TrustCoM

Deliverable
D63-B

TrustCoM Framework V4 Appendix B: Subsystem Architecture

AL1 – TrustCoM Framework

Michael D. Wilson, CCLRC
Alvaro Arenas, CCLRC
Lutz Schubert, HLRS

31/07/2006

V3.1

TrustCoM

A trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organisations

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2002-2.3.1.9

Networked business and governments

Prepared for the European Commission under FP6 Contract No. 01945 as a deliverable form

LEGAL NOTICE

The following organisations are members of the TrustCoM Consortium:

Atos Origin,
Council of the Central Laboratory of the Research Councils,
BAE Systems,
British Telecommunications PLC,
Universitaet Stuttgart,
SAP AktienGesellschaft Systeme Anwendungen Produkte in der Datenverarbeitung,
Swedish Institute of Computer Science AB,
Europaeisches Microsoft Innovations Center GMBH,
Eidgenoessische Technische Hochschule Zuerich,
Imperial College of Science Technology and Medicine,
King's College London,
Universitetet I Oslo,
Stiftelsen for industriell og Teknisk Forskning ved Norges Tekniske Hoegskole,
Universita degli studi di Milano,
The University of Kent,
International Business Machines Belgium SA .

Neither the TrustCoM Consortium, any member organisation nor any person acting on behalf of those organisations is responsible for the use that might be made of the following information.

The views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect the views of the European Commission or the member organisations of the TrustCoM Consortium.

All information provided in this document is provided 'as-is' with all faults without warranty of any kind, either expressed or implied. This publication is for general guidance only. All reasonable care and skill has been used in the compilation of this document. Although the authors have attempted to provide accurate information in this document, the TrustCoM Consortium assumes no responsibility for the accuracy of the information.

Information is subject to change without notice.

Mention of products or services from vendors is for information purposes only and constitutes neither an endorsement nor a recommendation.

IBM, the IBM logo, ibm.com, Lotus and Lotus Notes are trademarks of International Business Machines Corporation in the United States, other countries or both.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries or both.

SAP is a trademark of SAP AG in the United States, other countries or both.

'BT' and 'BTexact' are registered trademarks of British Telecommunications Plc. in the United Kingdom, other countries or both.

Other company, product and service names may be trademarks, or service marks of others. All third-party trademarks are hereby acknowledged.

Deliverable datasheet

Project acronym: TrustCoM

Project full title: *A trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organisations*

Action Line: AL1

Activity: 1.2

Work Package: WP27

Task:

Document title: **TrustCoM Framework V4 Appendix B: Subsystem Architecture**	
Version:	**v3.1**
Document reference:	
Official delivery date:	**31/01/2007**
Actual publication date:	
File name:	**D63 TrustCoM Framework V4 - Appendix B.doc**
Type of document:	Report
Nature:	official deliverable

Authors:

Reviewers:

Approved by: …

Version	Date	Comments
V3.1	25/12/2007	Updated document structure
V3.2		Minor updates in selected sections

Table of Content

Introduction

Within the following chapters, the subsystems as have been introduced in the main document, shall be explained in more detail with respect to their structure and how the according components interact in order to contribute to the individual lifecycle's functionalities.

I VO Management CCLRC

This section summarizes the architectural requirements and discusses the possible options for TrustCoM VO Management (VOM).

The VO management component is an advance on existing VO Management systems such as the LCG VO membership management system[1] since it contains the additional functionality for contract management and VO lifecycle support. The VO Management subsystem will provide the services necessary for coordinating the VO functionalities across its lifecycle. The VO management component is defined here in terms of its subcomponents, major data objects and interaction with other system components.

There is obvious scope for terminological confusion for the phrase "VO manager". To clarify this some definitions used are:

- VO Initiator:

  The responsible *person* initiating (and steering) the creation of a VO – this person may also verify the status of a member of the VO by performing user entries, assignment of roles, information updates and user removals. The VO management function can be performed by a group of persons delegated by the VO Initiator. Note that the VO Intiator for a VO can change during the lifecycle of the VO. A single person can be VO Initiator to one or more VOs. A person becomes VO Initiator by notifying the TrustCoM system (the VO Management service) of the intention to create a VO – even though the VO itself does not legally exist until a General VO Agreement (defined below) has been agreed (this is ontologically confusing since the person manages a VO that does not exist, but they are managing a VO which has been proposed, and there is no requirement yet to differentiate the manager of a VO proposal from the manager of a VO itself).´

- VO Management Component (also VO Management Service):

  The actual service providing the management functionalities and enacting them in the Virtual Organisation. It is the "representative" of the VO Initiator inside the VO. This is the component described in this section of the document.

- VO Management Organisation:

  The organisation to which the VO Initiator and/or VO Management Service belongs to. The person who is VO Initiator for a VO can change organisation while remaining the VO Initiator, and can remain at the same organisation and change status from a VO manager to a previous VO Initiator for a names VO. Also, the organisation hosting the VO Management service may alter from the one the VO Initiator belongs to.

- VO Membership Management Module:

  One of the major modules of the VO component which is defined below.

[1] Kelsey, D. (2004) Requirements for LHC Computing GRiD (LCG) User Registration and VO Membership Management. www.**grid**pp.ac.uk/tier2/LCG_User_Registration2004.pdf, and see LCG VO User Registration - http://lcg.web.cern.ch/LCG/users/registration/VO.html

- VO partner/member:

an organisation that is a member of the VO

- VO partner/member manager (also SP administrator):

The person responsible for a VO member within a VO. Different VO member managers can manage different VO's for the same VO member organisation.

- VO partner/member staff:

A person, not different for VO management organisation. From the VO perspective, this may be regarded as a plain resource.

- VO/EN partner profile:

Collection of services that a partner is willing in an EN or expected in a VO to perform. Included in the EN Agreement.

- VO/EN partner details:

Organisation details of name, address etc. – see the UDDI business entity class.

I.1 Components

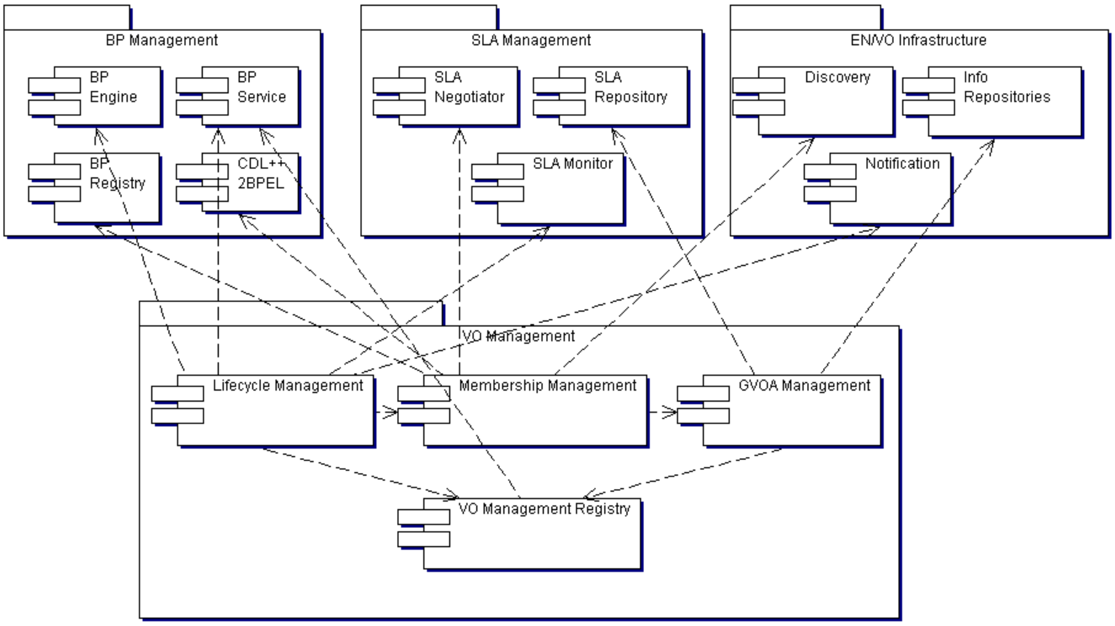

Figure 1: VO Management Component Overview.

The descriptions of the components are below, so that the design decisions will become more clear. During the design process the Role Management function has been considered within the VO manager, but following an analysis of the interactions required, this has been placed within the business process model manager, along with responsibility for Choreography support.

I.1.a Lifecycle Management

This module is a system guide of VO Management activities, based on the lifecycle defined by the VO Manager.

The VO lifecycle has been outlined in the main document, and is "*Identification → Formation → Operation → Dissolution"*.

To be a candidate member of a VO, an organisation needs to join an Enterprise Network. Therefore there is a need to have Enterprise Network Agreement which will:

- Permit – the organisation to be a member of the EN and grant access to EN resources, and permit that organisation to describe itself in terms of its capabilities which must be stored in the EN repository for use at the VO formation stage (to match against roles in a VO Collaboration Description).
- Require – that the organisation acts by the rules of the EN.

The Enterprise Network Agreement will act as a stage before the General VO Agreement on which it will build.

A Dynamic VO is a co-operation within a subset of EN members. Specific objectives and market needs trigger the establishment and operation of the dynamic organisation. An EN provides the infrastructure to rapidly set up new VOs:

- The EN may be static but the VOs can be dynamic
- Participation in an EN shows disposition to create VOs and offers infrastructure support for creating VOs but EN is not a VO

Below in Figure 2 the interrelations between the outlined concepts are depicted. As we can see, an enterprise may participate in more that one VO at any given point of time, delegating appropriate resources (via virtualised services) and playing different roles, according to its policies and those of the VOs the enterprise is involved in.

The VO process management, ensures that the members of a *VO play by the rules agreed by everyone involved and that members' behaviour is observable*, thus allowing to enforce these rules for common business need. In order to define the necessary services for the VO management we need to identify some other key concepts. As indicated earlier, we can perceive a VO as composition and interaction of three main components:

- The *collaboration agreement*, also called General VO Agreement (GVOA) - contracts that express the general rules each partner of a VO must abide to, in order to be acceptable as a member of the VO.
- The SLA for each role in the business process of the VO
- The participants who each fulfil a role in the VO
- The business process model which defines business of the VO, and the roles available for each partner.

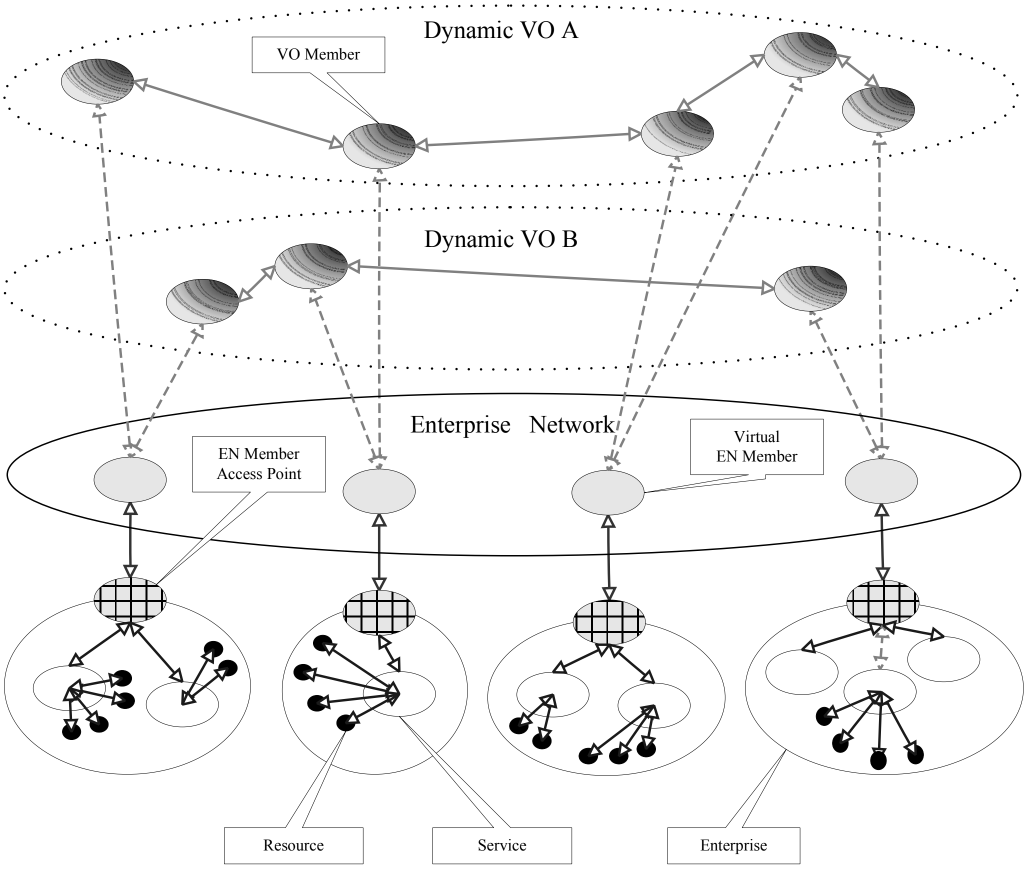

Figure 2: Relationship between the enterprises, enterprise networks and VOs.

Following the template-based approach for business processes and SLAs adopted by TrustCoM, it is envisaged that the initiator of a VO chooses a GVOA template, instantiates it and publishes its intention to build a VO on this basis.

The GVOA identifies a set of business process with a set of roles involved in the enactment of those processes. It will also list policies and SLA templates to introduce (non-functional) constraints on roles and on the enactment of business processes. Furthermore, a GVOA may specify one or more operational business processes in accordance with the business objectives that support the creation of the VO.

The roles identified in the GVOA must be defined in a Collaboration Description (CD) which states which resources those roles have access to. In turn the security policies for accessing those resources can later be derived from policy templates in the SLAs, the roles and resources from the CD.

When joining the EN, an organisation has declared the set of capabilities that it has. These will be mapped to the set of roles defined in the CD to determine if an organisation is a candidate member for a VO.

When an EN member declares its interest in participating in the future VO, it must indicate the roles it may be willing to assume, respectively *can* assume. Depending on the negotiation model, a stakeholder may also propose policy changes, trigger the deployment of new business processes, etc. Up to that point, the GVOA is considered to be non-effective.

Just as a general contract may need to be revised if the existing contract is detected to be unsatisfactory, the GVOA could possibly be re-negotiated at different points in time. One need that must be addressed is the consequences of modifying the set of partners, or modifying role-assignment of partners in an existing VO. Hence, in the defined VO Management process for VO modification, the GVOA is one of the components that are treated according to well-defined rules.

Consequently the management of the VO is closely linked to the SLA management and the BP management.

Subscription to :

- SLA Management
  - subscribe to SLA evaluator to be informed of SLA Violations such as failure to meet deadlines or desired QoS parameters.
  - Notify VO Manager, VO Partner Managers, or take action defined in rules when SLA conditions are breached; ultimately replacing an existing partner in a role.

- Reputation
  - input to the reputation system at the end of each business activities by subscribing to the BPM service and pass quality measures to reputation service ;
  - monitor reputation value of partners and act if it drops below a threshold.

When there is a violation the VO manager and VO partner manager will be informed. The VO manager can include automatic rules for action on the violation of policies, SLA conditions, or decline in reputation in the general VO Agreement which are implemented here.

I.1.b Membership Management

This module is responsible for the addition, modification and removal of VO-Members in either an active or persisted VO. It builds upon the Enterprise Network Infrastructure which stores details of those organisations which are members of the Enterprise Network and are therefore eligible to join any VO. It invokes the Discovery tool to identify potential VO members, then calls the SLA negotiator to choose the best partner for a role, and to negotiate the detailed SLA.

I.1.c GVOA Management

The General VO Agreement Manager hosts the General VO Agreement for each VO. It manages the creation of this during the initiation of the VO, and the evolution of this during the operation of the VO as partners join or leave, and as new SLA are defined for partners as they change roles. It mainly interfaces to the SLA Management Subsystem which creates the details for the SLA and monitors them, and to the BP Manager which creates the details of the Collaboration Model through a WS-CDL specification, and executes it. The GVOA Manager generates partner profiles based on these pieces of information.

A major sub-component of the GVOA Manager is the Collaboration Manager: the component responsible for managing the consistency of the collaboration definition and the other VO Management Services. The important role is to notify the Lifecycle, Membership and GVOA managers when the GVOA and SLA's change as the VO evolves.

The registry will store clauses for the VO Agreement defined below. Each clause is described both in natural language and, where applicable, in machine readable language in order to allow the policies to be enforced.

General VO Agreement components:

- VO template – a general set of terms and conditions that apply to all VO agreements
- Partner Details – details of the legal entity that is the partner
- EN organisation identifier – reference to the Enterprise network agreement for the partner that overrides the VO Agreement
- Objective and Role of Partner – defined as annotations to the CD
- BPM Definition – a pointer to the CD created for the VO as context for the Role definition
- VO Constraint – Constraints that apply to the whole VO, rather than an individual partner – e.g. automatic rules for action on the violation of policies, SLA conditions, or decline in reputation.
- Initiation and Termination Conditions – conditions for the BPM to begin before the Choreography can be initiated and that the VO cease operating when the Choreography has met them.
- Legal Issues section – Terms and Conditions for legal issues drawn from a VO model agreement
- Policies – General policies applying to the whole VO which are inherited by all SLA's leading to Trust Security & Contract Roles (see section 3.2.1.3) that result in Policy enforcement & decision points.
- References to SLA for the VO which include policies applying to the individual functional and non-functional roles and work units leading to Trust Security & Contract Roles (see section 3.2.1.3) for the non-functional roles that result in Policy enforcement & decision points.

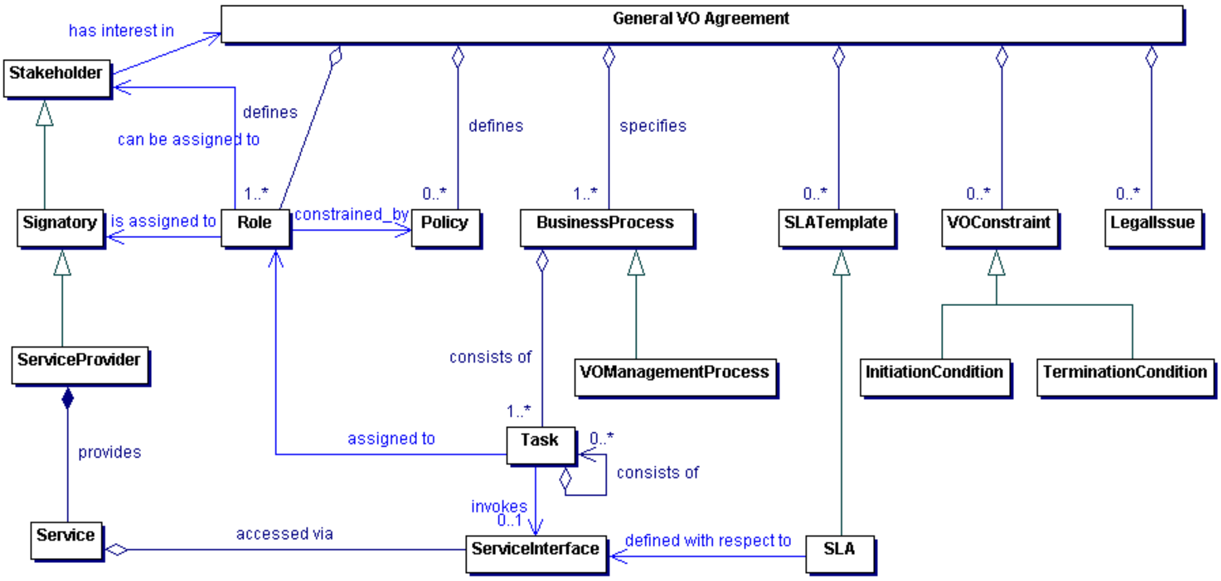

Figure 3: Static model of General VO Agreement

I.1.d VO Management Registry

The registry is a simple relational database to be accessed by the VO management component. The simple data structures are not all defined here to save space. Access is permitted for create/read/write from within the VO management component, and read only through Web Service interface to the rest of the Architecture. Write access for the rest of the architecture is only permitted through the interfaces to the modules within the VO management component. The VO Management Registry is not a UDDI Registry, merely a relational database used to store data.

I.2 Interaction Scenarios

The three main phases of the VO life cycle are considered for the operation for the VO management component and its interaction with other components.

I.2.a Identification *and* Formation

Steps in Identification and Formation:

1) Organisation registers its identity and available services with the EN infrastructure
2) An EN member wishes to create a VO and registers in the VO lifecycle tool to create a new VO. VOM calls EN Infrastructure to identify the VO manager. The VO lifecycle tool registers in its registry a VO identifier.
3) The VO lifecycle tool calls the BP Manager, passing the VO identifier, to allow the VO manager to define a BPM for the VO. The BP manager returns a CD, including:
    a. definitions of the roles of potential VO members
    b. choreography descriptions
    c. policies (TSC Roles – see section 3.2.1.3) associated with each role

   d. initiation conditions for the BP
   e. termination conditions for the BP.
   f. QoS requirements
   g. Trustworthiness per role

4) VO lifecycle tool calls the VO membership management tool, passing the VO identifier.

5) Membership Management tool invokes the Discovery Service (see section 3.6.1.3) passing the VO identifier. The Discovery Service will identify partners for the VO that match the role descriptions and constraints obtained and stored by the BP Manager (step 3 above). The Discovery Service returns identifier triples for:

   a. The role in the CD for the VO,
   b. The EN organization identifier
   c. The registered EN service identifier

   There may be more than one identifier EN candidate per role (ENBusinessEntity class, which extends the BusinessEntity datatype specified in the UDDI schema). The Membership manager stores in its registry the triples.

6) Cycling through the BPM roles, for each role the Membership management tool calls the SLA negotiator (section 3.3.1.4) passing it the VO identifier and the appropriate set of triples from the discovery service for each role. The SLA negotiator returns the triple with an identifier for the agreed SLA for each role. The SLA negotiator needs to choose the best potential partner for each role as part of the negotiation process. The Membership manager stores in its registry the SLA identifier for each role.

The process describe till this point corresponds to the identification phase. A summary of such process is presented in Figure 4

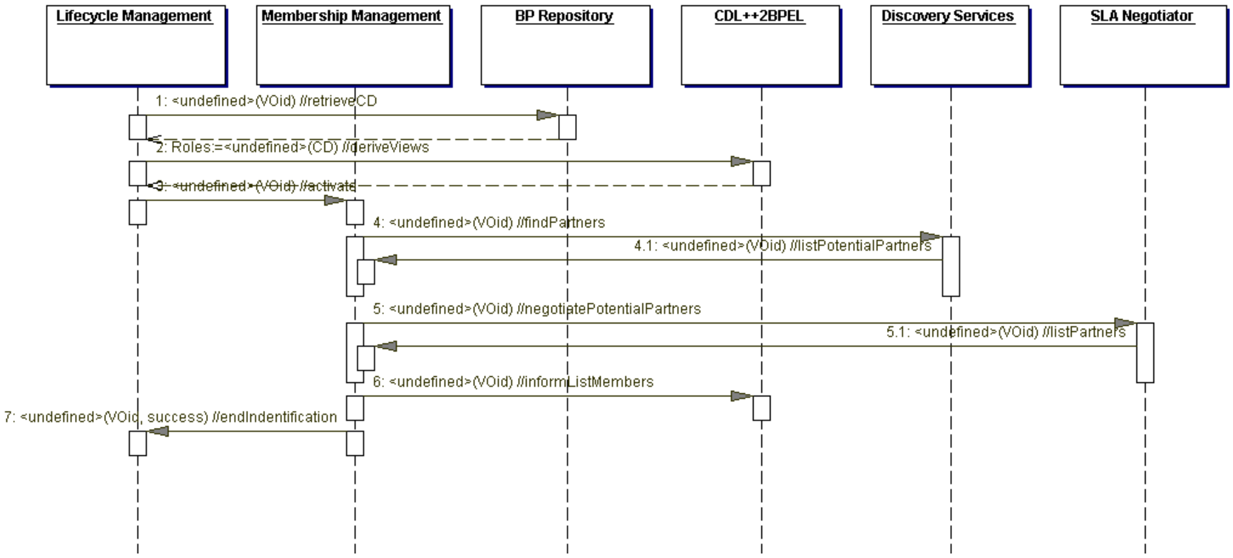
Lifecycle Management
Membership Management
BP Repository
CDL++2BPEL
Discovery Services
SLA Negotiator
1: <undefined>(VOid) //retrieveCD
2: Roles:=<undefined>(CD) //deriveViews
3: <undefined>(VOid) //activate
4: <undefined>(VOid) //findPartners
4.1: <undefined>(VOid) //listPotentialPartners
5: <undefined>(VOid) //negotiatePotentialPartners
5.1: <undefined>(VOid) //listPartners
6: <undefined>(VOid) //informListMembers
7: <undefined>(VOid, success) //endIndentification

Figure 4: Sequence Diagram for the VO Management in the Identification phase

7) The Membership manager now calls the VO Agreement Manager to create a VO agreement passing a VO identifier. The VO Agreement manager returns success or failure.

8) The VO Agreement Manager:

    a. retrieves the VO agreement template from the VO Management Registry

    b. Call the BP manager passing the VO identifier and retrieve the general VO policies that apply to all partners, the initial conditions and termination conditions for the BP.

    c. for each role stored in the registry for the VO, the VO Agreement manager will:

        i. Call the EN Infrastructure passing the EN identifier to retrieve the partner identity details from the partner profile.

        ii. Call the BP Manager passing the VO identifier and role identifier and retrieve the role objective and description, and the policies applicable to that partner in that VO.

        iii. SLA negotiator to retrieve the instance of the negotiated SLA for each partner in each role.

    d. Generate an XML document for the VO agreement and store it in the registry.

    e. Issue the document to each VO partner for signature, storing the returned signed copies from each partner.

    f. Return to the Membership manager acknowledging that an agreement has been reached and stored.

9) The Membership manager returns to the VO lifecycle manager that the VO membership has been created.

10) The VO lifecycle manager registers all services that can be called in the VO at the *Service Instance Registries* and the *Information Repositories* in the EN Infrastructure.

11) The VO lifecycle manager has completed the Identification and Formation stage and initiated Operation and Evolution Stage.

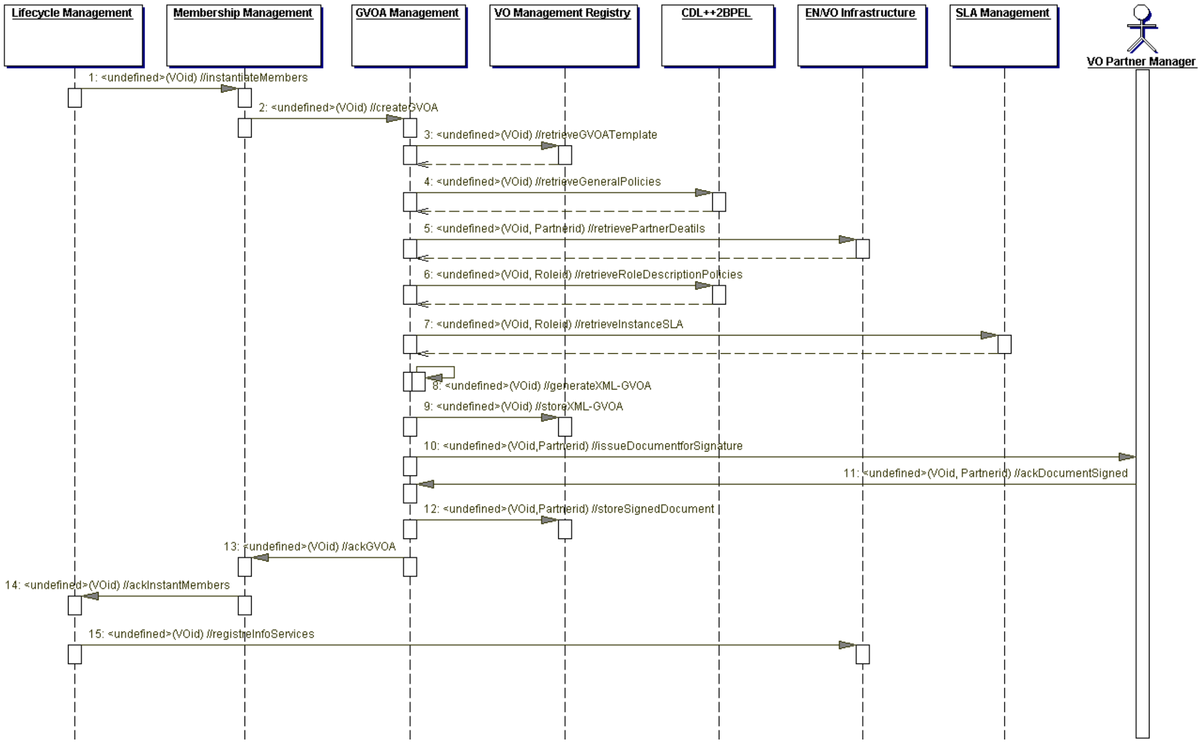
Lifecycle Management
Membership Management
GVOA Management
VO Management Registry
CDL++2BPEL
EN/VO Infrastructure
SLA Management
VO Partner Manager
1: <undefined>(VOid) //instantiateMembers
2: <undefined>(VOid) //createGVOA
3: <undefined>(VOid) //retrieveGVOATemplate
4: <undefined>(VOid) //retrieveGeneralPolicies
5: <undefined>(VOid, Partnerid) //retrievePartnerDeatils
6: <undefined>(VOid, Roleid) //retrieveRoleDescriptionPolicies
7: <undefined>(VOid, Roleid) //retrieveInstanceSLA
8: <undefined>(VOid) //generateXML-GVOA
9: <undefined>(VOid) //storeXML-GVOA
10: <undefined>(VOid,Partnerid) //issueDocumentforSignature
11: <undefined>(VOid, Partnerid) //ackDocumentSigned
12: <undefined>(VOid,Partnerid) //storeSignedDocument
13: <undefined>(VOid) //ackGVOA
14: <undefined>(VOid) //ackInstantMembers
15: <undefined>(VOid) //registreInfoServices

Figure 5: Sequence Diagram for the VO Management in the Formation phase

I.2.b Operation and Evolution

The VO lifecycle manager calls the BP Manager to initiate the operational phase of the VO. The VO lifecycle manager awaits the following alerts to act:

1) Each completed Transaction
    a. The BP Manager calls the VOM to declare that a BP transaction has been completed
    b. The VOM calls the reputation manager to update reputation information passing:
        i. VO
        ii. Partner id
        iii. the organisational unit of the partner
        iv. Role (context)
        v. 50 or so performance attributes defined by the SLA
2) Partner defaulted on policy from SLA Manager.
    a. Call VO Membership Manager to replace partner.
        i. Call VO Membership Manager to initiate Termination phase for this member

ii. Membership Manager follows step 5-10 for Identification and Formation

3) BP has completed from the BP Manager.
   a. Call VO Membership Manager to initiate Termination phase for all members.
4) A member of the VO has completed its roles in the VO from the BP Manager.
   a. Call VO Membership Manager to initiate Termination phase for this member.
5) The VO manager wishes to modify the VO, either the Business Process or policies.
   a. call BPM to modify BP
   b. call SLA negotiator to renegotiate SLA
   c. revise VO Agreement
6) The VO manager wishes to dissolve the VO
   a. Call VO Membership Manager to initiate Termination phase for all members
7) The VO manager wishes to change the VO structure in one of the following ways:
   a. Change a partner
   b. Change the role allocation between partners
   c. Change timescales within the BP
   d. Change costs within the BP
   e. Change policies in the VO Agreement
   f. Change the structure of the BP by dividing processes and re-aligning dependencies.

I.2.c Dissolution and Termination

VO Membership Manager enacts the termination conditions on the VO agreement.

When each partner is removed from the VO the VO Membership Management component will call the BP Management and Policy Service to gather information about the performance of the partner, Reputation Management Service to update the reputation of that partner.

I.3 Dependencies Overview

I.3.a EN Infrastructure

Organisations register to join the EN by making an EN agreement. Also called to identify EN members, and collect their details.

The VO lifecycle manager registers all services that can be called in the VO at the *Service Instance Registries* and the *Information Repositories* in the EN Infrastructure.

I.3.b BP Management

Called to define the BP and return role details, termination and initiation conditions on the BP, and to enact the business process in the operation phase of the VO. Called to gather information about the performance of a partner for the reputation service.

I.3.c EN Infrastructure - Discovery Service

Called to identify partners for the VO that match the role descriptions and constraints obtained and stored by the BP Manager

I.3.d SLA Management - SLA negotiator

Called to negotiate SLA with potential partners, and return SLA for inclusion in GVOA.

I.3.e SLA Management - SLA evaluator

Informs VO manager about a partner defaulting on a policy in an SLA.

I.3.f Reputation Service

During the operation phase:

- input to the reputation system at the end of each business activities by subscribing to the BPM service and pass quality measures to reputation service ;
- monitor reputation value of partners and act if it drops below a threshold.

Called during the termination phase for each partner to update that partner's reputation information.

I.3.g Policy Services

Read the VO Management Registry to retrieve machine readable policies from the VO Agreement. Called to gather information about the performance of a partner for the reputation service.

II Business Process Management SAP

In the section I.4 about TrustCoM's participant model, the operational baseline dealing with collaborative business processes (CBPs) from collaboration definition to executable private/public processes was introduced. The following sections refine the previous content and integrate the security concept for CBPs, called the TSC Concept referring to Trust, Security and Contract Management.

For the operational part, the BP subsystem offers the connecting piece between business applications and the VO management, infrastructure, and security functionality. Given that the business applications are available via service interfaces, TrustCoM's BPM subsystem allows interconnecting the services of multiple parties in a VO while applying trust, security, and contract management to it. Since VOs are formed quickly, on-demand and often include previously unknown business partners, a method for fast application and process integration must be available. The BPM subsystem offers this method while satisfying security requirements as well.

The trust, security, and contract (TSC) concept introduces fine-grained security control mechanisms into the control flow of processes, enabling fine-tuned reaction to security-related issues and compensation at the process level. These security controls invoke the relevant services, e.g., for the confirmation of claims, and use the gathered information to steer the process control flow. By using this method, expensive rollbacks of the CBP can potentially be avoided, e.g., solving process control flow related security conflicts, for instance the missing verification for a certain claim, within the process exception handling. In this case, the exception handler could try to verify the claim, instead of terminating the whole CBP and undoing the performed work as far possible. It is important that the CBP's TSC concept is uniform for the whole VO, so that partners have the same understanding of what credentials are to be provided with which message. A major feature of the TSC concept is the runtime configuration of the security controls inside a running CBP: Long-running business process instances can be adapted to changing conditions and business partners, even after they have been started.

II.1 Components

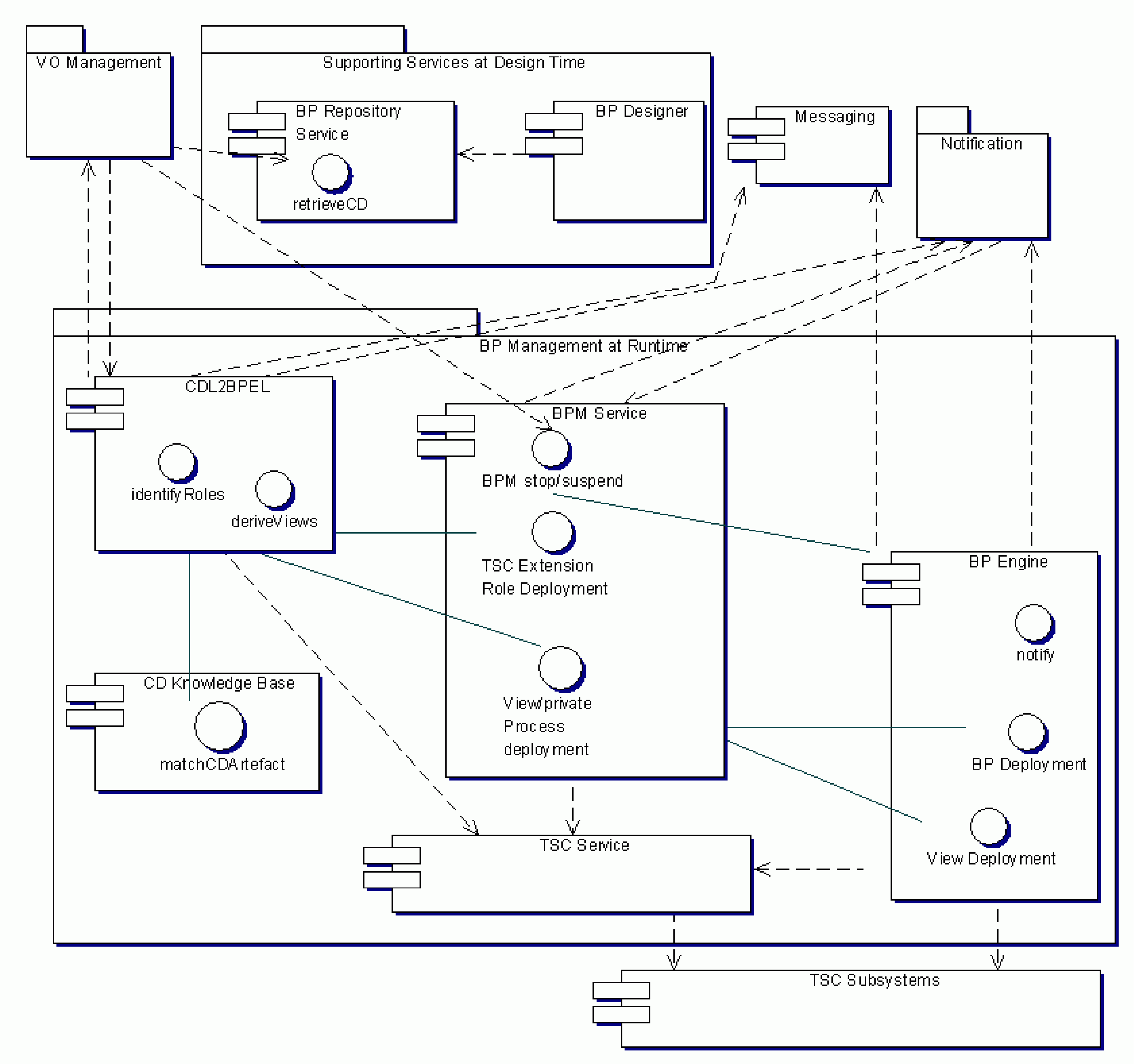

Figure 6: BP subsystem component overview

II.1.a BP Designer

A modelling tool, using a custom UML profile for UML activity diagrams, to model and store collaboration definition templates. In the first instance, a user (usually the VO manager) models processes from scratch, on the level of a choreography description language like WS-CDL. It is anticipated that the UML profile will be expanded to directly cover recurrent, pattern-like behaviour and interactions (e.g. call-for-tender), hereby providing a higher-level description of the collaboration. Also, a library of pre-defined commonly-used collaborations may be defined, thus supporting re-use and quicker definition of new collaborations.

The collaboration definition template defined by the UML model contains:

- the roles participating in the collaboration
- a description of the order of interactions between the participating roles
- an (abstract) description of the information types exchanged in the interactions between the participating roles
- an (abstract) description of additional trust, security and contract management (TSC) requirements that participating roles may demand.
- meta-data about the collaboration

Such a template can be instantiated to a collaboration definition by instantiating the roles, and replacing the information types and the TSC requirements with refined descriptions.

II.1.b BP Repository

The BP Repository offers design time storage and retrieval capabilities for collaboration definition (CD) templates. CDs are retrieved by specifying a list of criteria outlining the business objective, the VO is intended to meet. The BP repository is able to suggest CDs meeting the criteria by matching the CDs meta-data description.

II.1.c CDL++2BPEL Service

The CD contains the global, high-level description about how the VO will meet the business objective. This description has to be realised at runtime by executable BP components, the executable private processes and corresponding views.

The CDL++2BPEL Service addresses this gap by taking VO Members meeting the specified roles as input from VO Management. The service then automatically derives at least process views for each, or just one specific role. The service is also capable of deriving private processes as well, if no private processes fitting the views are available in the VO member's domain, assigned to the specific role. It is recommended that one instance of this service is offered in a VO, possibly by VO Management falling in the category of Choreography services, but it is also possible to run one service instance per VO member domain.

The Service performs its duty in three parsing stages:

1. Immediate mapping of activities and message exchanges to views and optionally BP activities and exchanges.
2. Identification of well known collaboration artefacts, e.g. Purchase Order, and generation of their view activities (and corresponding private BP activities optionally), in general for more than one role.
3. Handling of remaining activities and exchanges, in worst case by raising alerts and using exception handlers from the GVOA.

Views and optionally private processes may be deployed automatically in the role specific VO member domains, using the BPM service.
TSC requirements are already addressed as so-called TSC Extension Roles at this stage. If available at design time, TSC Extension Roles are annotated in the collaboration definition, and corresponding TSC Tasks are inserted according to the role type in the view

or private process. D16, the conceptual models (BP section) elaborates further on TSC requirements and the model behind it.
TSC Extension Roles (see WP21-ID section 4.1 for specification) are intended to capture all information at design time which is necessary for TSC Tasks to perform their enforcement or control duty at runtime. A TSC Extension Role is modelled as a data set containing all the required information to configure a TSC Task. So far, four types of TSC Extension Roles are modelled, each realizing a CBP control for exactly one TSC subsystem:

- Trust Extension Role – Trust and Reputation subsystem
- Security Extension Role – security subsystem
- SLA Extension Role – SLA subsystem
- Monitoring Extension Role – Messaging Subsystem

II.1.d CD Knowledge Base

This service is closely tied to the CDL++2BPEL service and the same domain affiliation as well as deployed service numbers are required. In the second parsing stage of the CDL++2BPEL service (see the CDL++2BPEL component description above), well known CD artefacts are identified and the CD Knowledge Repository Service is queried for view and optionally private process artefacts which are then inserted in the generated role specific subjective views. Such a CD artefact usually corresponds to a confidential part of the private process which may not be exposed to other entities, not even to other VO members.

II.1.e BPM Service

The BPM service provides deployment and runtime control for the BP engine, one in each VO member domain. Views and private processes are deployed, suspended, stopped or tested by using offered service interfaces. The BPM also subscribes to notification topics addressing the BP runtime, e.g. addressing the control of BP instances.

If TSC Tasks need further or altered configurations, TSC Extension Roles can be assigned to a deployed view/private process.

II.1.f BP Engine

The BP engine finally enacts the deployed private process and offer the process view as the externally visible behaviour to the outside. Usually, each VO member runs an engine in his/her own domain. The BP engine, following the service oriented architecture paradigm, is lightweight, implementing a process model with limited capabilities to work with workflow relevant data. The engine rather orders service invocations from within private process tasks providing implemented business logic.

A deployed process may be instantiated upon deployment, only once or in several instances flexibly meeting the requirements imposed by the business objective. A process instance is started by invoking the dedicated activity/method in the corresponding process view with initial start data. Note that a process instance completion is not equivalent to the termination of the VO. In fact, many instances of the CBP may be executed during the operation phase of a VO, but all instances should be completed or terminated before the dissolution phase is started.

II.1.g The TSC Concept

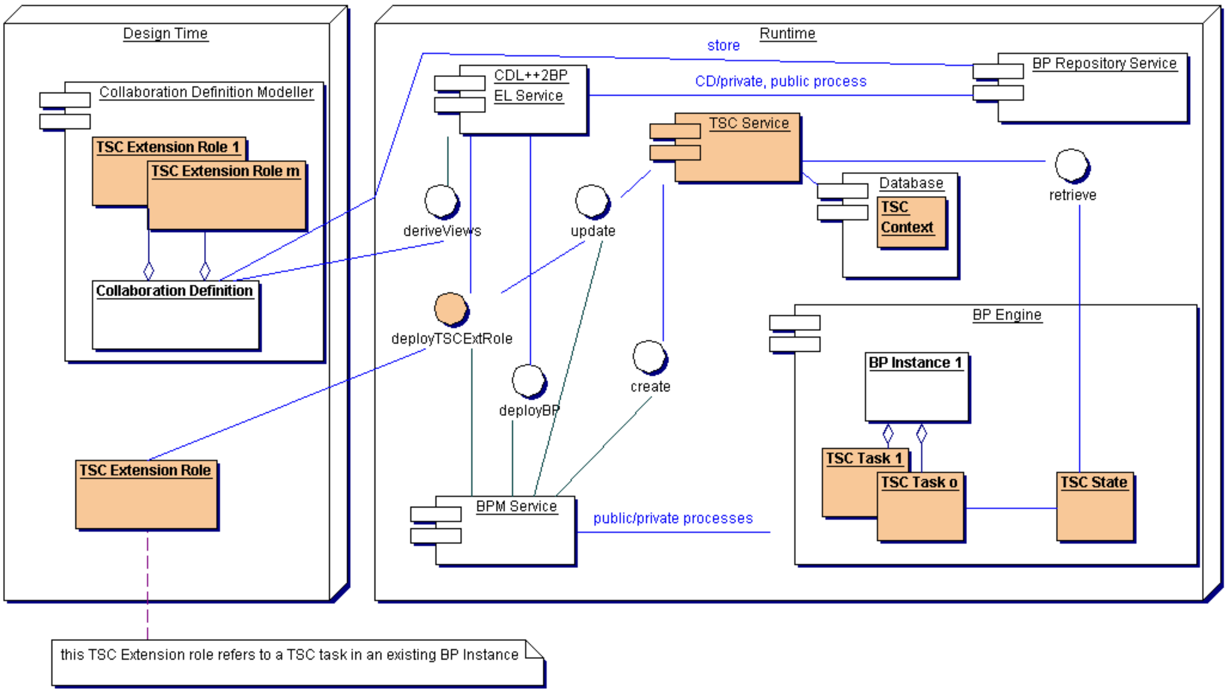

Figure 7: TSC related components

The security concept for CBPs is called the TSC concept. It was formerly in the stage of a purely conceptual model and is in this architecture version mature enough to become part of the BPM subsystem's deployment model. Figure 7: TSC related components shows the components belonging to the TSC concept in a darker colour. The remaining components belong to the above described operational baseline. Only the ones which are important to understand the TSC concept integration are shown again. The TSC concept components are now briefly described. The components are divided in runtime and design time components. The former are relevant for the operational phase of the VO, the latter for formation and identification.

TSC Context

The TSC Context contains all security relevant data for one BP instance. It is therefore stored in a trusted software component, e.g. a database. Since the TSC concept is about the offering of task based security controls for CBPs, the security relevant data entails the set of all security controls for a BP instance.

TSC Task

Security controls have to be enforced during the BP enactment. The BP model needs to offer an intrinsic means to enforce security policies in running process instances. The TSC Task is exactly this enforcement component. Its configuration data is stored in the TSC Context, since the BP engine is not necessarily a trusted software component. It enforces BP security controls by invoking TSC subsystem services (as shown in Figure 6: BP subsystem component overview) and retrieving security critical decisions from those trusted subsystems. Technically, the TSC Task is a design pattern in the chosen private process modelling language, e.g. WSBPEL.

TSC State

The TSC Task needs to retrieve its configuration data from the TSC Context when executed during process enactment. This security critical data is stored in the TSC State data structure only for the time of this TSC Task's execution.

TSC Service

The TSC Service's purpose is to abstract from the chosen TSC Context storage. It allows creating and updating the TSC Context. The TSC Context is created from TSC Extension Roles, which are deployed along with a CD via the CDL++2BPEL service or updated via the BPM Service when TSC Extension Roles are deployed at runtime.

TSC Extension Role

A TSC Extension Role is a design time document stating the requirement of a particular security control within a role in a CD. It therefore belongs to this role's BP and is inserted at a specific position in the sequence of business interactions within the CD. At runtime, there will be a TSC Task in the private/public process model, automatically modelled in by the CDL++2BPEL service. Four (SLA, Security, Trust and Notification) TSC Extension Role classes are so far identified and described in more detail in the Appendix. These correspond to four different types of security controls for BPM relevant for TrustCoM.

II.2 Deployment

II.2.a BP Designer

The BP Designer is used by the "business expert" who designs the choreography for the VO. It is therefore an optional component that could be installed by any VO member organization that wishes to participate in the choreography design process, must notably the VO initiator or manager.

II.2.b BP Repository

On the one hand there is an economic benefit from sharing business processes, such as standardization and common establishment. Therefore a common, central BP repository is supported by TrustCoM. On the other hand there might be business reasons, such as competitive advantage, for keeping even collaborative business processes as secret as possible. Therefore TrustCoM supports a local BP repository at the initiator's site as well.

II.2.c CDL++2BPEL Service

The CDL++2BPEL service needs to be installed by any VO member organization that wishes to participate in the top-down business process model of TrustCoM, i.e. every VO member organization.

II.2.d CD Knowledge Base

The CD Knowledge Base provides the private process parts to the CDL++2BPEL service. It therefore needs to be installed by every VO member organization.

II.2.e BPM Service

The BPM service manages the local business process at each VO member's site. Each VO member participating in the VO business process needs to install one.

II.2.f BP Engine

The BP engine executes the actual derived business process. Each VO member organization needs one.

II.2.g TSC Service

The TSC service allows accessing the TSC context which is necessary for implementing TSC extensions. That implies that each VO member organization that wishes to use TSC extensions needs to install a TSC service.

II.3 Interaction Scenarios

II.3.a Identification and Formation

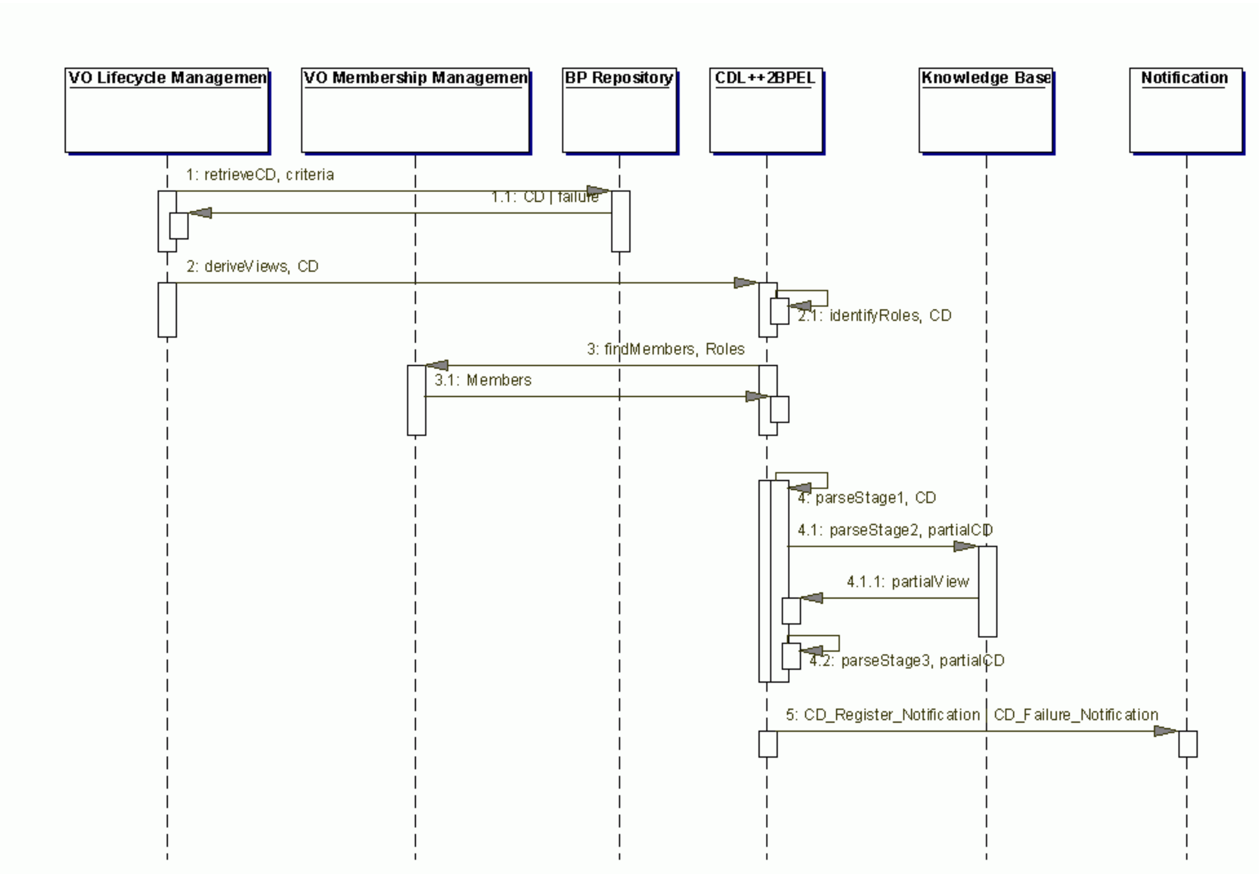

Figure 8: CDL++2BPEL Sequence

At first, VO Management is typically informed about the customer's business objective and needs to retrieve a suitable collaboration template from the BP repository describing in a global view/choreography how to achieve this goal (steps 1.*). A human business expert may be required to perform tailoring or adapt the template resulting in the final collaboration definition. The BP related services to the right deal with the CD further on and are deployed in the VO Member's domain.

VO Lifecycle Management verifies the received CD and invokes the CDL++2BPEL service for further CD processing (steps 2.*).

The member roles codified in the CD need to be met by a list of VO members. VO Membership Management is queried for this list (steps 3.*).

In the following (steps 4.*) the CDL++2BPEL service performs its main work, parsing the CD in three stages as outlined in the CDL++2BPEL component description above. Invocation 4.1 queries the BP Knowledge Repository service in the second stage with well known CD artefacts as parameters. When the CDL++2BPEL service encounters a TSC Extension Role document annotating the CD, it inserts a TSC Task into the BP model and deploys the contained data to the TSC Context (not shown here due to space reasons).

In the end (step 5), the CD is either successfully processed and counts as instantiated or failed because a VO member was not qualified to play a certain role or the automatic view derivation was not able to complete successfully. In the first case, a notification with the CD_ID is sent to register in the GVOA/partner profile, in the latter case, a failure notification is generated.

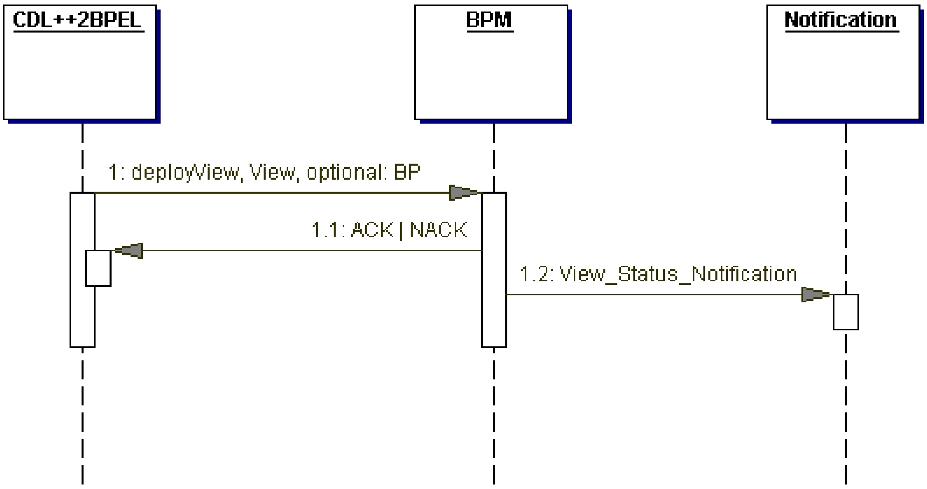
CDL++2BPEL
BPM
Notification
1: deployView, View, optional: BP
1.1: ACK | NACK
1.2: View_Status_Notification

Figure 9: View deployment

If the CD was processed successfully, Figure 9 shows the steps to finally deploy the generated views and optionally the private processes. The CDL++2BPEL service is able to do this by sequential or parallel invocation of the BPM services of each VO member's domain. The CD is fully deployed if positive acknowledgements are received for each view. Each BPM service also generates a notification containing the ID of a successfully deployed View for registering in the GVOA/partner profile. Private processes are not covered, since those – as the name suggests – are seen as private and are not exposed outside the VO Members' domains.

II.3.b Operation and Evolution

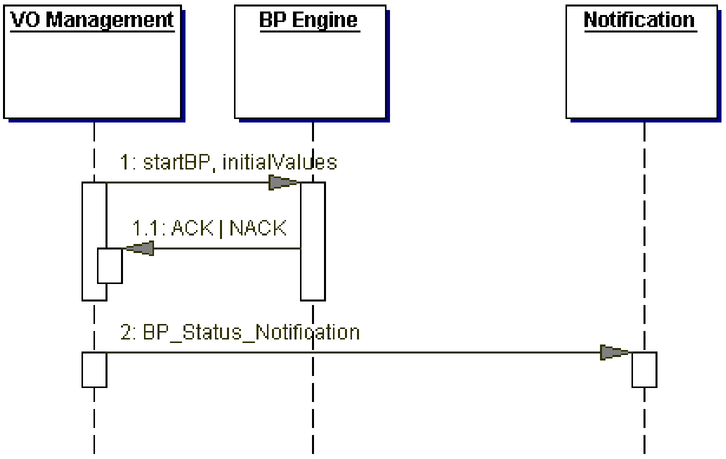

Figure 10: BP Execution

From the BP subsystem's perspective, the operation phase starts when the initial "overall" collaboration definition is set into motion. This still expresses the global view. From a VO Member's subjective perspective this means, the execution of the private processes implementing the beginning of the collaboration definition is started.

Typically, VO Management triggers this execution by invoking the process' view activity (here called startBP()) with initial values implementing the CD's start. The invoker receives acknowledgement, and a notification is sent out by VO Management.

It has to be noted that all steps in the initial VO phases may also be performed completely in the operation phase, e.g. recursive control leading into another VO structure, implementing a contingency plan or dynamically realizing further (sub-)collaboration definitions.

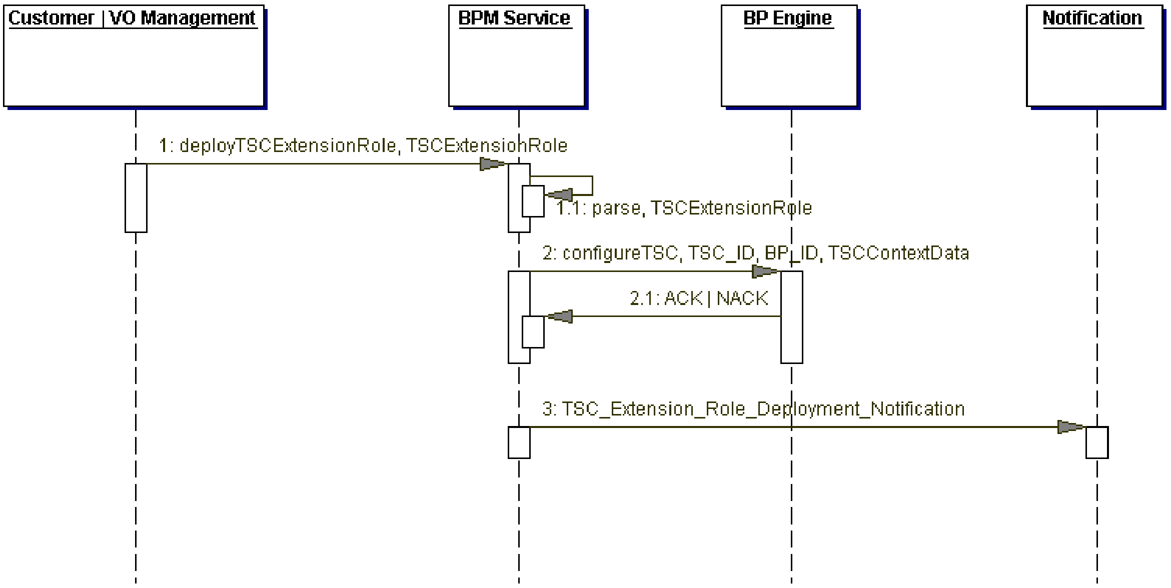
Customer | VO Management
BPM Service
BP Engine
Notification
1: deployTSCExtensionRole, TSCExtensionRole
1.1: parse, TSCExtensionRole
2: configureTSC, TSC_ID, BP_ID, TSCContextData
2.1: ACK | NACK
3: TSC_Extension_Role_Deployment_Notification

Figure 11: TSC Extension Role Deployment

During runtime, different authorised actors may want to update or initially configure TSC Tasks in views and private processes. The BPM service offers this method in its interface (steps 1.*). The TSC Extension Role document is also parsed by it to ensure validity and because the header information contains references to the target TSC Task, its ID and the view/private process ID which are required for further processing.

An apparent prerequisite is that the target process is already deployed, which is tracked in the partner profile. Since the instance of the target process (BPEL process) may already be running in the BP engine, steps 2.* perform the update on the engine's data itself.

The development work in AL2-WP29 explores the possibilities leading to an implementation of those models. The actual data deployment (after step 2.1 in Figure 11) may be realized by updating a data structure, the TSC Context (see WP21), stored in a trusted BP engine subsystem, but external to it. Each TSC Task maps to a dynamic list of attributes into the TSC Context which holds all TSC related data for one process instance.

Another possibility is extending the BP engine and handling the TSC Task data, essentially the TSC Context, engine internally. This approach requires the introduction of a "trusted" BP engine addressing threats of manipulation, malicious process models etc.

II.4 Dependencies Overview

II.4.a Supporting Services

BP design time functionalities such as the BP template repository are central to the VO and its members. The repository contains CD templates which are retrieved most importantly by VO Lifecycle Management. Since the BP subsystem offers generic services, other entities, e.g. a service provider, may retrieve CD templates in other VO phases as well. Access from partner domains has to be facilitated; therefore those services are offered as supporting services.

II.4.b VO Management

The BP subsystem offers services to VO management, particularly for the choreography services. The CDL++2BPEL service takes a collaboration definition, retrieved from the CD template repository, and matches public and private business processes for each involved VO member.

To also match VO members with roles required by the collaboration definition, VO membership management is required. Therefore, upon submission of a CD to the CDL++2BPEL service, the required role declarations in the CD's header are parsed. The service then calls VO Membership Management with the enumerated roles as parameters and expects VO members in return, meeting all role requirements.

When queried for Members matching CD roles, VO Membership Management also delivers a set of services described with business keywords. The latter match the internal member activities described in the CD. If those are not available or sufficient for dynamic service discovery and invocation from within a business process, the discovery service offered by VO Management can optionally be used (not depicted above). It can be invoked by the CDL++2BPEL service and also BP engine, containing a BP instance modelling a suitable invocation. When processes, public and optional private ones, are derived from a CD for each role by the CDL++2BPEL service and deployed in each member's domain, private processes already contain the correct sequences of service invocations. If the private processes were already present, required service EPR are assumed to be present as well; if these processes are also derived by the BP subsystem, the discovery service is used for the EPR identification.

II.4.c Notification

The notification subsystem allows sending notifications for BP subsystem related topics. VO Management and other interested subsystems need to subscribe to those topics.

It also allows receiving notifications from topics related to e.g. TSC subsystem, which are of interest for BPM.

III SLA Management Services SICS

The SLA Management subsystem encapsulates the functionality needed to help establish and enforce Service Level Agreements in a virtual organization. SLAs complement the description of services with non-functional aspects, mainly Quality of Service (QoS).

III.1 Components

Here is an overview over the components in the SLA Management Services subsystem. Note that the underlying WS-specifications are examined in more detail in the deliverable D18.

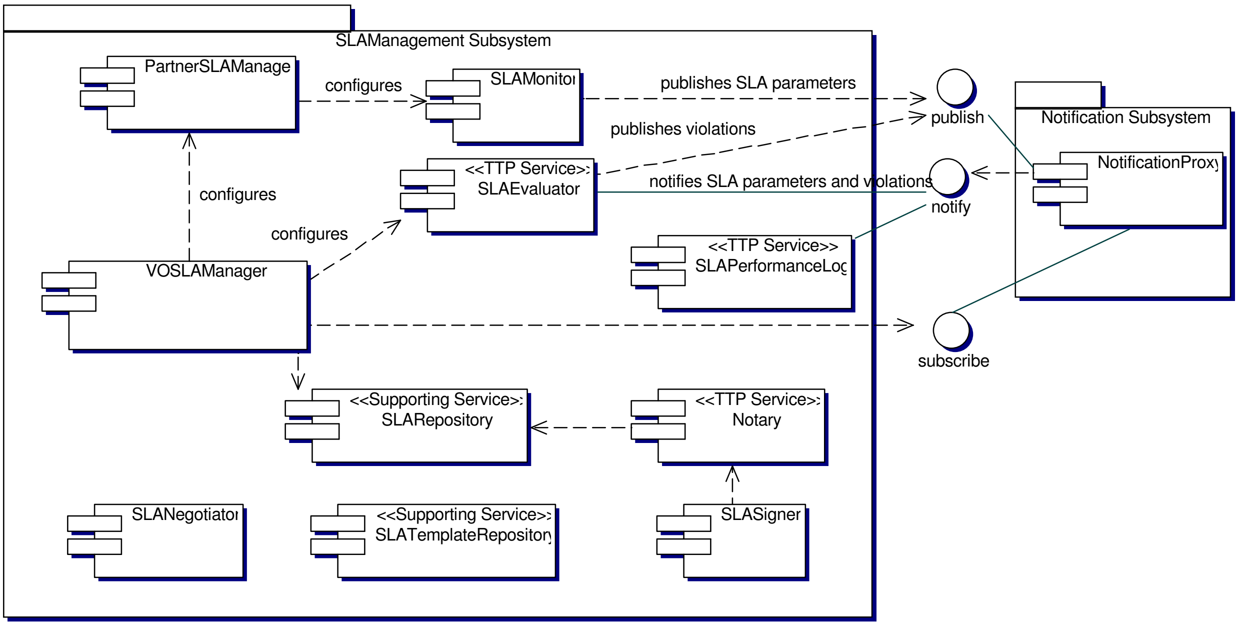

Figure 12: Components of the SLA Management Subsystem

III.1.a VO SLA Manager

This component functions as a coordinator for the different components in the SLA Management subsystem. In particular, it is responsible for the configuration of monitors and evaluators. It associates the Monitors and SLA Evaluators with an SLA and connects them with each other through the Notification subsystem. This component maintains a database of the Monitor-Evaluator connections and configuration. It is expected that the VO Manager controls this component, ordering it to configure the SLA Management subsystem during the formation of the VO, and to reconfigure it when the VO evolves.

III.1.b Partner SLA Manager

The VO SLA Manager delegates to the Partner SLA Manager the part of its functionality that is to be enacted at each participant organization (i.e. VO partner). Besides configuring local monitors and possibly some local evaluators, it may be responsible for converting abstract into concrete monitoring terms.

III.1.c SLA Template Repository

A service provider that wishes to announce the availability of a service may use this component to publish SLA templates. A template defines ranges for QoS parameters that the provider is willing to accept as starting point in any negotiation regarding service provision. Next to each SLA template, the SLA Repository stores the EPRs of the SLA Negotiator and SLA Signer services that would handle the negotiation, respective signing, protocols for the template at the service provider side.

III.1.d Notary

This (trusted third party) component witnesses the signing of SLAs between providers and consumers, and stores the signed SLA for later reference in an SLA Repository –the latter is a supporting service (see below).

III.1.e SLA Negotiator

This component provides support for negotiating agreements. The users of this component will include VO management processes in charge of signing agreements with service providers. A negotiator only offers functions and protocol implementations. The actual logic determining what is to be considered a successful negotiation lies outside the SLA Management subsystem. In other words, the "intelligence" required to negotiate an agreement would be in the hands of a human operator or some external module that will be consulted by this component.

III.1.f SLA Signer

This local component implements one of the sides in the signing protocol chosen for the VO and admitted by the Notary.

III.1.g SLA Repository

Each VO partner is assumed to have some sort of local SLA Repository service where it stores its own copies of the SLAs it is currently negotiated or has already signed. This local SLA Repository, that is not accessible from outside the organization, is used by the Partner SLA Manager when it is ordered to configure the local components in charge of monitoring the agreement.

There is as well another kind of SLA Repository, which is a *supporting service* (or trusted third party) and is not in the domain of any VO partner. It provides a secure store for signed SLAs where Notaries can upload SLAs into. TTP SLA Evaluators, which also lie outside the domain of VO partners, fetch the details of the SLAs they have to evaluate from this repository.

III.1.h SLA Monitor

Two types of SLA monitors have been identified so far:

a. Internal ASP and Host Domain Monitors

b. External or Trusted Third-Party Monitors

Internal monitors have direct access to the applications and resources they inspect. On the other hand, external monitors can only inspect web services at their interfaces and usually lie outside the control of the service owner. Both kinds, but particularly external monitors, are able to aggregate the metrics produced by other monitors into composite (higher-level) metrics.

III.1.i SLA Evaluator

An Evaluator receives monitoring information and keeps a state of each active SLA; it is also responsible for sending notifications on the event of SLA violation and/or fulfilment. The set of subscribed receivers for this kind of notifications may include VO Management and Policy services.

As with monitors, there could be both internal and external SLA Evaluators. Although their operation is identical, they serve different purposes. External SLA Evaluators are trusted by the signatories of the agreement to be impartial in their evaluations. Each participant may use internal SLA evaluators for self-management purposes, e.g. to detect low performance trends and establish corrective measures to prevent the violation of an SLA.

III.1.j SLA Performance Log

This component accumulates historical data on the performance of SLAs for future evaluation and use (e.g. for accountability purposes). This is done by subscribing to messages from the Notification subsystem and saving the messages in a database.

III.2 Interaction Scenarios

III.2.a Preparation

1) Register SLA template

A member of the Enterprise Network registers the services it is willing to offer (to potential VOs) by listing their descriptions in some Service Directory. These descriptions have associated SLA templates that are stored in the SLA Template Repository.

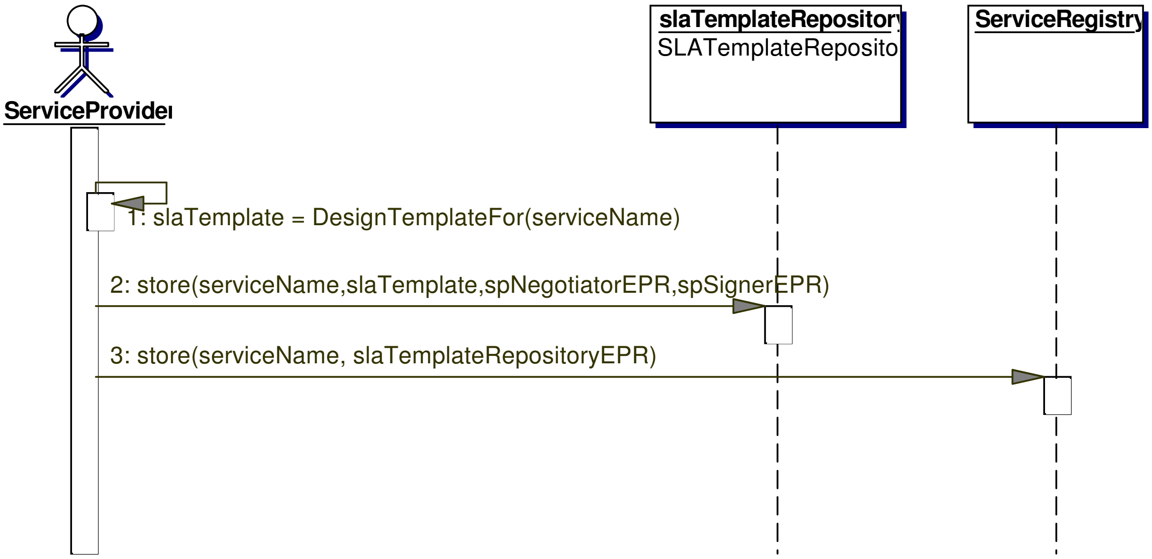
ServiceProvider
slaTemplateRepository
SLATemplateReposito
ServiceRegistry
1: slaTemplate = DesignTemplateFor(serviceName)
2: store(serviceName,slaTemplate,spNegotiatorEPR,spSignerEPR)
3: store(serviceName, slaTemplateRepositoryEPR)

Figure 13: SLA template registration

III.2.b Identification

1) Discovery service using QoS requirements

Regarding identification proper: The VO Manager will use the services of the (TTP) Discovery service to search for services to fulfil the requirements of the collaboration definition, including QoS requirements (cf. main document, sections I.3 and II.2.b).

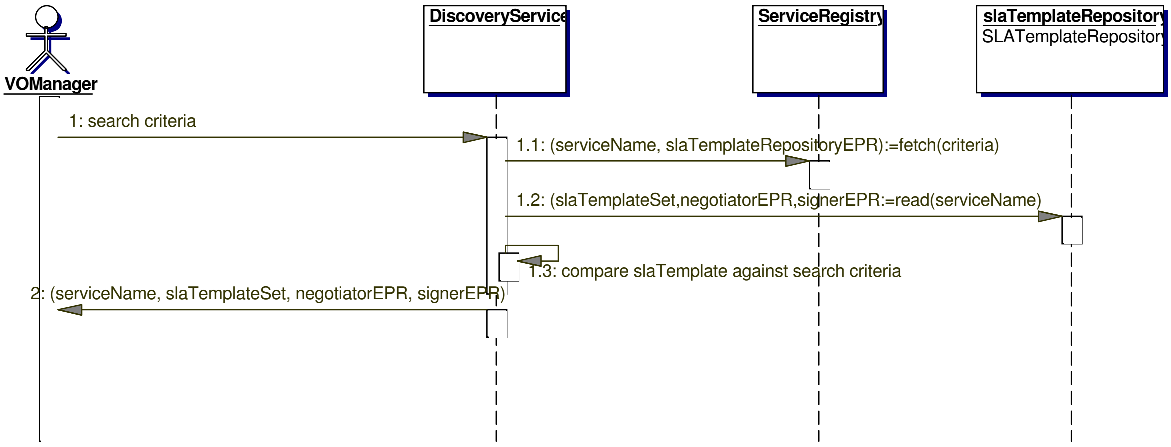
VOManager
DiscoveryService
ServiceRegistry
slaTemplateRepository
SLATemplateRepository
1: search criteria
1.1: (serviceName, slaTemplateRepositoryEPR):=fetch(criteria)
1.2: (slaTemplateSet,negotiatorEPR,signerEPR:=read(serviceName)
1.3: compare slaTemplate against search criteria
2: (serviceName, slaTemplateSet, negotiatorEPR, signerEPR)

Figure 14: Discovery using QoS requirements

2) SLA Negotiation

The TrustCoM Framework should provide flexible support for SLA negotiation, offering the possibility to apply several negotiation protocols. In its simplest version, the negotiation protocol is restricted to a single round where the offer, made by the service consumer based on the SLA template, is either accepted or rejected on the spot by the service provider. Figure 15 illustrates this protocol.

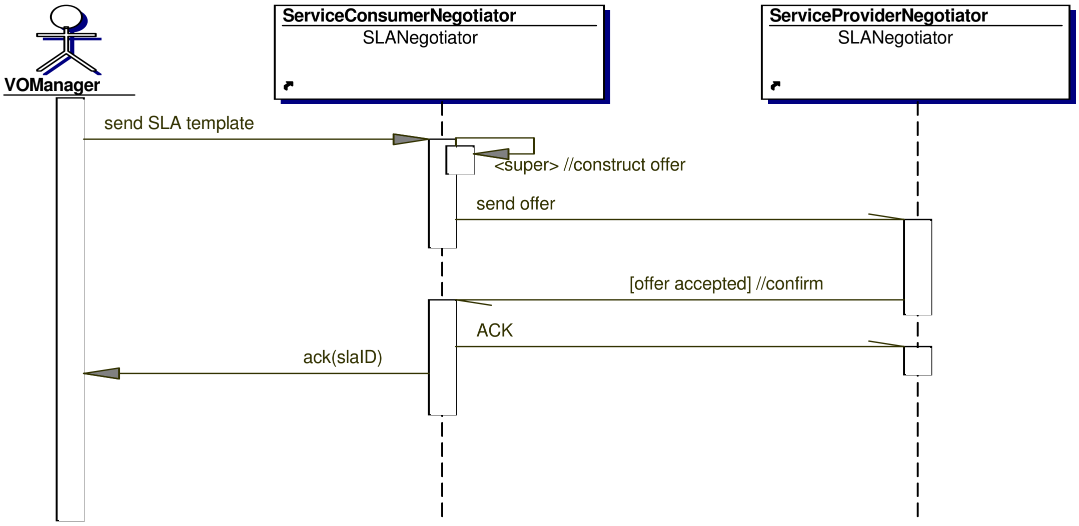
VOManager
ServiceConsumerNegotiator
SLANegotiator
ServiceProviderNegotiator
SLANegotiator
send SLA template
<super> //construct offer
send offer
[offer accepted] //confirm
ACK
ack(slaID)

Figure 15: Single-Round SLA negotiation protocol

Note that the implementation of the particular negotiation protocol is left in the hands of the SLANegotiator components of both the service provider and consumer. Here a “negotiation protocol” is understood as the sequence of message exchanges and their formats that form a negotiation. It does not cover the decision process involved in comparing and selecting the best offers. As already mentioned, such a decision process may easily require human intervention as well as the capacity to predict, at the provider’s side, the availability of the resources needed to comply with the offered QoS.

When searching for a number of services to fulfil the requirements of a particular business process, the VO Manager may want to make sure that the best agreements can be reached for *all* needed services before actually committing to signing any of them. In other words, a VO Manager may have to prevent the signing of an agreement if it is unable to conclude some other agreement, without which it becomes impossible to enact the business process.

III.2.c Formation

1) Signing and storage

When digitally signing a protocol it is reasonable to assume that no partner wants to be the first to sign, in fear that other would-be signatories could change their minds before signing. Even though a protocol is not valid until everybody has signed, there are situations, for example when resources need to be reserved, that can have a negative impact for the party that signs first, if the second party does not sign. The simplest solution involves a trusted third party (Notary) that first collects all signatures, verifies them and then distributes the signed contracts among the signatories. Figure 16 illustrates this protocol. In this case we have chosen to implement a protocol which needs an intermediary; however there exists alternative protocols which attempt reducing the performance costs incurred by the Notary. The TrustCoM Framework will only provide support for Notary-based protocols.

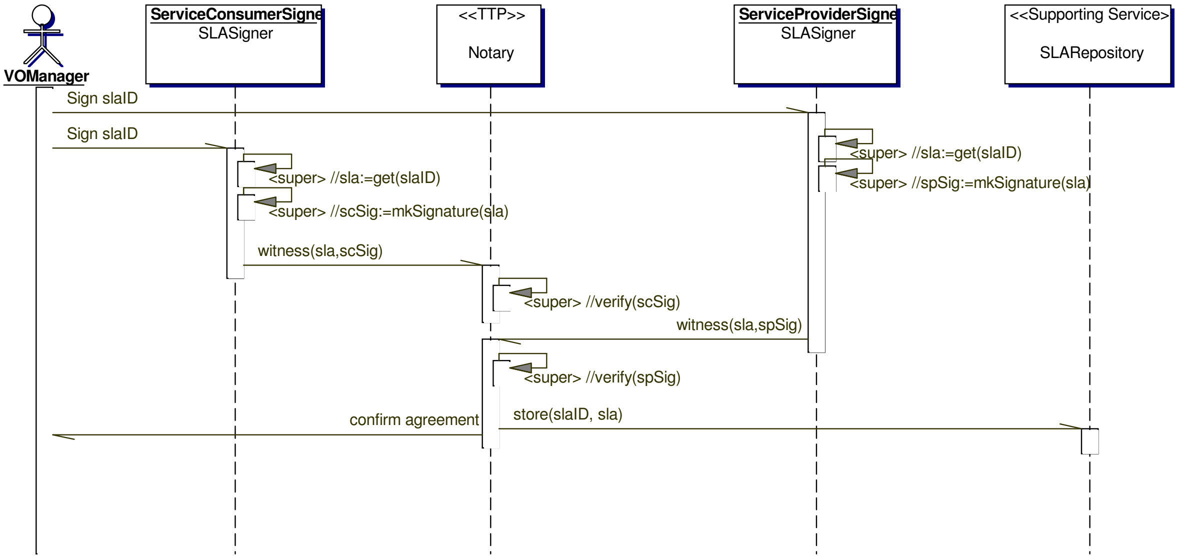

Figure 16: Signing and storage of SLAs

The Notary communicates the result of the negotiation to the VOManager. Without this step the VOManager would be unable to determine the successful formation of the VO.

2) Service provision configuration

Each Application Service Provider (ASP) manages service provision according to the signed SLA. During service instantiation, the Service Instantiator passes the identifier of the SLA document to the Service Factory. In this way, the right resources can be allocated to the execution of the service instance so that it can comply with the requirements posed by the agreement.

3) Configuration of Evaluators and Monitors

The VO Manager uses the VO SLA Manager to configure Evaluators and SLA Monitors in order to monitor SLA performance. This task is completed with the help of the Partner SLA Manager when there is a need to configure monitors and evaluators that are internal to an organization. The Partner VO Manager also maps “public” SLA parameters to internal configuration information.

The VO SLA Manager may configure the Notification subsystem to direct notifications from monitors to evaluators and other monitors. The configuration data sent to monitors and evaluators consists of relevant parts of the SLA and information on which Notification Topics will be used.

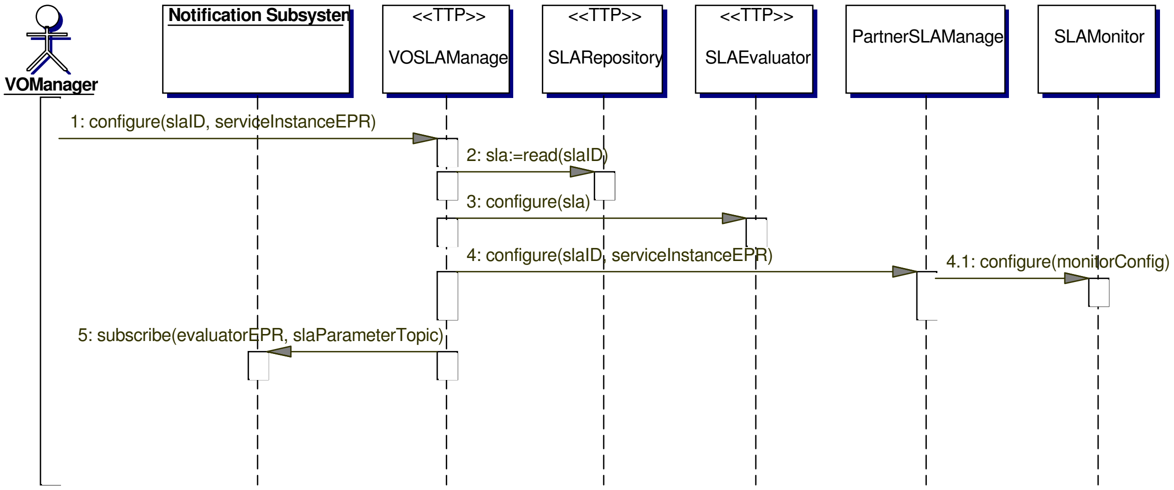
VOManager
Notification Subsysten
<<TTP>>
VOSLAManage
<<TTP>>
SLARepository
<<TTP>>
SLAEvaluator
PartnerSLAManage
SLAMonitor
1: configure(slaID, serviceInstanceEPR)
2: sla:=read(slaID)
3: configure(sla)
4: configure(slaID, serviceInstanceEPR)
4.1: configure(monitorConfig)
5: subscribe(evaluatorEPR, slaParameterTopic)

Figure 17: Configuration of SLA Monitors and Evaluators

III.2.d Operation & Evolution

1) SLA parameter computation

Simple and aggregating[2] monitors, using the configuration provided during the formation phase, observe the execution of a service, host process or even business process, and compute SLA parameters according to the metrics defined in the corresponding SLA.

2) SLA parameter communication

SLA parameters are communicated upon request by their consumers (Monitors and Evaluators) or upon the occurrence of events (including time events) (see Figure 18).

3) An Evaluator reports contract violation/fulfilment

Notifications of SLA performance are generated by the Evaluator and channelled to the Messaging/Notification component (and distributed to the receivers that were subscribed in the Formation phase).

Note that the SLA will not trigger VO adaptation actions *itself* (but via policies at the Policy Service) - accordingly the “consequences” are not defined in the SLA.

Monitors and Evaluators deployed at the level of the ASP domain or the host system should notify their respective domain managers.

[2] An *aggregating monitor* computes metrics using data probably computed by other monitors.

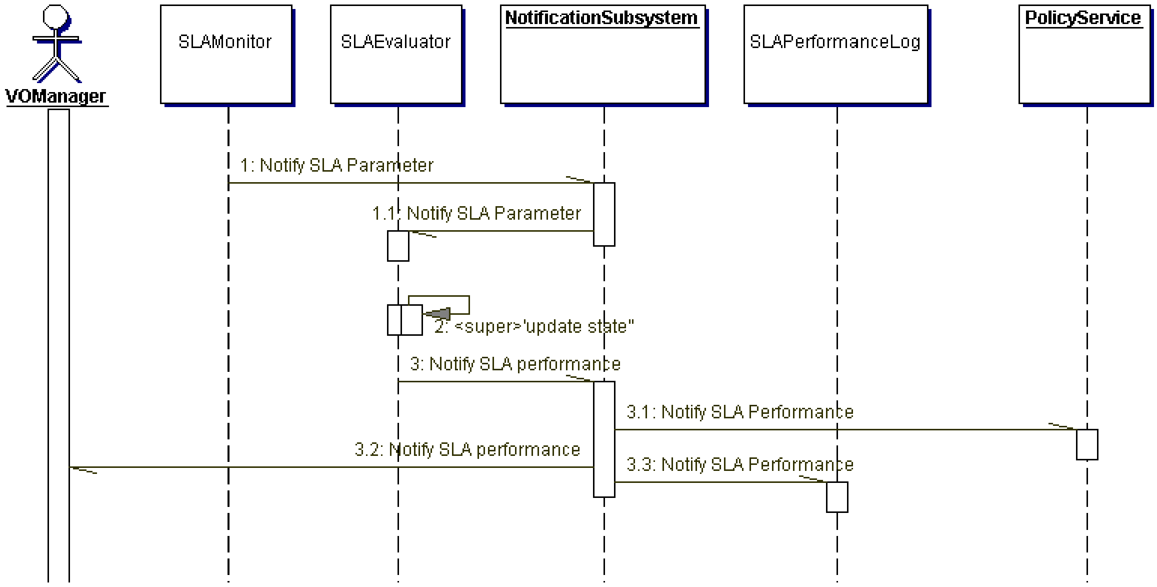
VOManager
SLAMonitor
SLAEvaluator
NotificationSubsystem
SLAPerformanceLog
PolicyService
1: Notify SLA Parameter
1.1: Notify SLA Parameter
2: <super>'update state"
3: Notify SLA performance
3.1: Notify SLA Performance
3.2: Notify SLA performance
3.3: Notify SLA Performance

Figure 18: SLA Monitoring (push model)

III.2.e Dissolution and Termination

1) Contract termination

SLAs and contracts that have reached the end of their validity period or have been cancelled due to broken contracts may be removed from the SLA Repository. Notice however that a contract or SLA may be kept in storage after its termination for accountability purposes. Evaluators and Monitors are likewise re-configured to release the resources dedicated to monitoring the terminated contracts. The whole process is instructed and coordinated by the VO SLA Manager.

Contracts can also be terminated actively by VO Management as part of VO adaptation procedures. However, in this case, special care must be taken to keep record of the contract violations incurred by the abrupt termination of the contract.

III.3 Dependencies Overview

The following table summarizes the dependencies between the SLA Management services and other subsystems in the architecture:

	Dependency
VO Mgmt	The General VO Agreement (GVOA) contains references to (signed) SLAs regulating the QoS properties of the services provided to the VO by each VO partner.
	Partner Profiles describe the services that an EN member is willing to offer to a VO. Therefore a Partner Profile refers (probably indirectly) to the SLA templates associated to each service offered by an EN member.
Policy	SLA Performance notifications (i.e. notifications of agreement violations) will usually be directed (by the Notification subsystem) to one or more ECA interfaces responsible for triggering the corresponding adaptation mechanisms.
	The Policy Service will distribute and install the access control policies that ensure that the VO SLA Manager is not interfered upon when it comes to managing SLA Evaluators and to configuring the

	distribution of SLA-related notifications.
EN/VO Infrastr.	The Coordination Infrastructure is used by the SLAM subsystem to implement distributed protocols (e.g. signing and notarization).
	The Discovery Service accesses the SLA Template repository to recover the SLA templates that correspond to a (probably not yet instantiated) application service.
	The Service Instantiator passes an SLA ID to the Service Factory to create a service instance that is ready to comply with the agreement. The Service Factory must have access to the local SLA Repository in order to recover the SLA document from its ID.
	The communication of SLA Parameters and SLA Performances within the SLAM subsystem and with other subsystems makes use of the Notification Subsystem. The VO SLA Manager performs the appropriate subscriptions to Notification Brokers and Proxies (see Formation phase).

Table 1: overview over the dependencies between SLA Management components and components of other subsystems

IV Trust & Security Services EMIC

The Trust and Security Services provide various technical means for the management of **trust**, and related concepts like **risk** and **assurance**, between partners in a VO, as well as provide supporting services for managing trust in nontechnical ways.

Security Token Services are used to authenticate previously unknown VO members across administrative domains. This for example assures that the VO member is trusted by the partner organization to act on behalf of it in the context of this VO. This in turn can be used to approximate the risk in interacting with this VO member by the (lack of) trust in the partner organization. Security token services are accompanied by configuration management and trust negotiation services. The configuration management services are used to adapt the security configuration to changes in the VO, for example making a trust relationship to a new VO partner organisation known to the security token services. Trust negotiation services can be used to specify disclosure policies for sensitive data in security token when authenticating to other VO partners or EN members. This is a form of **risk mitigation** in that the risk of misuse by the partner organisation of the sensitive data contained in the security token is reduced.

Besides the identity of VO members, which is authenticated resp. managed by token services, another technically tangible trust criteria is the VO member's prior behaviour. In order to support the judgement of behaviour that is not only observed by oneself, this subsystem provides a Reputation Management service that collects and combines individual ratings about a VO member's behaviour into a numerical value, it's **reputation**. Both this absolute value, as well as sudden big changes in it, can thus give quantifiable input into decision making processes like which partner to choose when forming a VO, or whether to expel a partner from a VO. Also, it provides a certain incentive for good behaviour, as misbehaviour will likely have negative consequences in the future, for example fewer contracts and therefore reduced business volume due to the reduced reputation. This equates to (a limited amount of) risk mitigation for the other VO partners.

Non-repudiation, as provided by the Secure Audit service, provides further mitigation of the risk involved in interactions between VO partners by assuring that misbehaviour will have (potentially drastic) legal implications. The Secure Audit thus also provides the evidence necessary for enforcing penalties specified in the legal contract that governs the VO.

IV.1 Components

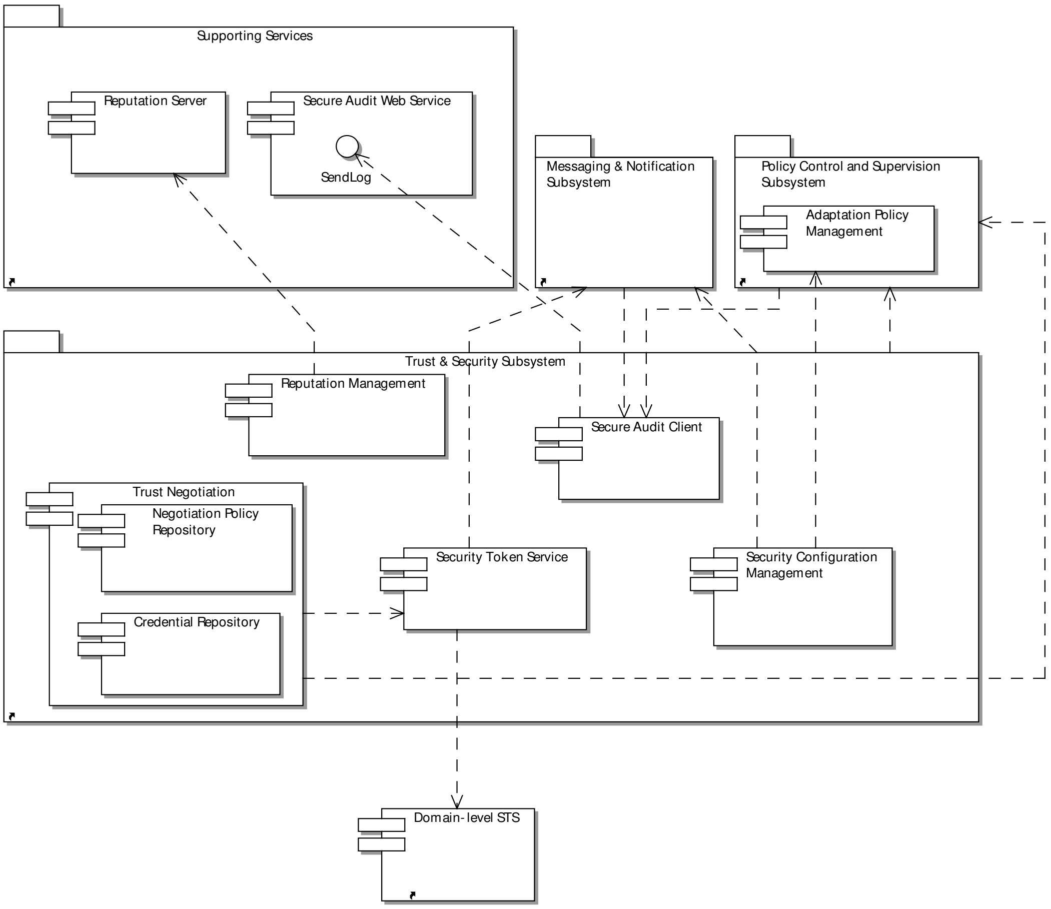
Supporting Services
Reputation Server
Secure Audit Web Service
SendLog
Messaging & Notification Subsystem
Policy Control and Supervision Subsystem
Adaptation Policy Management
Trust & Security Subsystem
Reputation Management
Secure Audit Client
Trust Negotiation
Negotiation Policy Repository
Credential Repository
Security Token Service
Security Configuration Management
Domain-level STS

Figure 19: Component overview.

IV.1.a Security Token Services

The security token services component provides services that enable the issuing and validation of security tokens across different security domains. The architecture allows for a layered structure, where security token services on lower levels can delegate the issuing and validation to token services on higher levels. For example, a domain-level STS can delegate to an organization-level STS, or a organization-level STS can delegate to a VO-wide STS.

IV.1.b Security Configuration Management

The security configuration management component provides services to adjust the host's/service's/domain's security configuration to changing conditions in the environment, for example to changes in the VO membership, or to detected attack trials. There are three possible ways of how the security configuration management component can be enacted at runtime:

- Configuration changes are manually triggered by an administrator using a custom management interface software.

- Certain simple rules can be defined statically and deployed together with the component
- In a more sophisticated scenario, a special component of the policy services subsystem would be responsible for storing and enforcing so-called adaptation policies. The enforcement of adaptation policies would - among other things - encompass the use of the security configuration management component.

IV.1.c Trust Negotiation

The Trust Negotiation component provides services that enable a participant in a VO to disclose just the security tokens to another party that are needed to access a service.

The component uses custom negotiation and disclosure policies, which describe which tokens have to be disclosed for accessing which service and which describe which credentials are sensitive and should not be disclosed to other parties (when not necessary).

Trust negotiation provides added value of protecting the privacy of EN/VO members in the sense that credentials may disclose more sensitive information than a organization is willing to provide to just anybody. The risk that is carried with unwanted disclosure of information inside security credentials can be imagined to be substantial in certain kinds of enterprise networks, especially in very open and loosely connected ones, In the cases of the application scenarios that we concentrate on in the TrustCoM project, however, this risk is not considered to be business-critical because there the participants are relatively well-known to each other. Because work will concentrate on components that are critical for the application scenario, the trust negotiation component, while part of the TrustCoM architecture, might not be part of the reference implementation to be provided during the time-frame of this project.

IV.1.d Secure Audit Service

The Secure Audit Client provides a common interface to both the generic messaging and notification services, as well as to the policy control subsystem, to securely safe a log of actions that happened during their execution (for example, positive or negative authorisation decisions, etc.). The secure audit client forwards these logging requests to a supporting (or trusted third-party) service, the secure audit web service.

IV.1.e Reputation Management

The Reputation Server is a supporting (or trusted third-party) service that provides reputation information about VO or EN members and is accessed by the reputation management component of the trust & security services subsystem

The reputation service is used to record trust relationship values between individual members, i.e., individual reputation ratings, and to calculate and disseminate the combined reputations of EN/VO members. The metrics used for measuring reputations are definable within the initial set up of the reputation service. The metric used will define the nature of the measurement of trust (i.e. what values will be used to record it etc) and what information will be held regarding the situation which merited the recording of trust (i.e.

what were the circumstances under which party A formed a trust relationship with party B, and how is this recorded in the metric).

IV.2Interaction Scenarios

We will concentrate on 3 core usage scenarios here: addition of an organisation, normal operational work, and removal of an organisation. The other usage scenarios presented in Section 2.5, like establishment of the VO, replacement of an organisation, and dissolution of the VO can be seen as combinations or iterations of these core scenarios, at least from the point of view of the Trust and Security Subsystem.

IV.2.a Addition of an Organisation

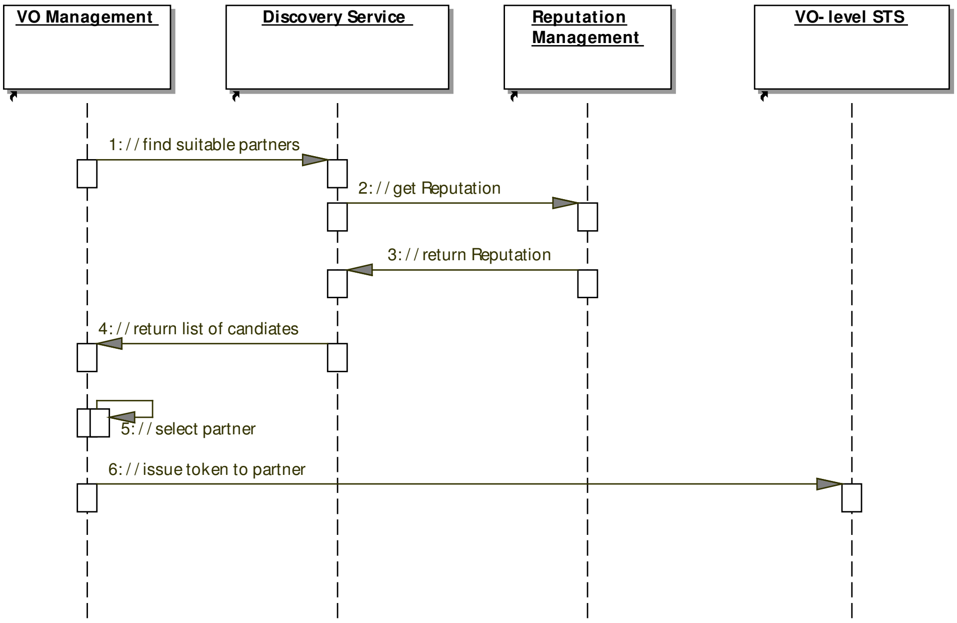

Figure 20: Scenario for adding an organisation to the VO.

When an organisation has to be added to the VO, be it in the identification/formation phase, or dynamically in the evolution phase, the following steps happen:

Triggered by the Membership Management component of the VO Management subsystem, the Discovery Service queries well-known repositories to find a list of potential organisations matching the request from the VO Management. Among other things, it asks the Reputation Management Service for the trust values of potential partners to filter for those organisations that fulfil the reputation requirements. When a suitable partner is found, and other negotiations with this partner (e.g. SLA negotiation, see above) are successful, VO Management triggers the STS to issue appropriate tokens for this new partner.

In case this scenario happens in the identification phase of the VO, instead of the evolution phase, the steps occur in the same sequence, however, the issuing of the tokens only gets started in the formation phase of the VO, after all other necessary partners have been discovered. The figure above shows the scenario happening in the evolution phase, where only a single organisation has to be added.

Furthermore, the figure assumes the existence of a VO-level STS, which is a valid option for the deployment of trustcom services, but not required. If there is no VO-wide STS employed, in step 7 the membership management component has to iterate over all VO partners to make the new organisation known to them.

IV.2.b Normal Operational Work

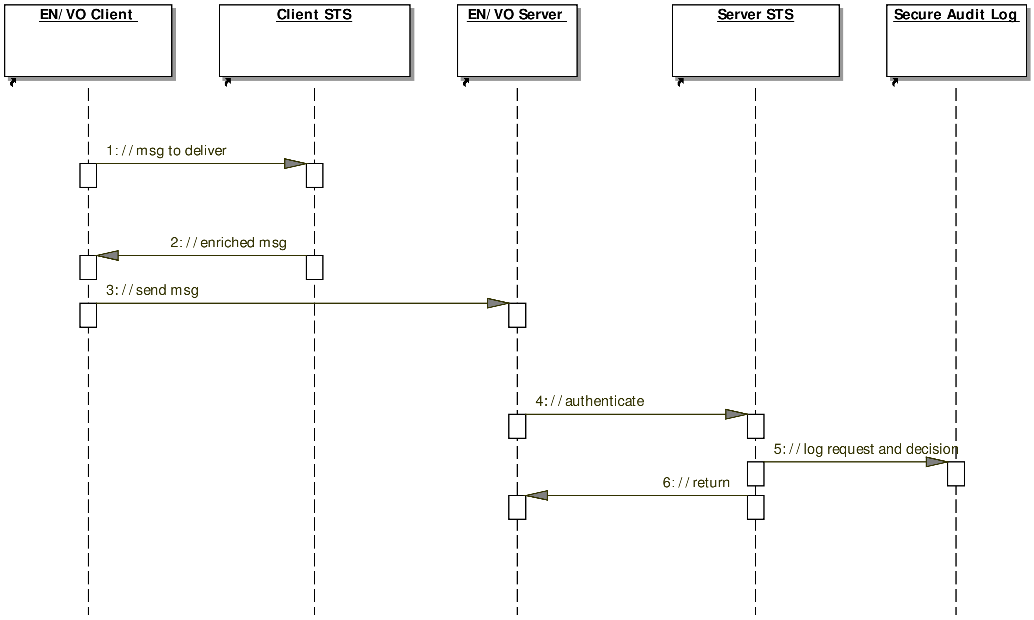

Figure 21: Basic Authentication Scenario.

During normal operation of the VO, when a client sends a message to an application web service, the Policy Enforcement Point (PEP) of the service forwards incoming messages to the token services, which authenticate the message by validating the contained tokens. This may involve several levels of token services, which each validate respective delegation claims; for example host-level, organisation-level and VO-level token services. Symmetrically, before sending this message, the PEP of the client has forwarded the message to its own token services in order to fill in necessary tokens and sign/encrypt the message according to the applicable policy.

After this basic authentication, additional authorisation checks are performed by the Policy Decision Point (PDP) for example access control checks on the content of the message (see below). The architecture allows for several options of how the communication between PEP, PDP and STS is organized: The PEP may ask the PDP and the STS seperately, the PEP may ask the PDP, which forwards the request to the STS, or the PEP may ask the STS, which forwards the reqeuest to the PDP. The choice of which option to take depends on the concrete application and the type of authorisation policies to be checked, as the requesting component (STS or PEP) has to send all information relevant to the authorisation decision from the message header to the PDP. See the section below on EN/VO for a further discussion of this issue.

The policy subsystem optionally uses the secure audit web service to log it's decision. Also, if additional tokens are necessary to authorize the request, the server may start a trust negotiation process with the client, in which they try to exchange the missing tokens.

Finally, the token services log the requests and validation decision using the secure audit web service.

IV.2.c Removal of an Organisation

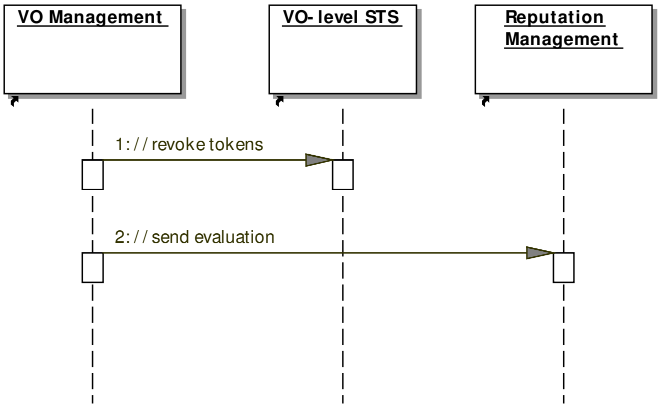

Figure 22: Scenario for removal of an Organisation.

The interactions for the removal of an organisation are in fact almost the reversal of the interactions for the addition of the organisation: First, the authentication tokens are invalidated, then the experience with this organisation is pushed to the reputation management service.

IV.3Dependencies Overview

The following table summarizes the dependencies between the Trust and Security services and other subsystems in the architecture:

	Dependency
EN/VO infrastr.	Discovery service ask reputation management during Identification phase and when looking for a suitable organisation in the evolution phase of the VO.
	Reputation Management informs about changes in a partner's reputation via the notification service
VO Mgmt	Membership management triggers the STS to issue suitable tokens during the formation phase.
	Membership management triggers the STS to invalidate the respective tokens during dissolution or when an organisation leaves the VO.
	Membership Management sends evaluation of performance to the reputation management service during dissolution or after the organisation leaves the VO
Policy	PDP may log their decisions using the Secure Audit Web Service
	PEP or PDP request the STS to issue token for outgoing messages
	PEP or PDP request the STS to validate tokens for incoming messages

Table 2: overview over the dependencies between Trust & Security components and components of other subsystems

V Policy Control IC

Policies define changes in the behaviour of systems and provide the primary means by which the configuration and operation of a VO can be customized and can be adapted at run-time according to changes in the requirements of the VO. The reason for encapsulating these behavioural aspects as policies (rather than as traditional programs) is that policies can be dynamically replaced without interrupting the system's functioning. In particular, the management strategy of the VO itself or the access rights given to partners may change dynamically during the life-time of the VO and may vary substantially between VOs. For example, in the CE scenario SLA violations are likely to require compensation actions (either payments or allocation of additional cycles on the analysis service) and will involve loss of reputation while in an aggregated services scenario SLA Violations may compromise the goal of the VO itself.

In TrustCoM we are focussing on two types of policies: obligation policies in the form of event-condition-action (ECA) rules and access control policies in the form of authorisation and delegation rules[3]. The rationale for this choice is twofold: on one hand these policies enable us to encode both the operational choices i.e. what should happen when specific events occur and the authorisation aspects of the framework, on the other hand the use of ECA rules in conjunction with the Notification Brokering system enables the development and deployment of a low-coupled architecture for VO frameworks which minimises the dependencies between the various components. To integrate these policies in the TrustCoM framework we need to provide the means to:

- Bootstrap the VO operation with a given set of policies. These policies may be either pre-defined (e.g., inside the Generalised VO Agreement), instantiated from policy templates, defined by human VO administrators or negotiated at run-time. The policy sub-system does not address how these policies come into being but how they are deployed and enforced in the VO.
- Change the policies inside the VO in a pre-defined manner. This is achieved by considering policies as managed objects themselves that can be instantiated, deployed and enforced through the actions of other policies. Thus a policy may trigger the instantiation and dynamic replacement of policies in a VO.
- Change the VO policies through human intervention as it is unlikely that all circumstances can be foreseen at the VO formation stage. It is therefore necessary for a human administrator to be able to dynamically interact with the system and change its behaviour by specifying new policies or removing the old ones in order to adapt to unforeseen events or new customer requirements.

In summary, policies are used to: enable the creation of different VOs with different operating rules according to the requirements of the Enterprise Network, adapt the VO's functioning to the operational requirements and implement access control.

[3] Note that the term: "*policy*" is overloaded and even within the TrustCoM framework it is occasionally used to refer to other declarative specifications. A discussion on the general meaning of the term as well as all its uses in TrustCoM has been presented in D16 Conceptual Models

V.1 Components

The main components are: the Policy Service which fulfils the role of policy deployment manager and policy enforcement component for obligation policies (ECA-rules), the Policy Interface Component through which administrators can interact with the running VO system and the Authorisation Policy Decision Point.

- **Policy Interface** - The policy interface are the user interface elements used by an administrator in order to interact with a policy service. This will allow listing the policies loaded in the service (including their status) and other elements present in the domain service component of this service (see next section), instantiating new policies as well as enabling and disabling them.
- **Policy Service** - The policy service has primarily four functions: deploying the authorisation policies to the authorisation policy decision points and deploying obligation policies to other policy services, managing the policies specified and their life-cycle, managing the adapter objects for the target services it performs actions on and implementing the obligation policies defined in the form of event-condition-action rules. Each policy service has a local repository, called the Local Persistence Store, in order to ensure the persistence of policies.
- **Policy Decision Piont** - The policy decision point (PDP) implements authorisation and delegation policies by evaluating requests sent by the policy enforcement point against the policies that have been loaded to it. Policy Enforcement Points act as interceptors for each request, interface with the trust and security services in order to validate the credentials received with the request and forwards all the information to the PDP who returns a permit/deny reply to the PEP

V.1.a Policy Service Architecture

The Policy Service itself is a composite component, which comprises four main components: the local domain service, the event propagation engine, the policy interpreter and the local repository. The structure and main interfaces of this component are represented in the figure below.

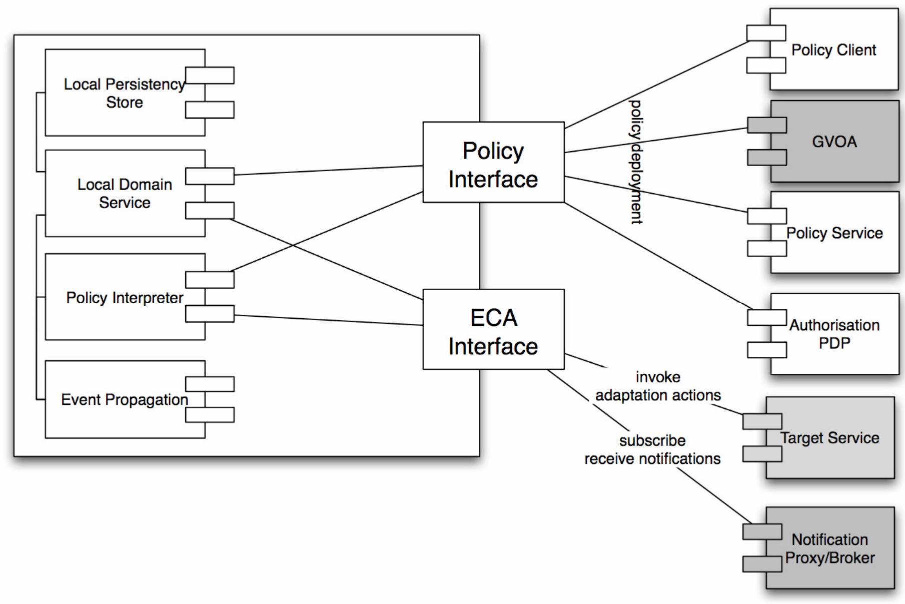

Figure 23: Overview of the Policy Service Architecture

Conceptually the policy service has two interfaces. One interface relates to the specification, instantiation, loading and unloading of policies. It is used by the policy client or the other policy services in order to interact with the policy service and by a policy service to deploy access control policies to the Authorisation PDP. The other interface is used to receive events from the notification broker and perform actions on target services.

Policies apply to different sets of subjects (for authorisation policies) and targets. These components may present heterogeneous interfaces though which invocations may be performed on them. For example, re-configuration actions on a web-service may be performed through a standard SOAP messaging interface or through a WSRF interface. Infosets required for their configuration may also differ from service to service. For this reason the policy service should be able to maintain adapter objects for the services which it invokes in order to present a unified interface to the policy interpreter. Furthermore, policies, need to be managed themselves in terms of their life cycle (enabled, disabled, etc). In order to help with the categorisation and deployment of the adapter objects and policies all policy services use a domain service component. The domain service provides a hierarchical structure in which objects can be grouped in order to apply a common policy to them and policy objects can be managed (including through other policies). Domains in conjunction with policies are considered as one of the most effective means of categorising and uniformly applying policies in distributed systems management.

Policies are persistent and must be maintained across interruptions and re-starts of the policy service. Each policy service will therefore use a local repository for managing and ensuring the persistency of policy object.

Finally, the policy service maintains several policies that apply to different objects. Several policies may be triggered by the same event and different policies may be triggered by different events. To this end the policy service must internally implement the functionality of an event (notification) local system that enables to match the events against the policy triggers.

V.1.b Policy Decision Point (PDP) Architecture

The policy decision point receives access control policies including both authorisation policies and delegation policies in a format suitable for implementation by the access control decision engine (XACML). The architecture of this component is comprised of a policy interaction module which keeps track of the policies loaded into the current engine so that they can be unloaded on request from the policy service, a request handler which receives requests from the PEP applies any data processing required and handles all communication with the PEP and the access control engine evaluation. These components and interactions are represented in the figure below.

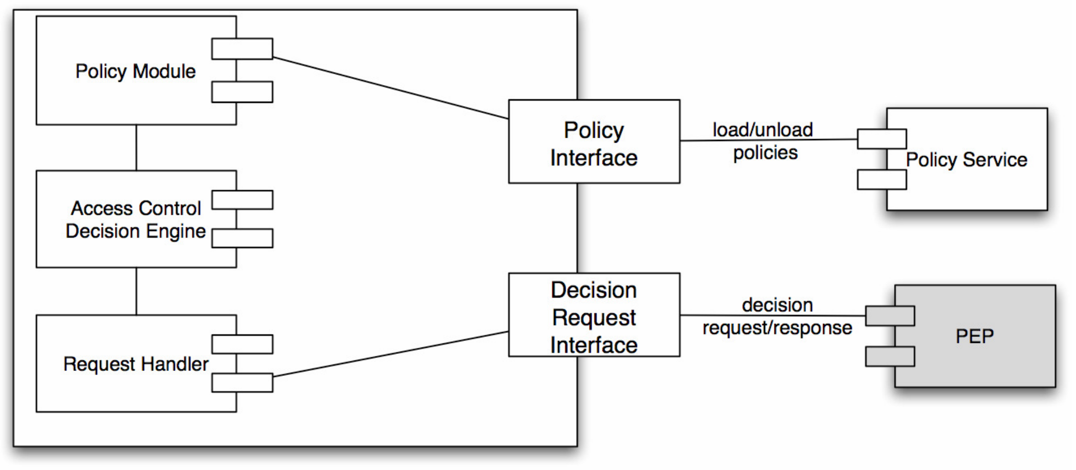
Policy Module
Access Control Decision Engine
Request Handler
Policy Interface
Decision Request Interface
load/unload policies
Policy Service
decision request/response
PEP

Figure 24 Policy Decision Point Architecture

V.2 Deployment

For each of the components of the policy sub-system several deployments are possible depending on the size, complexity and services offered within the VO. The policy decision point (PDP) makes access control decisions for all requests arriving to a service. Consequently, each service must be associated with exactly one PDP – because access must be controlled to each service and associating a service with several PDPs requires additional semantics for their ordering and precedence. However, the same PDP may control access to several services. Typically, the PDP remains within the control and the administrative domain of the VO participant i.e., requests must ultimately be permitted by the participant itself. Using a PDP outside the administrative domain of the participant amounts in effect to surrendering entire control of the service to an external party and although there are cases in which this arises (e.g., the entire physical resource is shared within the VO) such cases remain rather rare.

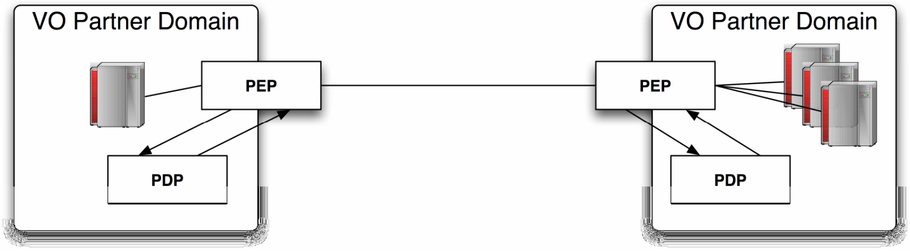

Figure 25 Typical PDP deployment during invocations

Figure 25 shows a typical deployment of PDPs during the invocation of a service. Requests to/from a web service are intercepted by the Policy Enforcement Point (PEP) which requests an access control decision from the PDP. The PDP on the server side (right hand side of the figure) is required in order to control access to the services provided by that partner. Access control on the client side (i.e., on the left hand side of the figure) is often optional and based on the partner's policies more than the VO policies. Cases where access control on the client side is needed include aspects relating to the dissemination of information where the target cannot be trusted to enforce the authorizations. Note that there can be multiple PDPs inside an administrative domain to cater for different services. However, only the PEP and the Policy Service need to be aware of such separation. The former because it needs to know which PDP is responsible for making access control decisions for the specific service and the latter because it needs to distribute authorization and delegation policies to the PDP. In all cases the PDP remains transparent to the invocation path.

Several deployments are also possible for the Policy Service component. A VO must have at least one policy service in order to a) distribute the authorization policies to the PDPs and b) to enforce adaptation policies in the form of ECA-rules during the operation of the VO. This is likely to be the simplest and most frequent form of deployment as the VO partners do not require additional infrastructure support, yet the strategy for VO operation can be dynamically changed according to changes in requirements Figure 26. In this deployment, each of the VO Partners exposes a limited management interface to their services, and access control can be used in order to restrict these invocations in the same way as for normal invocations. The policy-service is then part of the administrative infrastructure of the VO. Although Figure 26 shows a single policy service, multiple instances of the policy services can be used in practice provided that the GVOA identifies the policies that each service needs to implement.

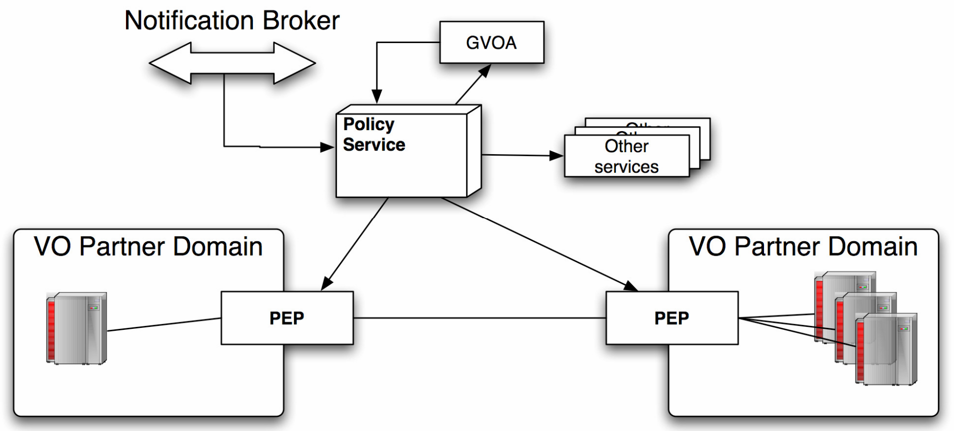

Figure 26 Typical Policy Service Deployment

A second type of deployment can be achieved where each of the VO partners have their own policy service instance. This is shown in Figure 27 below. In this case an administrative policy service for the VO may be optional depending on the functions required by the VO. In this deployment the GVOA (or an administrative policy service) distributes policies to each of the partners in the VO. This deployment has two advantages: it permits scalability to larger VOs and permits policies to be used for more complex tasks at the local level of the partner domain. However it is also much more difficult to realise.

Firstly, because although the policies are enforced in the administrative domain of the VO partner they are defined externally at the VO level. This implies that the partner must expose any information required for the definition of those policies. For example, if policies are triggered by events that remain internal to the VO partner these events must be known at the VO level in order to specify the policies. Secondly, because the VO partner would need to verify and validate the policies received. Whilst, it is possible to verify that a policy is the same as one agreed in the GVOA (i.e., verify their integrity) it is not possible at present to verify the contents of that policy with respect to the context in which it must be deployed. This is an area which requires significant additional research (none of the existing policy systems provide such functionality) and will not be addressed within the context of the TrustCoM project.

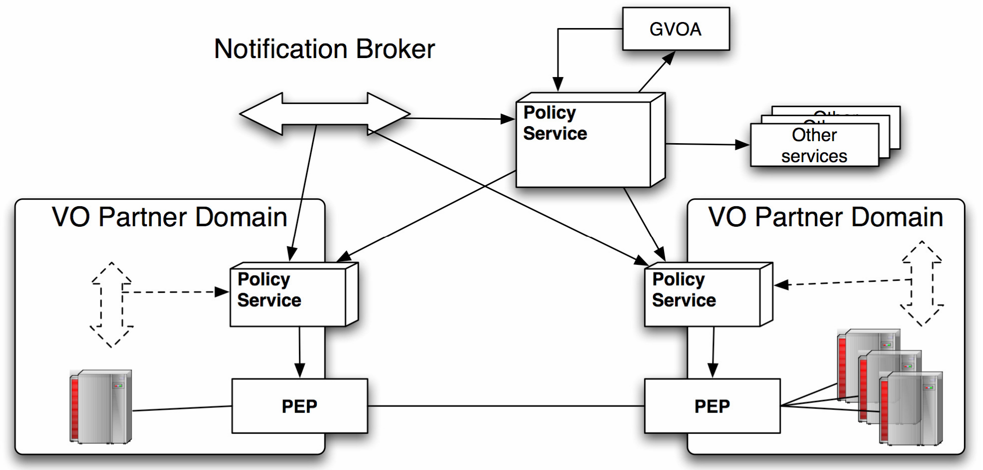

Figure 27 Policy Service Deployment, one service per partner

A third type of deployment consists in using the policy service as a simple Event-Condition-Action rule interpreter embedded in services of the TrustCoM administrative framework Figure 28. This permits each of the services to autonomously and concurrently react to the events occurring the in VO. It de-couples the services from one another and provides flexibility in constructing VOs that combine different sets of administrative services depending on the requirements of the VO. This type of deployment has been realised to a certain extent with the reputation service. However, it is in the more general case a more difficult type of deployment to achieve because it requires a notification broker that provides guaranteed delivery of notifications (which is not covered by the relevant standards) and because it requires a more difficult programming style (based on asynchronous events) of the VO functioning.

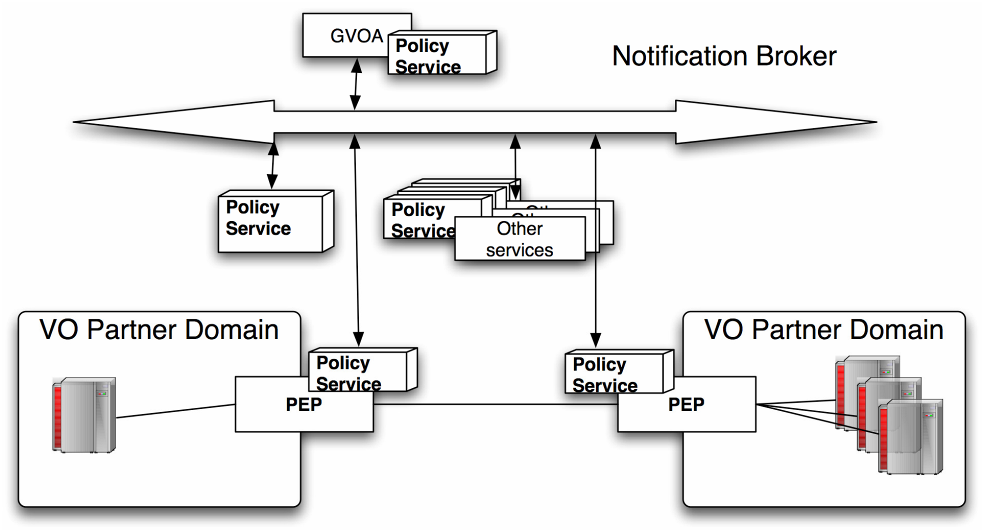

Figure 28 Policy Service; Embedded Deployment

V.3 Interaction Scenarios

In essence, policy services are used during the operation, evolution and termination phase of the VO. At the end of the formation phase the GVOA must contain a description of the roles and policies which apply to the VO. If any updates occur during the life-time of the VO such changes will need to be reflected in the GVOA. When prompted the policy service retrieves the policies pertaining to each role from the GVOA and deploys the authorisation policies to the PDP (if the binding is known) and enables the obligation policies.

V.3.a Identification

The policies are defined and stored as part of the GVOA. This is independent of the components of the policy subsystem.

V.3.b Formation

Once the policy specification is loaded into the policy service of the VO, the policy service will load and instantiate any policies on remote services if the specification requires it, it will instantiate the adapter objects for all the target services and events mentioned in the policy specifications as well as creating and activating the objects implementing each ECA rule. For access control policies the policy service load the policies into the appropriate policy decision point (PDP). These interactions are represented in the figure below.

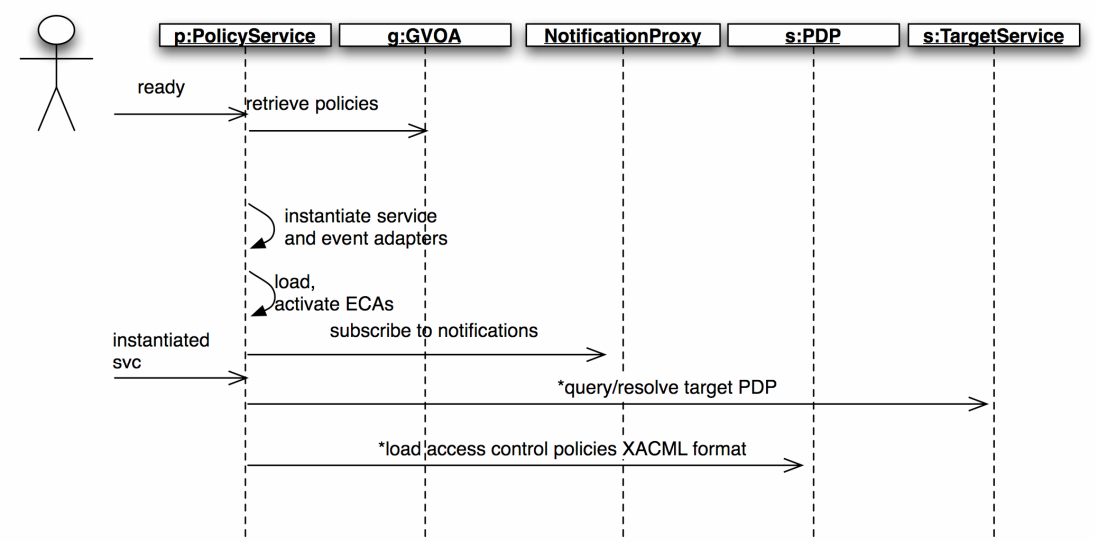

Figure 29 Overview of policy deployment

V.3.c Operation and Evolution

During the operation phase of the VO three scenarios are possible:

- A (human) administrator changes some of the operational policies of the VO. In this case the interactions follow the same sequence as indicated above. If the change has been initiated through the policy interface rather than the GVOA the policy service will generate a notification to inform all other components (in particular the VO Manager) of the change.
- A notification specified in one of the policies occurs. The applicable policies are triggered and the adaptation (management) actions are invoked on the target service.
- The PEP (see EN/VO infrastructure) receives a request for accessing a service. The PEP interacts with the STS in order to verify the authenticity of any security tokens presented and requests a decision from the PDP. The PDP encapsulates an authorisation decision engine which returns then a decision to the PEP.

These interactions are represented in the figure below. This representation remains agnostic to the VO type, target services or particular application environments. Typical examples of policy usage include:

- reacting to SLA Violations by either updating the reputation, requesting membership changes in the VO or enforcing compensation actions such as triggering the request for additional analysis cycles in the CE scenario,
- reacting to significant changes in reputation by either requesting membership changes, triggering the enforcement of secure audit of all interactions with that particular partner by reconfiguring the interceptor's handler chain or
- reacting to membership management events in order to trigger the instantiation of new services or reconfiguration of access controls.

In each of these examples different interaction paradigms with the target service may be employed and it is essential that the architecture of the policy service permits re-use of the generic ECA paradigm across all of these application scenarios. Internally to the policy service this can be achieved through the use of factories for the objects and adapters as exemplified in the following diagram, which shows the loading of a policy.

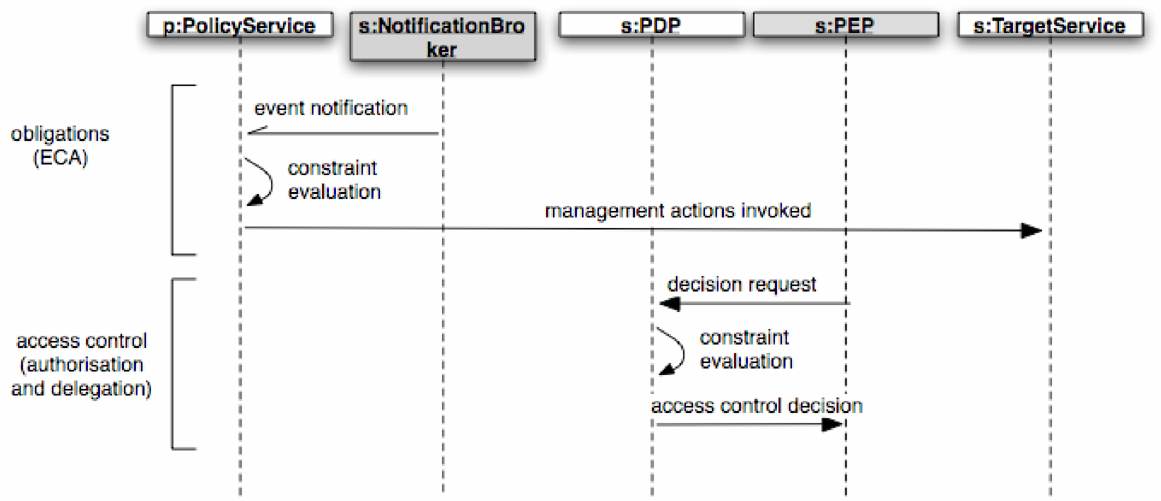

Figure 30 Summary enforcement for obligation and access control policies

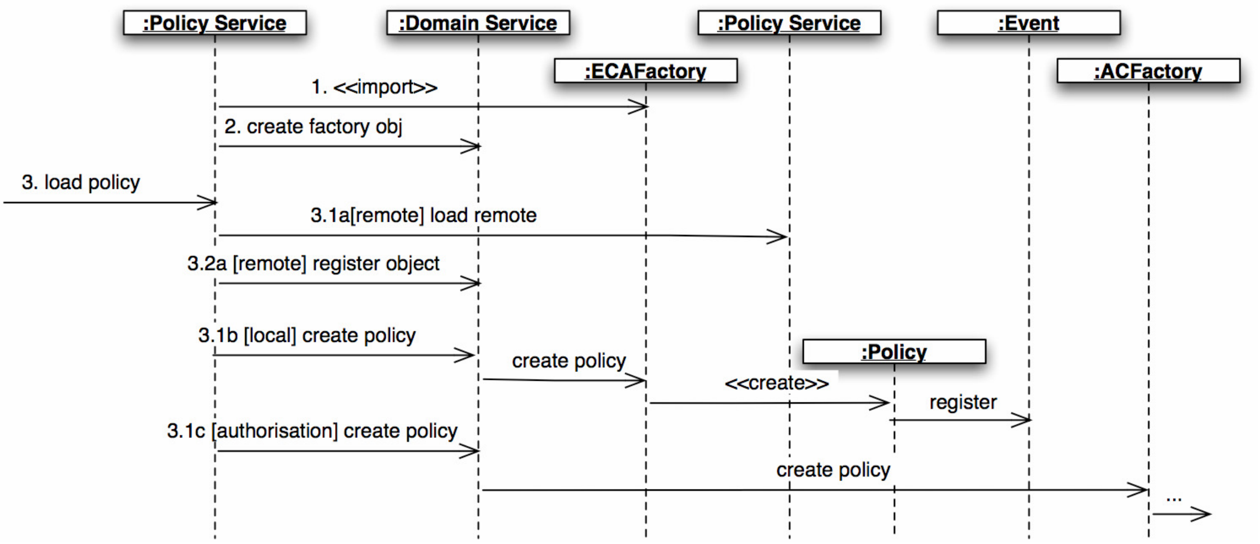

Figure 31 Policy creation invocation

In essence policies are objects which are created through the use of a factory object (ECAfactory) for obligation policies, (ACfactory) for access control objects. The specification received from the GVOA through the VO Manager contains the infosets needed for the parameterisation of the new instance. Every new instance of a policy leads to the creation of a policy object which can initiate any invocations on any object known to the Domain Service. Such objects may for example be adapters for external web-services (created through the instantiation of a SOAPadapter factory) or for the notification proxy when the required action is to raise a notification to the other subsystems. Note that policies are now objects too on which specific operations like enable, disable, unload, delete can be performed. In the case of access control policies these invocations would be transmitted to the relevant PDP. Furthermore, these actions may be performed as the result of another policy being triggered. Using such a paradigm it should be feasible to extent the scope of use of policies in a simple way through the definition and implementation of factories for new object adapters. Furthermore, although only obligation policies and access control policies are described here such a design offers the possibility to extend the scope of the policy service to different kinds of policies.

V.3.d Dissolution and Termination

Policies are equally a good means of triggering procedures related to the dissolution/termination of the VO. They can encode under which circumstances the VO can be dissolved and which actions need to be performed in order to remove all the elements created for the purpose of the VO.

V.3.e Conclusions

The policy subsystem in a VO framework should provide support for declarative specifications of behaviour which define the strategy, operations and access controls for the VO. Event-condition-action rules have been identified as the most flexible paradigm for supporting a wide range of adaptive strategies and trigger reconfiguration or administrative procedures inside the VO in response to events such as SLA violations, changes in reputation and changes in membership. The paradigm can also be applied in order to define a number of other behaviours including aspects of membership management, service categorisation and administrative re-configuration of web-service behaviour. To achieve this in a flexible manner the design of the policy service must be easily extensible to cater for a broad range of invocation protocols and the rules themselves must be clearly separated from the specific aspects of the implementation. From an access control view both delegation and authorisation policies must be supported. Their management should however be done in a uniform way with the other policy constructs for two reasons: so that grouping constructs such as roles and relationships can apply to both kinds of policies and so that adaptation policies such as ECAs can be used in order to trigger changes to the access control configuration.

V.4 Dependencies Overview

	Dependency
VO Mgmt	The General VO Agreement (GVOA) contains the policies that need to be distributed within the VO and should be updated with any changes to the policy specifications occurring during the operation phase.
EN/VO Infastructuire	Notifications corresponding to all events specified as part of the policies will be received from the Notification broker in order to trigger the adaptation actions specified as part of the policies. The policy service itself will generate notifications regarding policy deployment or failures of the adaptation actions.
	The policy enforcement point queries the policy decision point in order to obtain access control decisions on each invocation.

All	The adaptation actions performed by the policies may include membership management procedures, and reconfiguration actions (in particular on the security services). These actions which depend on the specific scenario for which the VO is configured need to be implemented by the corresponding services.

table 3: overview over the dependencies between Policy components and components of other subsystems

VI EN/VO Infrastructure BT

The EN/VO Infrastructure workpackage does not provide one logical, integrated subsystem, but rather a set of functionalities, realised as separate components, respectively packages. These functionalities contribute to the base layer of TrustCoM, allowing for interactions across enterprise borders (notifications, messaging and logging), distribution and management of service and component instances (service instantiator, service instance registry and information repositories), as well as supporting the discovery process (discovery service and repositories).

Please refer to deliverable D16 for an overview over the underlying concepts, respectively to D18 for more details about the usage of WS specifications in the EN/VO Infrastructure subsystem.

VI.1 Full Components Overview

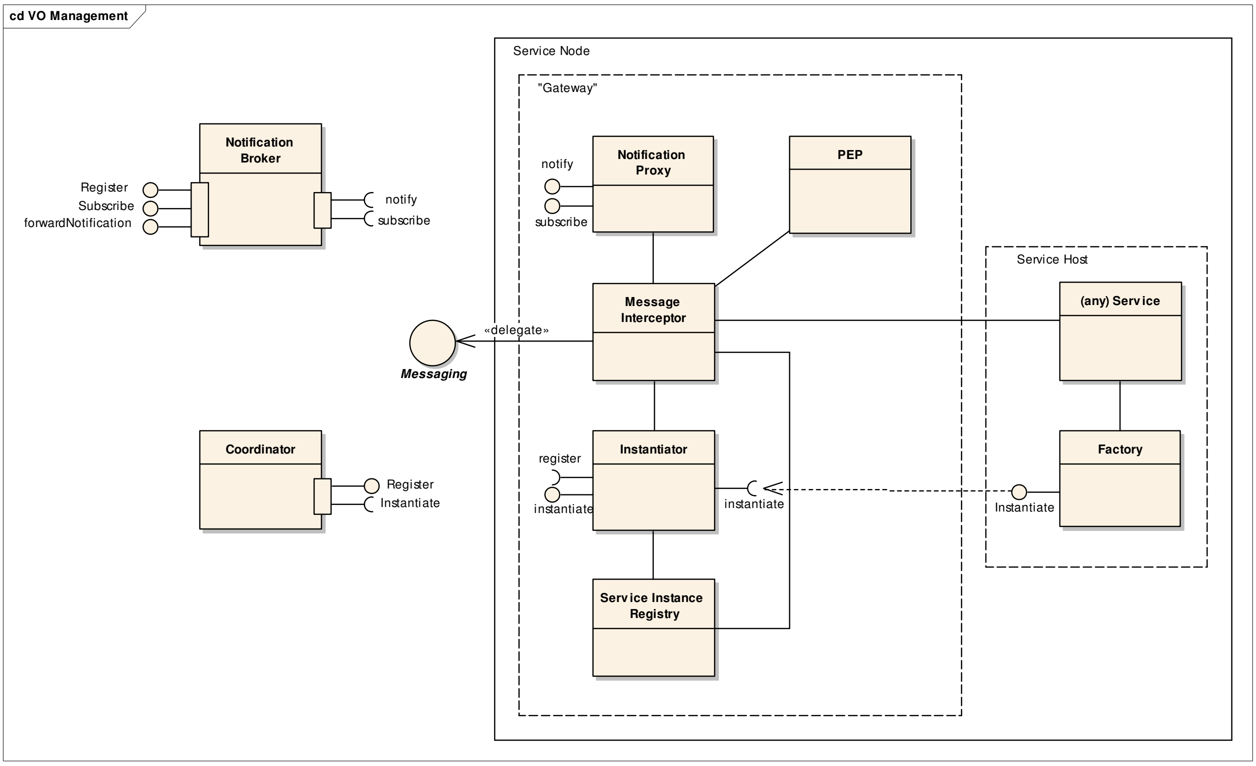

Figure 32: Composite Structure diagram of the components related to instantiation and communication.

Such VO architecture would need:

- A messaging infrastructure that supports various interaction patterns (e.g. one-way notification, request-response, brokered delivery, mediated and unmediated coordination protocols). The infrastructure should be product and platform neutral.
- Operational management and security infrastructure services (service and network resource discovery, security, message reliability, transaction management, service composition, federation, and policy enforcement)

- Service exposure which includes the ability to deploy the applications, advertise them as capabilities and enable on-demand creation & management,

The framework infrastructure can therefore be broken down into three major layers as shown in the diagram hereafter:

- The infrastructure services
- The service bus
- The application exposure layer

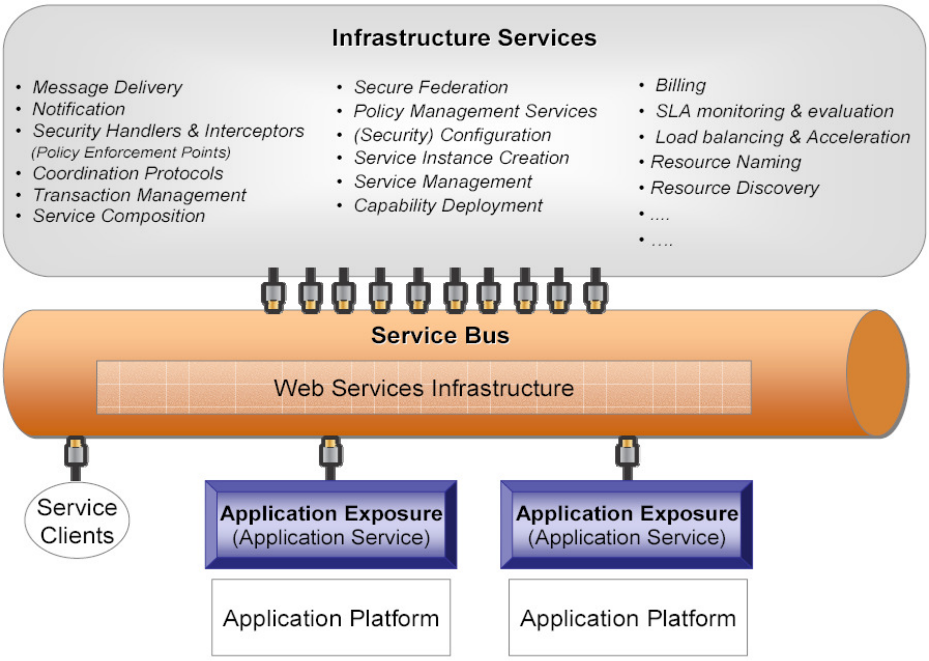

Figure 33: Framework infrastructure overview

In addition to the basic functionalities the infrastructure services provide, the latter can be combined to meet more complex needs in a VO – for instance – secure federation. This will be dealt with later on. Let's now have a look at the elements that constitute the base of a VO infrastructure.

VI.2Basic VO elements

VI.2.a The enforcement point

Concept

Enforcement is carried out by SOAP 'interceptors' inserted into the message path between sender and recipient. The insertion is achieved by dynamically selecting and chaining handlers (a.k.a. interceptors) based on the contents of the SOAP message and statements contained within the enforcement policy. The enforcement policy is assigned to every instance of the resource by the Configuration Manager via Management Interface.

There are four types of policy present in the system: *Enforcement Configuration Policy (ECP)* (describes the enforcement logic for a particular instance of the resource); *Capability Exposure Policy (CEP)* (used to advertise the security capabilities of a particular resource); *Interceptor Reference Policy (IRP)* (reflects the mapping between a particular operation type and the software code responsible for the enforcement of this functionality; it can be altered only by a deployment of a new enforcement package); *Utility Services Policy (USP)* (contains locations of the infrastructure services which might be invoked during the enforcement process).

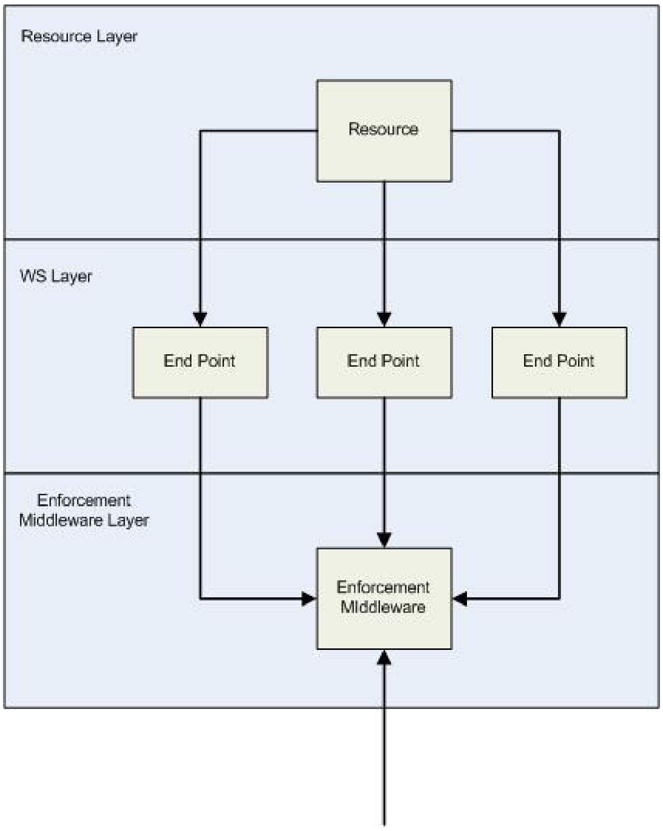

Figure VI-34. High-level view of the concept

Policy structure

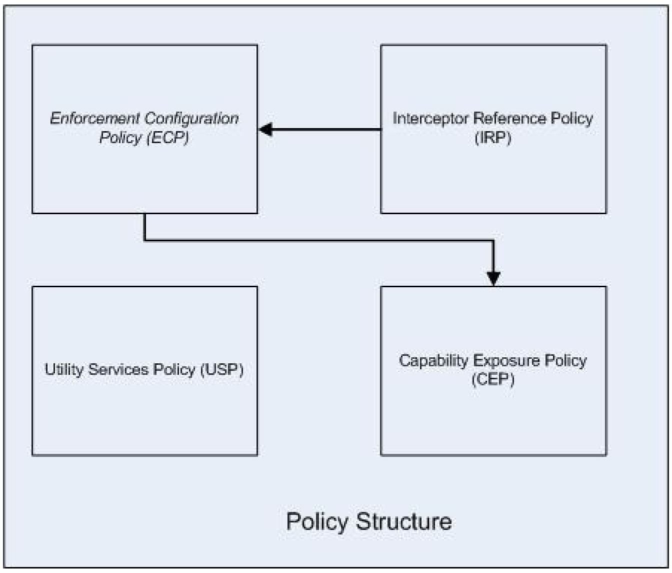
Enforcement Configuration Policy (ECP)
Interceptor Reference Policy (IRP)
Utility Services Policy (USP)
Capability Exposure Policy (CEP)
Policy Structure

Figure VI-35. The structure of a typical security policy

The enforcement point at the endpoint level

The standalone architecture assumes that all of the components of the enforcement middleware are implemented and deployed as part of the same software package (Figure VI-36). This option is especially beneficial in the situations when the ultimate recipient and the enforcement middleware are placed on the same host and no routing is required. Then the Enforcement middleware is plugged directly into the SOAP engine which is used to provide web service wrapper for the protected resource. In this case:

a. Enforcement actions take place between the resource protected and the network endpoint that is exposed on the host implementing the resource virtualisation, i.e. at the host where the network service exposing the resource has been deployed.

b. The enforcement middleware is transparent to both the client and the resource protected.

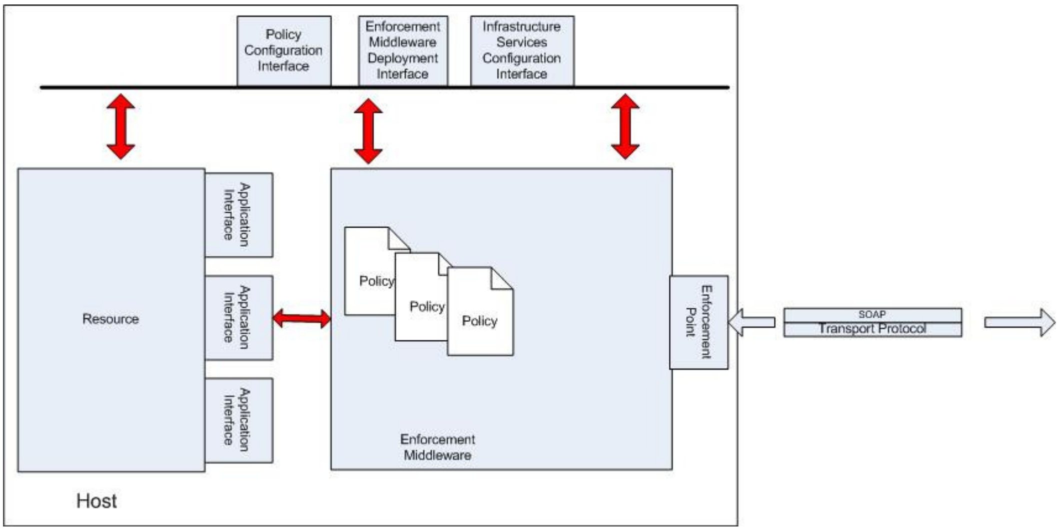

Figure VI-36. System architecture with security middleware deployed locally to the protected resource

Standalone enforcement point

The second design option is to have an enforcement middleware on a separate host from the protected resource (Figure VI-37). In this case the enforcement middleware is deployed as standalone intermediary. As in the previous case, all interceptors are implemented as part of the same software package. The enforcement middleware deployed in this fashion can either be visible on the network and have its own network address or take advantage of protocol binding techniques and be transparent to both the client and/or the resource at the service/application level of the network. Examples of standalone intermediaries include:

- SOAP intermediaries which are visible to the application network and
- HTTP or TCP routers which are specific to intercepting SOAP messages over HTTP or TCP, respectively, and are transparent at the SOAP layer but visible at the HTTP or TCP layers, respectively

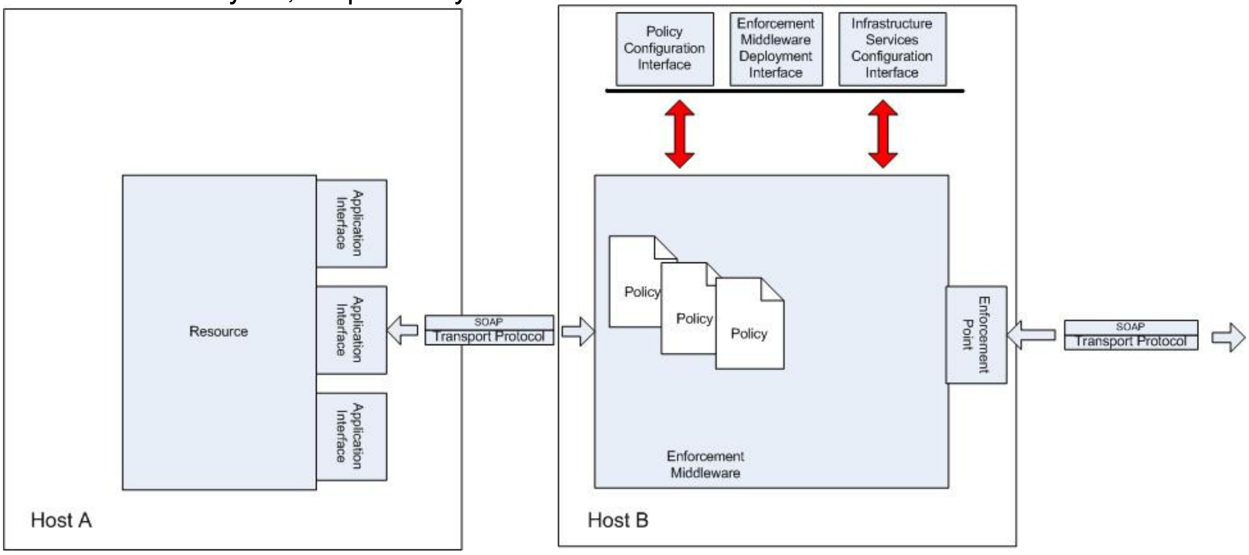

Figure VI-37. System architecture with security middleware deployed remotely to the protected resource

Enforcement process

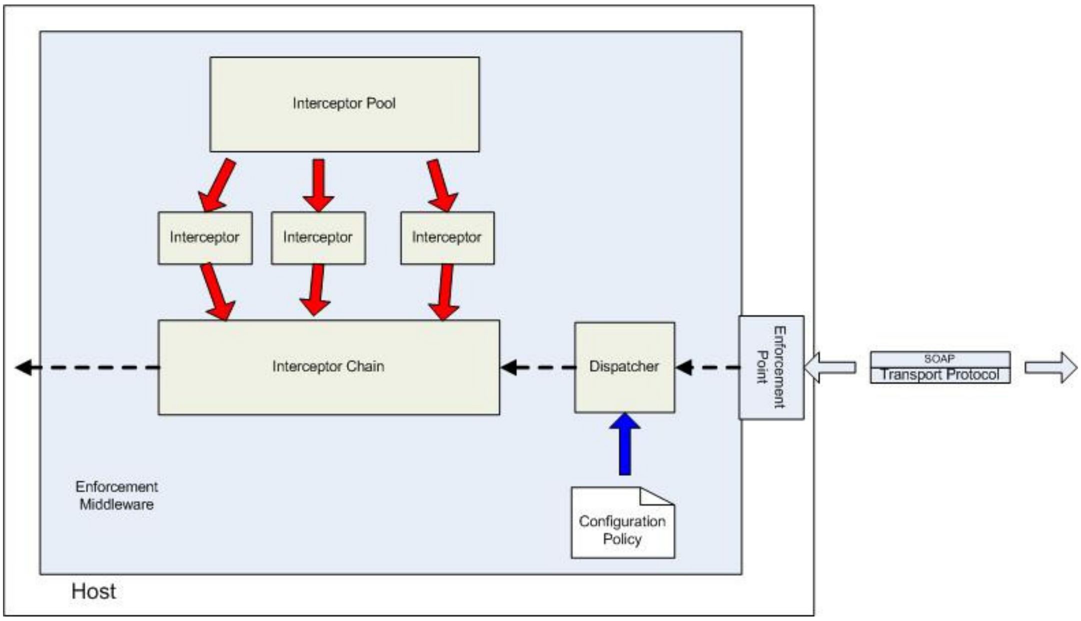

Figure VI-38. The process of enforcement where the interceptor chain is formed out of interceptors deployed locally

The process of the enforcement which relies on the construction of the configurable interceptor chains can be seen in the Figure VI-38. The composition of interceptor chains is process based on the amalgamation of the message content analysis and the security requirements of the protected resource. Based on the outcome of this fusion the interceptors are selected from the interceptor pool and inserted into the chain. As mentioned earlier the interceptors in a chain may be deployed locally or they can be distributed over the network and be invoked remotely.

Security configuration adaptation

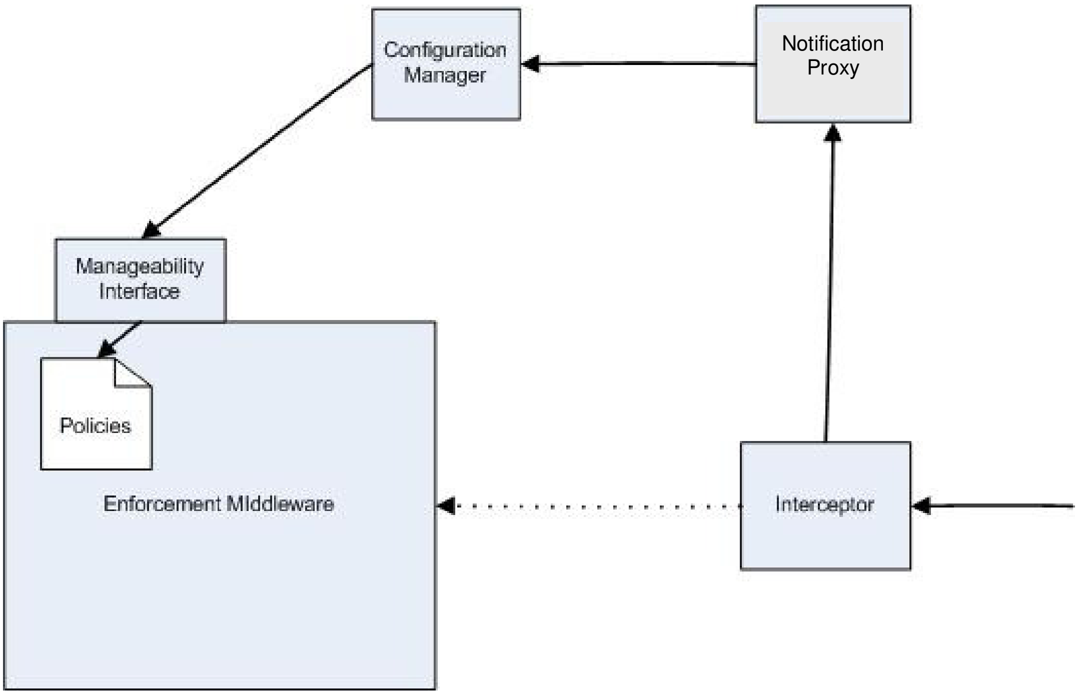

Figure VI-39. System components responsible for the security configuration adaptation

The process of adaptation is described in Figure VI-39. Once the failure has been detected enforcement middleware produces the notification and submits it to the Notification Proxy. Notification Proxy then notifies the Configuration Manager about the event. Based on the information received and its internal supervision policy the Configuration Manager can react adequately and update a configuration policy, say ECP. This update will have an immediate effect and will influence the way interceptor chains are constructed

VI.2.b The messaging layer

Overview

Services that want to communicate or engage in a conversation will be able to send their requirements regarding their security policies to our third party, in exchange of guarantees (in form of liabilities) that these policies will be upheld. The implementation of services will be carried out as a set of protocols whose execution will provide certain guarantees. The third party infrastructure will handle messages in certain predetermined ways in order to guarantee a message's anonymity, sender's privacy, accuracy and others (see Figure VI-40).

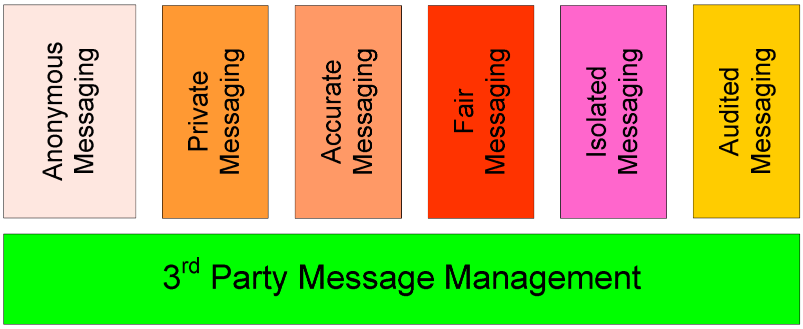

Figure VI-40. Third Party Infrastructure Services

Services will be used to guarantee certain attributes of the message exchanges as part of a wider conversation.

Routing component

The routing component is one step down from the enforcement point. In the example of message delivery happening over the HTTP protocol, the routing component reads the HTTP headers (message headers) and determines where to send on the message it has just received. The routing component can be combined with the enforcement point to deliver more complete scenarios. Such an example would be upon the reception of a non encrypted message, the routing component would forward the message to an end service that does not require messages to be encrypted.

Anonymity

This service will make use of current web service standards to mediate the delivery of a message from one party to another without revealing the sender's identity if this not desirable. The third party infrastructure can, in this case, guarantee the anonymity of the message while at the same time guaranteeing other properties such as message integrity.

Privacy

The proposed infrastructure will provide guarantees regarding a message's integrity and confidentiality as well as guarantees about the accountability of the sender without compromising one's sensitive information.

Accuracy

Third party infrastructure will provide guarantees about the accuracy of the message. Senders and Recipients will have guarantees that the integrity of the message has not been compromised. Although there are a number of mechanisms for providing message integrity, there is an additional challenge in providing message integrity in the light of our earlier discussion on anonymity and privacy.

Transaction Isolation

Transaction Isolation is a mechanism that enables two services to participate in several VOs, even if the message exchanges are identical for all of these VOs. We want to distinguish between different sets of transactions that are associated with a particular objective even if the transactions are between the same services. Such a mechanism would let services participate in several different contexts, each of which may implement different policies regarding the security of those messages. Services will also be able to process messages from different contexts enhancing at the same time the degree of dynamicity of those services. Services will therefore not be bound by VOs, which would normally inhibit their participation in other VOs, but by context which is a more flexible concept to deal with.

VI.2.c Coordination

The coordination infrastructure provides coordination protocols able to determine and to agree on whether the common goal among services has been reached. Some of the service coordination (such as reliable message exchanges, e.g. as in the OASIS WS-RM specification) does not require explicit coordinator, whereas transaction protocols such as service instantiation and service instance destruction offer examples where mediation is required in order to assure coordinated behaviour (including the reservation / release of VO infrastructure resources). The coordination infrastructure offers both the message correlation framework and the mediating services that may be required in order to implement the coordinated message exchanges between web service instances that support the operation of VO services.

Coordination context is of a form of a complex XML element which can be adapted to include various contextual information. Some of the representative examples with respect to the TrustCoM architecture are:

- **VO formation and enactment**. By including a VO identifier as a part of the context, one can distinguish management –related interactions to the scope of the particular VO. In addition, a separate context is created to support the execution/ interactions of the application services (for that VO).
- **Service instantiation**. Since service instantiation includes creating a token, setting security policy and defining the SLA for the service instance, a number of components from different subsystems are included in the process. There may be a number of instantiations (of same/different service types and for the same/ different VOs) occurring concurrently. Use of the context would allow setting a clear separation between the interactions related to different activities. An identifier of the service instance being created may be an obvious choice to include in the context in order for services to be able to distinguish different contexts.
- **Federation of trust realms**. This refers to dynamic creation of a group of participants, where different participants may belong to different trust realms. The member requesting a group creation is activating a context for the group (using WS-C Activation service). The context has a unique identifier and is included in messages (using WS-C Context element). The context is passed with every message relating to the group. All

recipients of a contextualised message attempt to register with this context (WS-C aware PEP and WS-C registration service). The context has a token associated with it which includes a public key that is bound to the context and successfully registered participants receive the corresponding private key upon their successful registration. This token is unique to the context identifier and the client activating. Among others, the context/key-pair mechanism is used in order: to protect content that is shared among participants, and to correlate message exchanges in the same group.

VI.2.d Notification service

The notification infrastructure provides the capabilities for publishing, subscribing to and managing subscription to notifications about single events or families of events of interest. It also provides mechanisms for organising notification message types so that subscribers can conveniently understand what messages they can subscribe to, as well as capabilities to broker notifications and transform from one family of notification types to another.

The initial Notification subsystem consists of two components:

- the Notification Proxy (local to the nodes on which the notification producing and consuming services reside) and
- the Notification Broker.

Their task is to allow notification passing in the VO Infrastructure with a minimal implementation effort for the service providers – this involves topic handling, notification forwarding, subscription management, etc. In general, the Notification Proxy takes over notification management at the service provider nodes, while the Broker is responsible for managing multiple topic sources and subscriptions to them. The Broker may also act as an intermediary for notification forwarding, if subscribers are unknown to the notification source.

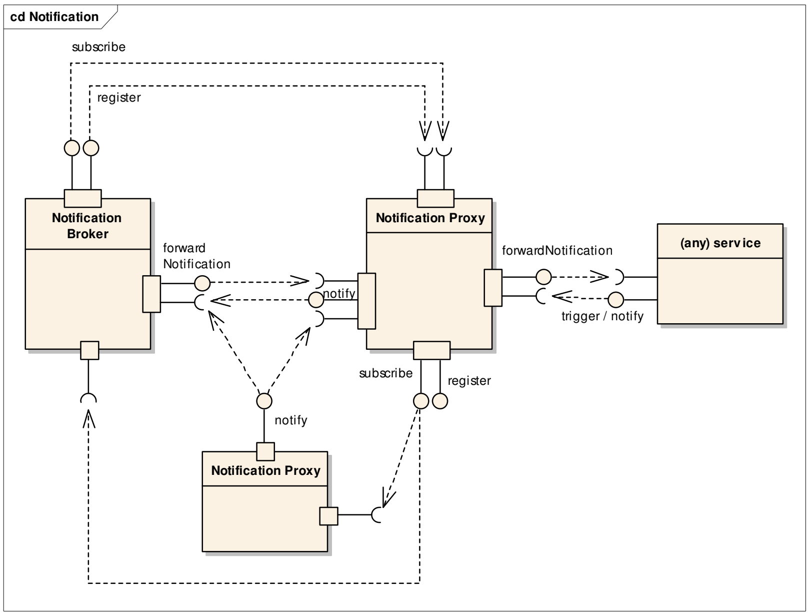

Figure VI-41. Notification infrastructure overview

VI.3 Advanced VO functionalities

With the interaction between the basic components of the VO infrastructure, one can achieve further accomplishments. The following paragraphs give an overall overview of component interaction and the services these interactions provide.

VI.3.a Service instantiation

Process of service instantiation is described via example taken from the collaborative engineering scenario.

The basic storyline of the scenario is that a VO member wants to expose a new service to be used in the VO. The following steps comprise the application part of this activity:

1. On-demand creation of a service instance: Requester (from a GUI, or a user agent representing the client, or another service). Request is sent to the dedicated Instantiation service.
2. Request to Service Factory for creation of new Service Instance (i.e. creation of new ResourceProperties document, in WSDM/WSRF terms). This includes creation of EPR for the instance, which is passed back to the Instantiator.
3. Creation of the Service Endpoint for the service instance. This is a SOAP interceptor which is the access point for the service instance and performs the function of the PEP.
4. Configuration of the PEP/ Service Endpoint. This step also includes binding to other TrustCoM services. It can be performed as the following **independent** activities:
   a. Security-related: creation of the token for the service instance and binding to the STS);
   b. Policy-related: configuration and activation of the policy instance for the service instance, and binding to the PDP);
   c. SLA-related: configuration and activation of the SLA instance for the service instance, configuration of monitors.
5. Update of the VO membership state (Active SLA, services available).
6. EPR of the new (configured) Service Instance returned to the Requester.
7. Requester invokes the service - test service operation:
   a. Successful request using new service instance: show access control, monitoring in operation.
   b. Unsuccessful request or violation using new service instance: show access control, monitoring in operation.

Before any interactions are allowed between different services, they need to register to the activity with the Coordinator. These “control” messages are shown in a different colour and without numbering). In summary, Instantiator (which initiates the activity), activates the context for this activity with the Activation service of the Coordinator. Afterwards, the context is passed to the services which need to participate the activity, causing them to register. After that, the interactions follow message exchange as defined for this particular transaction.

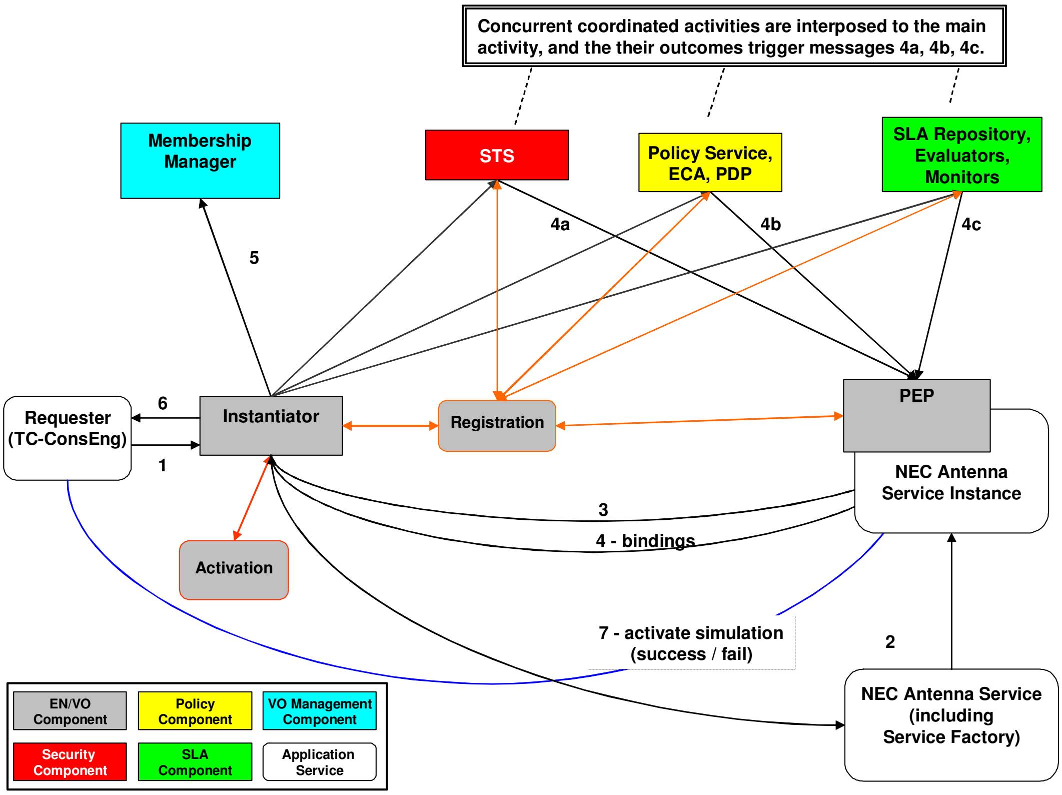
Concurrent coordinated activities are interposed to the main activity, and the their outcomes trigger messages 4a, 4b, 4c.
Membership Manager
STS
Policy Service, ECA, PDP
SLA Repository, Evaluators, Monitors
4a
4b
4c
5
6
1
Requester (TC-ConsEng)
Instantiator
Registration
PEP
NEC Antenna Service Instance
3
4 - bindings
Activation
7 - activate simulation (success / fail)
2
NEC Antenna Service (including Service Factory)
EN/VO Component
Policy Component
VO Management Component
Security Component
SLA Component
Application Service

Figure VI-42. High-level diagram of the “Service Instantiation” activity. Application services are shown in the context of CE scenario. Please note that all the services (may) contain the PEP, but it is shown only for the new service instance

For an activity with a lot of interactions and a lot of services involved, the transaction (and therefore any potential rollback) can become complex. Therefore, it is advisable to separate independent interactions as separate, interposed activities. In the example given in Figure VI-42, “security, policy, and SLA part of” the Instantiation activity includes a number of interactions within the corresponding subsystem, without dependencies on the full process. Therefore, these are broken down in the separate (interposed) transactions, and include interactions as outlined in the steps 4a, 4b, 4c above. Typically, this is done in the following way: a service that receives registration request from Instantiator will register for the main activity, at the same time activating its own interposed activity, with a different coordination context. The participants registered with this new context can interact only within the interposed activity, and do not have the access to the main activity. Upon completion, the result of the interposed activity is communicated back to the main one via the initiator of the interposed activity.

Such an approach decreases dependencies and coupling of different stages of the activity and makes any potential rollbacks easier, as the transaction itself is less complex.

VI.3.b Federation support

Enterprise-wide services are specific to a trust realm, but may interact with any other services from the enterprise. In addition, STS services interact across trust realms in order to establish trust relationships required for operation of context-management and group services. One is to assume the existence of this basic trust relationship (i.e. it has already been established between STSs from different trust realms, via some of the already known federation mechanisms).

Context management services may operate across trust realms, making use of the existing basic secure federation. Normally, registration services are those that interact over different realms, as typically only one activation service is used (normally from within the realm of a network service initiating the group).

Context management service may be provided by a third party, following the establishment of the federation which includes realms of the enterprises that contribute network services to the group, and the realm of the third party provider.

Group includes network services (typically web services) that are brought together for the purpose of an activity.

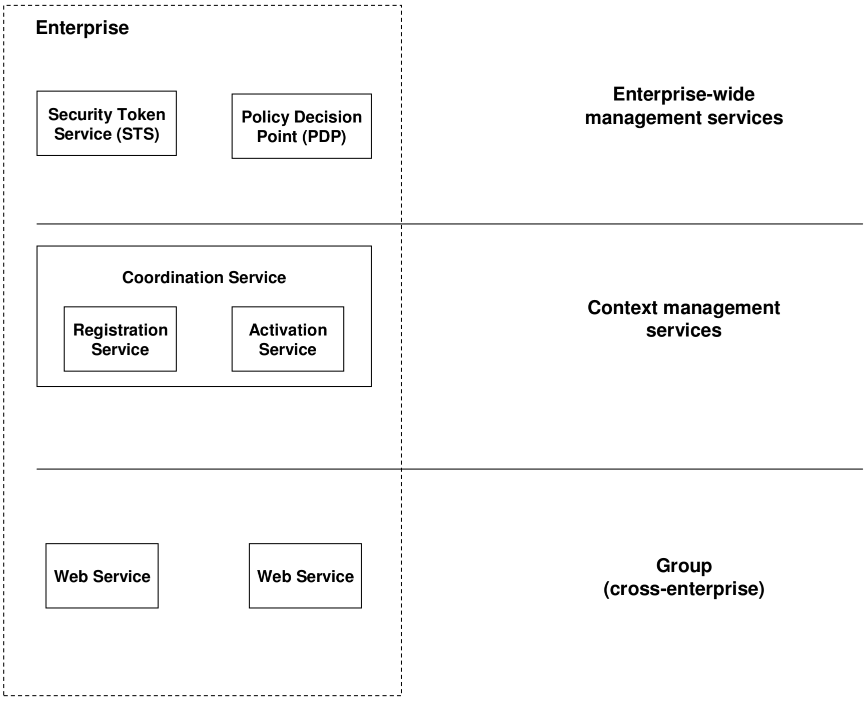
Enterprise
Security Token Service (STS)
Policy Decision Point (PDP)
Enterprise-wide management services
Coordination Service
Registration Service
Activation Service
Context management services
Web Service
Web Service
Group (cross-enterprise)

Figure VI-43. Basic system

The following figure displays the system for group-bootstrapping across two trust realms, with a single coordination service. It is assumed that STSs from different trust realms have means to establish trust relationship (e.g. some of the WS-Federation models).

The following summarises main interactions shown in Figure VI-44: upon the activation request from WS1, credentials presented to the activation service are evaluated by respective PDP and STS services.

If the request is accepted, WS1 is returned coordination context for the group. This enables WS1 to register to the activity (if required). In a similar way, WS2 is enabled to register to the group when the context is passed from WS1 to WS2 (as a part of the application message). Similar to the activation request, registration acknowledge is subject to (and preceded with) the evaluation of the credentials by respective PDP and STS services.

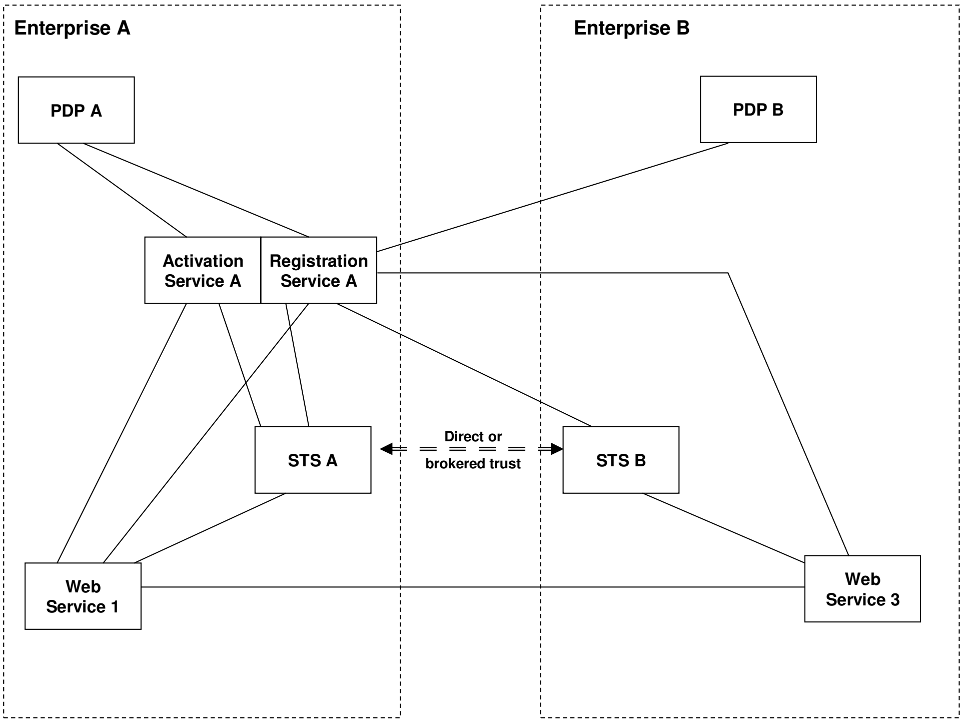

Figure VI-44. High-level Overview of Interactions for Creating a Federated Group of Services

As a part of creation of the coordination context, activation service may interact with other (security) services in order to provide group-specific, security-related information which will augment the basic context. While the basic context (as prescribed by WS-Coordination specification) allows a potential participant to request registration to the federated group (at the registration service), the security-related part of the context (which would normally be of the form of an encryption key and/ or a token) would enable a participant to secure the interactions within this particular context, upon successful registration.

VI.3.c Validation/ Authorisation of a SOAP Request

Policy Enforcement Point-side

- Overview

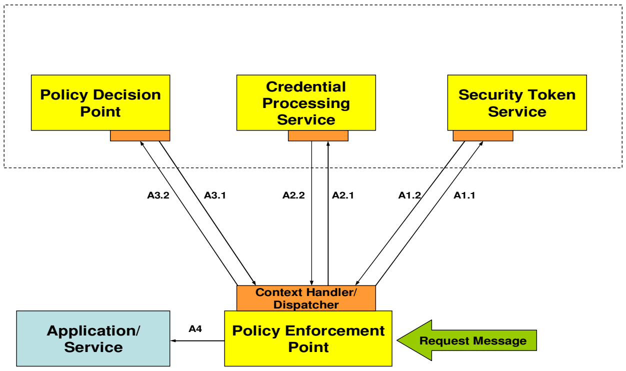

Figure VI-45. PEP-biased Model

- Description

This model relies on PEPs to facilitate interaction and control data flow with miscellaneous federated security services. Potentially this model requires a PEP to have knowledge about the available security services in a trust realm and of the interaction protocols they support. This model also delegates to the PEP the responsibility for maintaining the order of interactions necessary to fulfil enforcement duties, error handling and management of notifications. The typical order of invoking the services would be as described in the paragraph below (see also Figure VI-45 for overview).

- Incoming messages:

1. STS – the PEP protecting the recipient asks the STS to validate the token and to approve the claims included in the token (A1.1, A1.2). Depending on the type of tokens different STS services may be invoked.
2. CPS – credentials supplied with the request are further processed if needed (A2.1, A2.2).
3. PDP – an authorisation decision is made about the actions requested (A3.1, A3.2). Commonly this requires the action identifier (and often action parameters) to be sent to the PDP along with attributes corresponding to the validated claims and other contextual information (e.g. time, transaction context, etc). Depending on the PEP/PDP protocol, the PDP may be allowed to return an obligation statement which may trigger an additional round of collecting tokens, processing credentials and requesting for a further authorisation in view of the additional evidence provided.

- Outgoing messages:

1. PDP – the PEP acting for the requestor obtains an authorisation to proceed with the request. Such authorisation decisions are based on policies that are specific to the realm of the requestor, and they are not necessarily concerned with whether the recipient will authorise the request or not. In particular, the requestor may not be allowed to request an action even if that action could have been authorised at the recipient's side. Depending on the architecture of the requestor's trust realm there may be cases where the PEP will have to validate the requestor's credentials prior to this step. Depending on the PEP/PDP protocol, the PDP may also be allowed to return an obligation statement which may trigger an additional round of collecting credentials prior to successful authorisation.
2. CPS – the requestor obtains the set of credentials associated with the request.
3. PDP – the requestor obtains a token asserting a collection of claims that match the credentials provided.

The major benefit of this model is the flexibility it provides and the fact that it can scale to a chain of PEPs focusing on the different types of enforcement actions within the same realm. It allows for deployments where STS, PDP, CPS and PEP are loosely coupled which can be advantageous when upgrading any of these services. This is beneficial when a potentially large number of PEPs need to share a potentially smaller number of STS, CPS, PDP services. It also facilitates the reconfiguration of PEP in order to invoke different types of STS, CPS and PDP depending on the contents of the message (e.g. token, credentials provided, types of action requested, etc.), the state of the enforcement and the context of the interaction. This is particularly important for large-scale decentralised networks where the infrastructure may need to adapt to contextual changes and where the availability of some of these components cannot be always guaranteed. Finally, it facilitates the inclusion of trusted third party services for validation or processing of credentials if needed.

Disadvantages of this model include the fact that the PEP is required to implement intelligent decisions or use complex configuration information in order to manage the sequence of enforcement actions. Furthermore, caching of security related information for the duration of conversations between requestor and recipient may be required .to avoid the traffic overhead incurred if a single request requires several conversations between security services to take place.

Policy Decision Point-side

- Overview

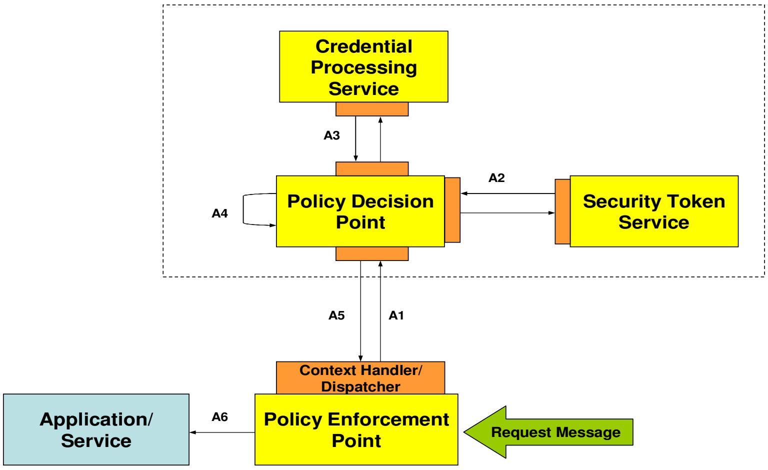

Figure VI-46. PDP-biased Model

- Description

This model places more emphasis on the processing of security information and the reasoning performed by the PDP. In this model the PEP effectively has only knowledge about available PDPs and the protocols they support. Token and credential validation and processing are performed as a part of the authorisation policy evaluation within the PDP.

During the processing of an incoming message, PEP sends all evidence available to the PDP, unprocessed, in the context of an authorisation request. The response from the PDP includes the authorisation decision.

During the processing of an outgoing message, PEP, sends an authorisation request to the PDP and obtains an authorisation response that includes any tokens required as evidence to support the authorised request.

In this model the actual PDP needs to implement STS/CPS functionality (or to invoke external STS and CPS services) and handling of message contexts. Since the interactions with STS/CPS are out of the chain of enforcement actions it may be possible for it to execute validation of credentials and evaluation of authorisations concurrently in order to reduce the delay times.

However, this model is less flexible than the one described earlier. For example, the introduction of a new STS for validating new token types will require upgrade of the whole

PDP configuration. It also allows for a tighter coupling between the authorisation policies and the types of credentials that may be used and fusion of authorisation, and privilege or attribute management schemes, which may lead to architecting systems that are difficult to maintain and manage. In particular, it encourages models where the authorisation policy pre-determines the type of evidence that can be processed, which increases the difficulty of federating Trust Realms.

Security Token Service-side

- Overview

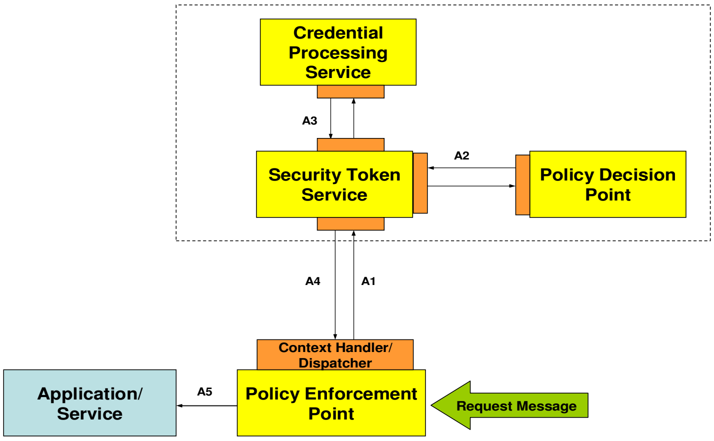

Figure VI-47. STS-biased Model

- Description

The approach taken in this model is interchanges the roles of PDP and STS relative to the PDP-biased case. In this model the STS needs to implement PDP/CPS functionality or the necessary protocols for invoking external PDP and CPS services. For an incoming message, PEP requests the validation of tokens for the purpose of a requested action, and the result of the validation depends on the tokens being valid and the action being authorised. For an outgoing message, PEP requests the issuing of tokens for a requested action and the result depends on the result of the authorisation decision and the success of token issuing.

Similarly to the previous model the interactions with other security services can be executed concurrently in order to reduce delays. For example, the STS may initiate an authorisation request assuming the validity of credentials while it is validating and processing the credentials. Then accepting a positive authorisation response will depend on the result of the credentials validation.

In contrast to the PDP-biased model, this model allows deployments where the PEP has a choice of which STS to invoke depending on the type of tokens provided. Furthermore, the model does not encourage authorisation policies whose semantics depend on the type of evidence provided. In other words the authorisation policy evaluation and the processing/validation of evidence can be decoupled in this model.

However, this model also suffers from scalability and flexibility problems similar to the PDP-biased case. In this model the STS is now expected either to evaluate authorisation policies or to implement protocols for requiring and obtaining authorisation decisions. Furthermore, the model suffers from the fact that it increases the dependency on the specific types of token supported by some STS.