeAccount.getEndpointRef();
            System.out.println(tradeaccountepr);

            }


      //Create a proxy to the SLA Service
```

```
      RemoteSLAService slaService =
(RemoteSLAService)repository.getOrCreateObject(RemoteSLAService.class
,ConversationID.getEPR(SLA_SERVICE_ENDPOINT));
      SLAConversation sla = null;

      System.out.println("Do you want to create a new SLA?(y/n)");
      String ans1 = stdin.readLine();
      System.out.println("answer is :"+ans1);

      if (ans1.equals("y")){

      EndpointReferenceType[] eprs = slaService.getResources();

      /*
       * psaxnei gia SLA templates gia na ta proteinei kai na ftiaxei
SLA*/
      //int i =0;
      EndpointReferenceType slaTemplateEPR = null;
            for (EndpointReferenceType epr : eprs){

            String resourceType =
ConversationID.getResourceType(epr);
            String labelresource = ConversationID.getLabel(epr);

            System.out.println("Searching for SLA's templates to
use");
            System.out.println(resourceType);
            System.out.println(labelresource);
            }

            System.out.println("Give the name of the sla-template you
want: ");
            String slatemplatename=stdin.readLine();
            for (EndpointReferenceType epr : eprs){


                  String resourceType =
ConversationID.getResourceType(epr);
                  String labelresource =
ConversationID.getLabel(epr);

                  //System.out.println("Searching for SLA's templates
to use");
                  System.out.println(resourceType);

           if (resourceType.equals("http://www.it-
innovation.soton.ac.uk/grid/resource/sla-
template")&&(labelresource.equals(slatemplatename))){
            slaTemplateEPR = epr;
            System.out.println(epr);
            break;
      }
            }


            //propose to sla-template pou dialexame
            SLATemplateConversation slaTemplateConversation =
repository.getOrCreateObject
(SLATemplateConversation.class,slaTemplateEPR);
            SLATemplate slaTemplate =
slaTemplateConversation.getSLATemplate();
```

```
            System.out.println("You chose the SLA-template:
"+slaTemplateConversation);
            SLAProposal slaProposal = new SLAProposal();
            slaProposal.setSlaTemplate(slaTemplate);
            slaProposal.setStartTime(new GregorianCalendar());

            System.out.println("Give a name for the new SLA: ");
            String newslaname=stdin.readLine();
            //dimiourgia tou SLA
            EndpointReferenceType eprsla =
slaService.createSLA(slaProposal,newslaname);


            //SLAConversation sla = null;
      //sla=slaService.createSLA(accountepr1,slaProposal,"SLA for
Hello Gria Package4");



      sla=(SLAConversation)repository.getOrCreateObject(SLAConversati
on.class,eprsla);
            //}
            if(sla == null){
                  System.out.println("No SLA's Created");
                  System.exit(1); //termatizei
            }//else if  (sla.getName().equals(defaultsla))
            else{
                  System.out.println("New SLA Created '"+sla+"'");

                  }

      }
      else {
            //psaxnei gia SLAs pou einai agreed oxi gia SLA templates
            EndpointReferenceType[] eprs = slaService.getResources();

            String defaultsla = null;
            System.out.println("Searching for SLA's to use");
            for(EndpointReferenceType epr : eprs){
                  //System.out.println(epr);
                  //System.out.println("hi");
                  String
type_resource=ConversationID.getResourceType(epr);
                  String classname=SLAConversation.class.getName();
                  String labelresource =
ConversationID.getLabel(epr);
                  System.out.println("Type resource is
"+type_resource);
                  System.out.println("label resource is
"+labelresource);
                  defaultsla= slaService.getDefaultSLA();
                  System.out.println("Default SLA  is "+defaultsla);
                  //if (type_resource.equals ("null"))
{System.out.println("null resource type!");}

                  System.out.println("Class name is "+classname);
                  //if (classname.equals (null))
{System.out.println("null classname!");}
                  //if(type_resource.equals(classname)){
            }
            System.out.println("Give the name for the SLA you want");
```

```
            String slaname=stdin.readLine();

            for(EndpointReferenceType epr : eprs){
                  //System.out.println(epr);
                  //System.out.println("hi");
                  String
type_resource=ConversationID.getResourceType(epr);
                  String classname=SLAConversation.class.getName();
                  String labelresource =
ConversationID.getLabel(epr);
                  System.out.println("Type resource is
"+type_resource);
                  System.out.println("label resource is
"+labelresource);
                  defaultsla= slaService.getDefaultSLA();
                  System.out.println("Default SLA  is "+defaultsla);
                  //if (type_resource.equals ("null"))
{System.out.println("null resource type!");}

                  System.out.println("Class name is "+classname);
                  //if (classname.equals (null))
{System.out.println("null classname!");}
                  //if(type_resource.equals(classname)){


      if(labelresource.equals(slaname)&&(type_resource.equals("http:/
/www.it-innovation.soton.ac.uk/grid/resource/sla"))){

                        System.out.println("hi2");
                        sla =
(SLAConversation)repository.getOrCreateObject(SLAConversation.class,e
pr);
                        //break;
                  //System.out.println("hi2");
                  }
            }

            if(sla == null){
                  System.out.println("No SLA's Found");
                  System.exit(1); //termatizei
            }//else if  (sla.getName().equals(defaultsla))
            else{
                  System.out.println("SLA Found '"+sla+"'");

            }
      }


      //Create the New HelloResource

      // String name;
            System.out.print("Give a name for the new resource: ");
            String name = stdin.readLine();
            System.out.println(name);
            HelloConversation result = (HelloConversation)
helloservice.createHelloResource(name,sla.getEndpointRef());
            System.out.println(result.HelloWorld());
      }
      }
```

Αναφορές:

[1] I. Foster, “What is the Grid? A Three Point Checklist”, Grid Today, July 2002
[2] SLA available at: http://en.wikipedia.org/wiki/Service_Level_Agreement
[3] EGEE-II DSA2.1 “Institution of SLAs and appropriate policies”
[4] A. Sahai, S. Graupner, V. Machiraju & A. van Moorsel, “Specifying and Monitoring Guarantees in Commercial Grids through SLA”, Technical Report HPL-2003-324, Hewlett-Packard Company, November 2002
[5] The Gria middleware available at:http://www.gria.org
[6] L-O. Burchard & B. Linner, “A Quality-of-Service Architecture for Future Grid Computing Applications”, IEEE Parallel and Distributed Symposium, pp.132a-132a, Denver, Colorado, April 2005
[7] L-O Burchard, M. Hovestadt, O. Kao, A. Keller & B. Linnert, “The Virtual Resource Manager: An Architecture for SLA-aware Resource Management”, IEEE International Symposium in Cluster Computing and the Grid, pp.126-133, Chicago, USA, April 2004
[8] M. Hovestadt: "Fault Tolerance Mechanisms for SLA-aware Resource Management", proc. 11th International Conference on Parallel and Distributed Systems, 2005
[9] D.A Menasce & E. Casalicchio, “A Framework for Resources Allocation in Grid Computing”, IEEE International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunications Systems, pp.259-267, Volendam, Netherlands, October 2004
[10] R. Ranjan, A. Harwood & R. Buyya, “SLA-Based Coordinated Superscheduling Scheme for Computational Grids” Superscheduling Scheme for Computational Grids”, IEEE 8th International Conference on Cluster Computing, Barcelona, Spain, September 2006
[11] C. Dumitrescu & I. Foster, "GRUBER: A Grid Resource SLA Broker", Euro-Par Parellel Processing, Portugal, September 2005
[12] C. Dumitrescu, I. Raicu, & I. Foster, “DI-GRUBER: A Distributed Approach to Grid Resource Brokering”, ACM/IEEE conference on Supercomputing, pp.38, Washington, USA, November 2005
[13] C. S. Yeo & R. Buyya: "Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility", proc. the 7th International Conference on Cluster Computing, (CLUSTER 2005).
[14] K. Czajkowski, I. Foster, C. Kesselman, V. Sander & S. Tuecke: "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems", proc. 8thWorkshop on Job Scheduling Strategies for Parallel Processing, Edinburgh Scotland, July 2002
[15] E. Al-Masri & Q.H. Mahmoud, “QoS-based Discovery and Ranking of Web Services”, IEEE International Conference on Computer Communications and Networks, pp.529-534, Hawai, USA, August 2007
[16] R. J. Al-Ali¤, O. F. Rana & D.W. Walker " G-QoSM: Grid Service Discovery Using QoS Properties", Computing and informatics (Comput. inform.) Special issue on grid computing 2002, vol. 21, no 4 (27 ref.), pp. 363-382
[17] B2B Contextualized Registries available at: http://www.gria.org/documentation/5.2/tutorial/registry-component-tutorial/introduction

[18] M. Surridge, S. Taylor, D. De Roure & E. Zaluska: "Experiences with GRIA-Industrial applications on a Web Services Grid" ,Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05)
[19] Apache Tomcat available at: http://www.tomcat.apache.org
[20] The eXist Database available at: http://exist.sourceforge.net
[21] The Apache Maven available at: http://maven.apache.org