Deliverable

D29

D35

D36

TrustCoM Framework V2

AL1 – TrustCoM Framework

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

31/01/2006

V2.0

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

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 V2**	
Version:	**v2.0**
Document reference:	
Official delivery date:	**31/01/2006**
Actual publication date:	
File name:	D29_35_36 TrustCoM Framework V2 (with comments).doc
Type of document:	Report
Nature:	official deliverable

Authors:	Michael D. Wilson, Alvaro Arenas, Jochen Haller, Lutz Schubert, Tobias Mahler, Joris Claessens, Pablo Giambiagi, Ingo Weber
Reviewers:	Paul Kearney, Yücel Karabulut, Stefan Wesner, Michael D. Wilson, Jakka Sairamesh
Approved by:	

Version	Date	Comments
V1.1	06/12/2005	New document structure
V1.2	07/01/2006	Business Model section
V1.3	10/01/2006	Introductory sections

V1.4	16/01/2006	Added comments on legal aspects; added WSLA profiles
V1.5	18/01/2006	Relationship overview, WSCDL by Ingo
V1.6	25/01/2006	Pre-final version of Relationship chapter, updated contract section
V1.7	31/01/2006	Pre-final version of Deployment chapter, initial Introduction to profiles section
V1.8	01/02/2006	General polishing, added WS-Trust & SAML
V1.9	08/02/2006	Reaction to comments from internal review:
		Refined sections I.2, I.3
V2.0		Final Version

Table of Content

Introduction ... *7*

I The TrustCoM Conceptualisation ... *9*

I.1 The TrustCoM Vision ... **9**

I.2 The Trust Model in the TrustCoM Framework CCLRC ... **9**

I.3 The Business Model of TrustCoM SAP ... **11**

I.3.a Business Processes ... 12
I.3.b Structure of Virtual Organisations ... 16

I.4 The Contract Model of TrustCoM CCLRC, NRCCL ... **17**

I.4.a EN Contract ... 19
I.4.b VO Contracts ... 19
I.4.c Contracts and the VO lifecycle ... 20
I.4.d Examples from the TrustCoM test bed scenarios ... 20
I.4.e Drafting EN and VO contracts ... 21

I.5 Usage Scenarios / VO Lifecycle Phases CCLRC ... **22**

I.5.a Establishment of a Virtual Organisation (Identification and Formation) ... 22
I.5.b Normal operational work ... 23
I.5.c Dynamic addition of an organisation during operation ... 23
I.5.d Dynamic removal of an organisation during operation ... 24
I.5.e Replacement of a participant by another during operation of the VO (Evolution) ... 25
I.5.f Dissolution of the Virtual Organisation ... 25

II The TrustCoM Architecture CCLRC ... *27*

II.1 From Concepts to Architecture CCLRC ... **28**

II.1.a Abstract Structure HLRS ... 29
II.1.b The Subsystem Segmentation ... 31

II.2 The Relationships with respect to the individual VO lifecycle phases ... **34**

II.2.a Preparation ... 35
II.2.b Identification ... 36
II.2.c Formation ... 38
II.2.d Operation ... 40
II.2.e Evolution ... 42
II.2.f Dissolution ... 46

II.3 The Relationships in the underlying EN/VO Infrastructure ... **48**

II.3.a Setup (Formation) ... 48
II.3.b Messaging (mostly Operation) ... 50
II.3.c Reconfiguration (Evolution) ... 52

III Towards a Deployment Model ... *54*

III.1.a Establishment of a Virtual Organisation (Identification and Formation) ... **Error! Bookmark not defined.**
III.1.b Normal operational work ... **Error! Bookmark not defined.**

III.1.c Dynamic addition of an organisation during operation............ **Error! Bookmark not defined.**
III.1.d Dynamic removal of an organisation during operation............ **Error! Bookmark not defined.**
III.1.e Replacement of a participant by another during operation of the VO (Evolution) **Error! Bookmark not defined.**
III.1.f Dissolution of the Virtual Organisation ....**Error! Bookmark not defined.**

III.2 Deployment Overview (One possible VO Instance).. 58
III.2.a Service Level view.. 58
III.2.b Component Level view 60

III.3 A Potential VO Instance 65
III.3.a Gateway Functionalities.. 66
III.3.b Main Functionalities.. 67

IV Profiles EMIC, CCLRC 79

IV.1 WSLA SICS 80
IV.1.a Introduction 80

IV.2 WS-Trust & SAML EMIC, ETH & UoK 84
IV.2.a Namespaces and supported specifications 86
IV.2.b WS-Trust 86
IV.2.c SAML Assertion Profile... 89
IV.2.d Custom elements 92

IV.3 WSCDL SAP.. 93
IV.3.a Overview 93
IV.3.b WS-CDL Language Elements and Representation 94
IV.3.c Annotation of Trust, Security, and Contract (TSC) Tasks 97

IV.4 XACML SICS 97
IV.4.a Attributes... 98
IV.4.b Policies 101
IV.4.c Transport formats 103

IV.5 ECA-Policies IC... 104

Glossary all.. 106

Key to diagrams 109

References... 114

Introduction

The TrustCoM project[1] is developing a Framework for Trust, Security and Contract management for dynamic virtual organisations operated through an open service architecture. The Framework includes a way of conceptualising the Trust, Security and Contract issues associated with dynamic virtual organisations, an architecture in which the operating open services can be implemented and profiles of proposed and standardised web service specifications tailored to the VO application. The project will also produce a reference implementation of the architecture, and demonstrators whose evaluation will show the strengths and weaknesses of the framework.

VO Management as developed for academic Grids, has previously only addressed membership issues and has so far ignored the Trust, Security and Contract management issues addressed in the TrustCoM framework. Consequently, these aspects of the Framework and the architecture are novel innovations. Supply chain management systems for large organisations do address contract development issues but do not address most of the security and automated contract and SLA monitoring and policy enforcement issues that are addressed by TrustCoM, so these remain innovations. Stand alone systems address role based security, but not the contractual context that TrustCoM does, so even the approach to this technology is innovative. The main innovations in the TrustCoM approach are in integrating these technologies to refine a contract and ancillary SLAs stated in business terms into deployable processes that monitor and enforce those agreements between organisations.

This document provides an overview of the results achieved so far in order to realise the TrustCoM framework.

The first version of the framework was produced as three individual documents (concept, architecture & framework specifications). This second version integrates all three issues in order to improve readability and reduce repetition.

The structure of this document follows the initial structure by first providing an insight into the underlying ideas and concepts that guide the development process. They also specify the requirements that define the basic structure of the framework. Section 1 of the document presents the business model and structure of the TrustCoM VOs. The following chapter depicts the architecture and the concepts of the individual components, i.e. how they contribute to the overall concept of TrustCoM as detailed in the preceding section. Since Virtual Organisations as envisaged by TrustCoM may be realised in many different ways (cf. section 3 for details), the following chapter will describe one potential instance of a TrustCoM VO and how the deployed components would interact to realise the functionalities and requirements in a specific use / business case. As such this chapter will already address some implementation issues as they are faced by Actionline 2 of the project. Finally, chapter 4 will go into further details regarding implementation, by

[1] The project is structured into Action Lines, Activities and Work Package which produce internal deliverables as well as externally available ones such as this. Consequently, references occur in this document to these internal project management entities which are meaningful to those within the project, but may not be to other readers. The authors and editor ask you to tolerate these references.

elaborating the profiles of the open specifications that are adapted to meet the TrustCoM requirements.

The document makes no claim for completeness of the TrustCoM framework as it is a ''living document'' and will undergo further changes as the work in this project proceeds and two new releases are made on a six monthly cycle until January 2007.

Most of the issues in this model have already been covered in more detail in the documents D16, D09 and D18 during the last elaboration cycle and we will refer to these documents rather than repeating all the information presented in them.

I The TrustCoM Conceptualisation

This section outlines the concepts used in the TrustCoM framework, these start from the vision that TrustCoM is trying to achieve, and consist of a set of models required to support that vision, and scenarios that put those models together during the operation of a virtual organisation.

I.1 The TrustCoM Vision

Given the economic competitiveness of a global economy, and the conflicting desire for a high quality of life in Europe, it was agreed by the heads of government at Lisbon in 2000 that Europe was entering a knowledge economy rather than one based on manufacturing or agriculture. In a knowledge economy, competitive advantage comes from the flexibility of organisations to respond to market opportunities. One mechanism to efficiently manage such flexibility is to automate the supply chain management for large organisations, or to provide environments to support the formation of Virtual Organisations (VO) of SME and large organisations which can recruit sufficient resources to take advantage of the opportunities where no organisation could alone.

Such an environment to support the formation and operation of VO has to both be trusted itself, and provide a basis for trusting other organisations with whom business could be done. Trust between VO members can be supported by each being transparently aware of the obligations and performance of others, so that business risks are both mitigated, and monitorable. The TrustCoM project pursues the goal of supporting the realisation of dynamic virtual organisations in a secure and contract managed environment. Thus TrustCoM envisages specific structures of collaboration between participants that actually form the basis for the framework as described in this document.

The current presentation differs from previous documents covering this issue insofar as the concepts have evolved during the project and have become both clearer and more detailed. Also, the underlying concepts are now being treated with a particular focus on the business and legal aspects of TrustCoM rather than just the technical ICT ones.

Business collaborations of the form envisaged by TrustCoM have significant impact on legal, business and technical resources of each participant in a VO. In particular, each participant needs to ensure the legal compliance of its interactions with other partners, the integrity of the business process within which it is involved, its reputation with regards to performance and service delivery, and the availability and confidentiality of its shared resources according to its agreement with the VO.

I.2 The Trust Model in the TrustCoM Framework CCLRC

Trust is an attitude of individual humans, and is applied only metaphorically to organisations or to objects; although often, more precisely, as a metonymy where the trust relationship holds between the individual senior managers of two organisations but is generalised to the whole organisations. Social scientists differentiate trust in a person from a judgement of competence in a person – the judgement that a person is competent to fulfil a role. The residual notion of trust is usually defined in terms of the commonality of intentions between two parties – that is, that another person is on your side. Trust becomes important on occasions when other supports to the relationship have broken down – for example when contracts are breached by error and the other party is willing to forego redress, or when unexpected circumstances occur in which a party is willing to make short term losses in order to maintain the relationship, in the uncertain expectation of receiving long term benefit. Across the population, individuals vary considerable in the variety, number and power of other supports to relationships. Consequently, they vary considerably in the ease and frequency with which they must rely upon trust alone to guide them. Within the computer mediated VO relationship, the TrustCoM framework is designed to provide both many supports that can maintain a relationship before relying on trust, and a basis for establishing trust itself before they fail.

A commonly attributed untrustworthiness in software is shown by web browsers that pop-up extra windows to advertise products that the user does not want, or which transmit personal data to another organisation against the user's wishes for privacy. In these cases the distrust of the software is generated by it appearing to act against the user's interests or intentions[2]. Obviously all notions of trust of a computer system are metaphoric, since it has no intentions itself, and the intentions of neither the creator nor the owner can be known, but only inferred from the behaviour of the system. Thus, for the software implementation of the TrustCoM framework the trust of the user in the software, and the organisation presented through the software will be both metaphoric and inferred from the behaviour of the system and the organisation operating through it. Since that has been stated, the term trust will be used from now on with reference to organisations and software without constantly noting its metaphorical nature.

Consequently, TrustCoM is developing a framework where the behaviour of the software and the organisation operating through it are explicitly constrained, are transparent, and provide a basis to predict future behaviour in order to foster trust in the user. In practice the mechanisms to constrain the behaviour are contracts and service level agreements (SLA) linked to collaborative business process models (BPM) which between them define what operations can be done, what access is permitted by whom and for what purpose, and what are the consequences of breaking the agreement – the operations and their context are clearly and securely defined. Transparency is provided by a publicly available agreement and BPM whose operation is implemented. Both transparency and the basis to predict future behaviour are provided by the monitoring of the performance of the BPM, recording the time and quality of performance according to the SLA, and drawing on this as a record to predict the competence of an organisation to fulfil its role[3]. The consequence of these mechanisms is that business risks are mitigated. Consequently, reliance can be placed on business partners because partners can be selected on the basis of a record of their past performance in a role, and each organisation will be informed as soon as they fail to be reliable, so that the risk can be managed.

---

[2] Social scientists also address the complex case of judgements of trust when an individual acts against another's short term interests and intentions, while acting for their longer term interests. However, such cases are too complex for consideration within the present project.

[3] "A contractor's past performance record is arguably the key indicator for predicting future performance." (US Department of Commerce and the Office of Federal Procurement Policy) but "a fund's past performance does not necessarily predict future results" (US Security and Exchange Commission).

Two further terminological points arise in this context. Firstly, the term used in computing research for recording historic performance information and using it to support decisions is reputation management. This term has unacceptable connotations in many fields, where such technologies are called “supplier qualification systems”, however the term reputation management will be used in this framework although the less worrying term can be substituted.

A second terminological confusion can arise from an idiomatic use of the term trust in computing to refer not to the trust concept as described above, but to refer to the method of transmitting trust. That is, a trust technology is one that transmits authority to trust the statements (tokens or certificates) of an issuer. Consequently, a trusted entity is one where authority to trust has been transmitted. This idiomatic restriction is perfectly consistent with the conceptualisation above although limited to avoid the complex issues of what trust is, or how it is brought about.

I.3 The Business Model of TrustCoM SAP

Virtual Organisations as envisaged by TrustCoM can be regarded as the coordinated collaboration between individual (legal) business entities that share a common goal – generally, we may claim that such a business goal is the business opportunity the swiftly formed virtual organisation seeks to exploit. Thus the required expertise required from business entities, their limitations and the general requirements are implicitly defined. Entities participating in such a VO all contribute in a defined way to this goal and need to pool resources in order to perform their respective tasks, i.e. the overall collaboration may be highly interactive. The collaboration during the VO’s operation phase involves the exchange of messages realising the aforementioned level of coordinated collaborative activities. Even though such VOs may provoke the impression to be static in realising the goal, the actual participants may constantly change their private configurations and even an entire entity may either be replaced or added and dispatched dynamically over time. While the former does not necessarily have an impact on configurations of the VO itself, the latter however does, requiring the ability that a VO is able to adapt to fundamental organisational changes. This allows for collaborations that are highly dynamic and in principle capable of adapting to changes in the midst of VO operation.

For TrustCoM, collaboration takes place between VO members which are, regarded just by themselves outside the VO context, otherwise independent legal entities. They exchange messages to connect separate business tasks contributing to the VO goal which are encapsulated by individual web service implementations. From a high-level, global point of view, a TrustCoM VO may thus be regarded as a coordinated interaction between individual web services (providers). This global collaboration perspective is called the collaboration definition or, better known in the web service world, the choreography of the VO.

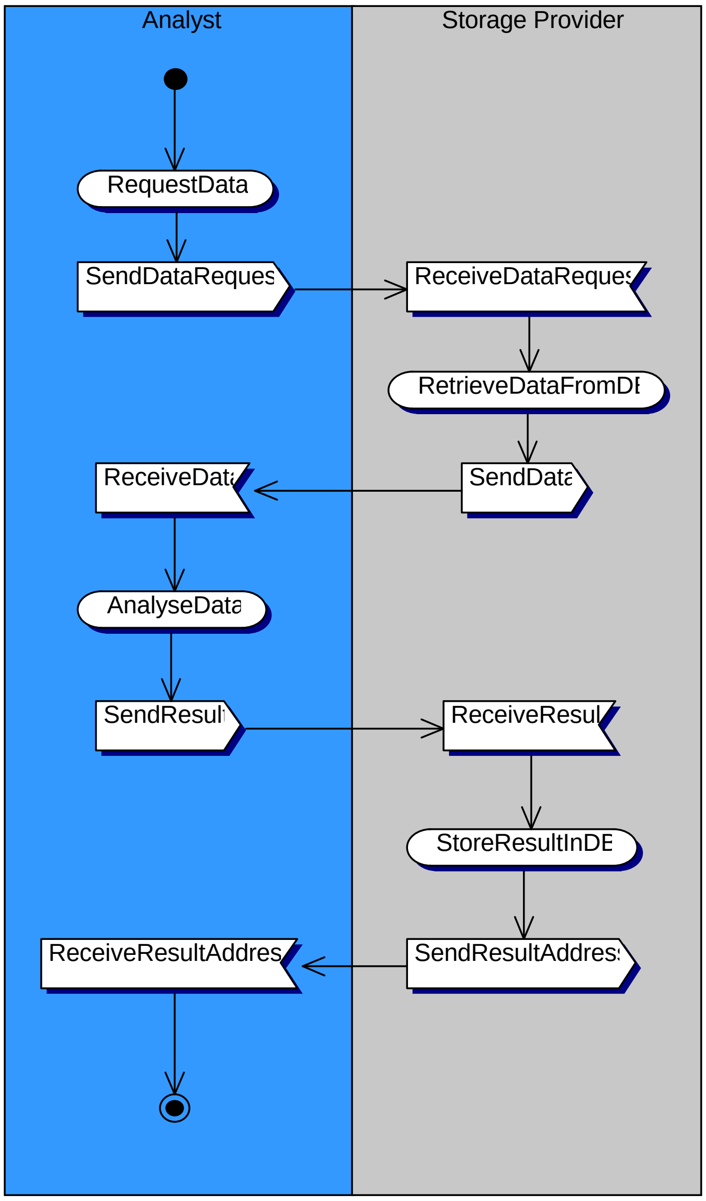


Figure 1:Sample collaboration definition as a UML activity diagram

I.3.a Business Processes

As shall be detailed in the following, we distinguish four "levels" of business processes according to their degree of abstraction, namely:

1. the goal description
2. the collaboration definition
3. the (individual) public business processes
4. the (individual) private business processes

The Goal Description

Every Virtual Organisation pursues a specific business goal as defined by a customer or VO initiator. Such a definition will generally be formulated abstractly without providing any details regarding how to realise this goal - in the case of the collaborative engineering (CE) testbed provided in WP35 to demonstrate the TrustCoM framework in operation, this goal may be formulated as "redesign and adaptation of an aeroplane regarding onboard entertainment". The CE testbed scenario is set in the aerospace industry and details a plane maintenance and upgrade scenario. The scenario subset discussed here comprises of only two participants, each playing enacting one business role. The first business role is the one of a design analyst, the second f a storage provider. Notably this definition does not even detail the goal but nonetheless - as will be described below - carries enough information to form a VO.

By making use of such abstract definitions, the virtual organisation will be allowed much more flexibility and dynamicity during execution whilst at the same time it releases any VO initiator from the requirement of having to know execution details: even though the initiator **may** specify a complete business process including all the details and requirements, we must generally assume that he or she lacks the respective expertise, thus also addressing the average customer as a potential initiator of a virtual organisation.

The Collaboration Definition

From the goal, an actual description of the high-level processes and the required actors may be derived. This generally requires the help of some kind of “business expert” who knows how to define a collaboration definition and has a good understanding of what tasks are involved in the respective goals. The main contribution of the “business expert” is his knowledge on how to divide the work needed to be done to achieve the VO goal. This division leads to a separation of activities to business roles for which actors have to be discovered. For TrustCoM it is of no particular interest for the concept **how** the definition is derived from the goal statement – without loss of generality we may assume that such an expert either provides his/her support either as a web service or feeds a public repository with sets of potential collaboration descriptions for various goals (see Appendix, I.2b for details).

The actual collaboration definition covers three main issues:

- a description of the involved actors, consisting of participants and their business roles
- the requirements and restrictions
- high-level activities
- the interaction between these actors

This way, a collaboration definition provides not only all the relevant information for reaching the goal by specifying the sequence of interactions and data-exchanges, but also provides the relevant information for actually identifying the required actors, i.e. their description. TrustCoM extends this concept by adding some means of deriving the requirements from the overall restrictions as provided by the initiator – this covers e.g. how to calculate budget-limitations for each party given the available budget or individual time constraints on basis of the overall deadline etc.

For instance Figure 1 depicts a simple sequence of interactions between two business roles, a design analyst and a storage provider within a collaborative engineering scenario where they are respectively analysing aircraft designs and providing the storage to hold the analyses. The analyst is billed by the storage provider for storage space needed for a plane's design data. The analyst performs analysis work on such data. To find an actor for the role of storage provider, the role of an analyst for instance imposes a budget restriction such as the storage space for the entire collaboration time period should not be more expensive than 3000€. A time restriction might be that the access time to the analysis data should not take longer than 3 seconds. The latter would then result to a bandwidth requirement for the storage provider.

For TrustCoM, such a collaboration definition is a fully valid "collaborative business process" comprising the global view of the entire set of participants and their business roles and in general the only process related description the VO has to take care of on its highest level, even though it is **not** a fully detailed description of all involved tasks, interactions and as such not on the same level of detailed modelling as an executable business process. The latter is a means to enact a certain business role in a collaboration definition and it is up to the actor how to conform to the required sequence of message exchanges and related requirements and restrictions. This specific sequence will be referred to in the following sections as the business protocol.

The (Individual) Public Business Processes

Actors enacting the "business roles" in a collaboration definition conform to the required business protocol to reach the VO goal. While keeping their assets such as internal services and optimised processes private, they are obliged to at least expose the communication endpoints for participating in the business protocol. Figure 2: Private and Public Business Processes depicts this sequence of message exchanges with the communication arrows between the "AnalysisPartner" and "StoragePartner" swimlane. This diagram follows the line of the previously introduced example collaboration definition by illustrating the actual private and public business processes.

Public Business Processes are the conceptual components which facilitate the controlled exposure of only those endpoints. Those are not executable business processes, rather the interface layer to the latter – keeping private processes protected inside the own domain while allowing for collaboration in the VO.

The (Individual) Private Business Processes

Actors in the collaboration definition are not identical to tasks in a business process and accordingly the actors are not the actual web services, as shall be described in more detail in section I.3.b. An actor is a business entity, e.g. an organisation, company or department, which aggregates services. In fact, an actor may not even publish all the tasks, web services and resources that are involved in performing a specific "business role" due to privacy issues. Even though a business entity is free to do so, TrustCoM supports the issues involved in **only exposing those assets in a controlled manner which are necessary to participate in a VO**. Without loss of generality we hence assume that the actions defined in the collaboration description do not map directly to the tasks that are actually performed by the actor, i.e. the service provider plays a "business *role*" in the collaboration that implies the enactment of individual *tasks* **internal** to that actor.

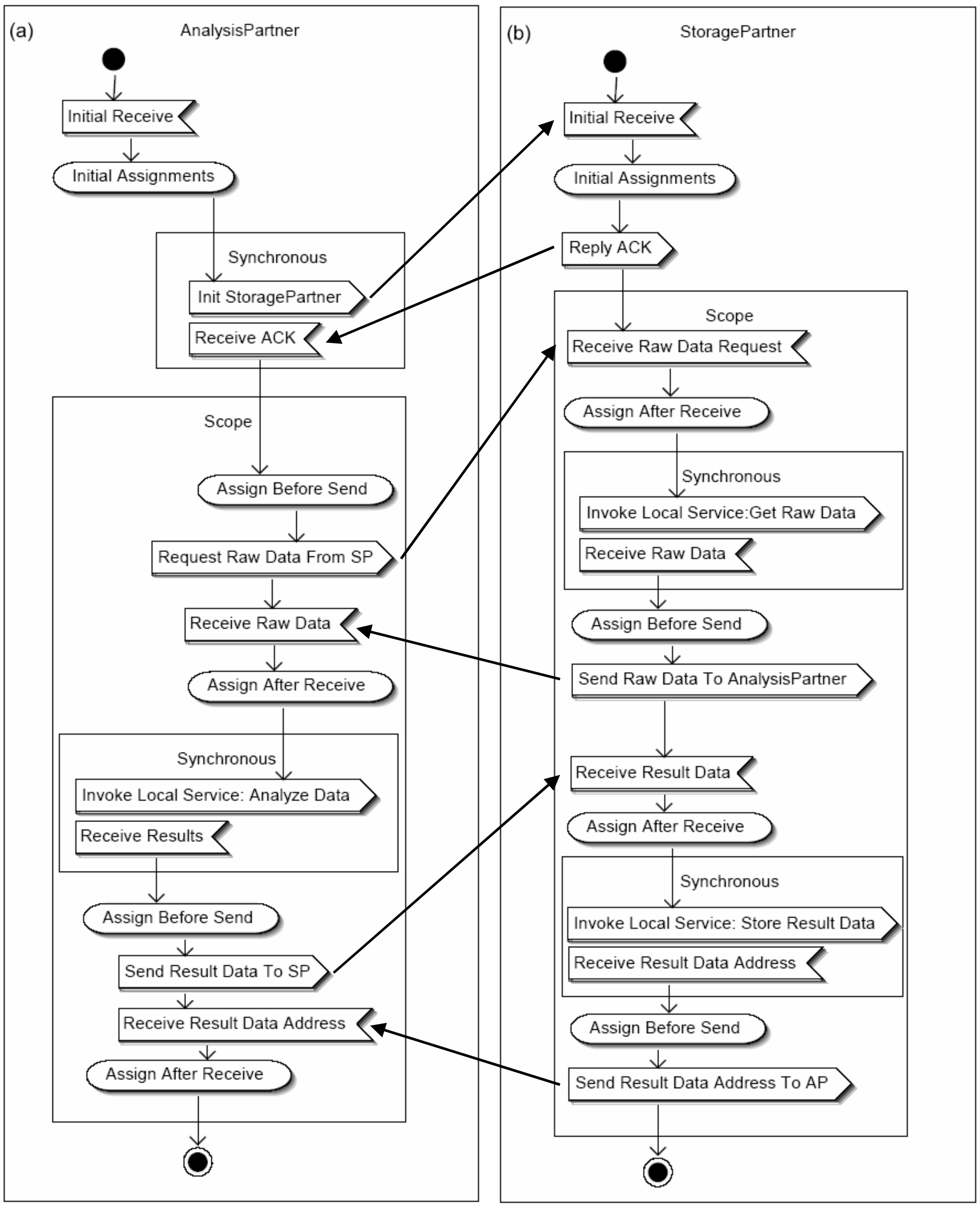


Figure 2: Private and Public Business Processes

To follow the above example, Figure 2 shows the private processes for each actor in a separate swimlane as a UML activity diagram. The “analysis partner” (business role), after agreement on an assignment, will actually have to analyse the existing design (set of activities in scope), e.g. analyse the requirements from the “entertainment designer” (business role, not shown here), based on the design data provided by the “StoragePartner” (business role). From the customer’s, as well as the other actors’ point of view, these details are of no importance however, and will unnecessarily complicate the overall collaboration description. Accordingly, this (individual) business process is internal to the business role and the execution details, e.g. a BPEL private process model, may be completely unknown outside the respective domain. Note however, that these business processes still have to comply with the overall requirements, e.g. if the customer explicitly stated that no subcontracting will be accepted, the business process may not involve any tasks that have to be performed by actors that are not part of the VO etc.

In summary, the TrustCoM VO principally undergoes three phases regarding its “business description”: (1) The customer or initiator provides a description of the goal that is to be achieved by the VO. (2) From this goal a collaboration definition is derived that specifies what kind of actors are required, which high-level activities they have to perform and how they interact etc. This information will support the identification of and negotiation with potential VO participants. (3) Each business entity that actually participates in the VO will “convert” the respective role descriptions and requirements into individual business processes that can be enacted it.

The business processes in (3) entail private and public processes. Since each business entity performs this step locally, the private knowledge of highly optimised business processes, e.g. efficiently retrieve and store design data, can be used for the private process. The public process needs to comply with the required interaction sequence and message types for participating with another business role, only exposing the required interface information while hiding the private process.

I.3.b Structure of Virtual Organisations

TrustCoM aims at realising Virtual Organisations on basis of web service interaction, thus allowing secure and coordinated interactions across enterprise boundaries. All actors in such a VO expose their functionalities through standard web service interfaces.

However, as has already been shown by such projects as GrASP, this does not imply that the actual processes can only be plain web services, but rather that all interaction between those are *exposed* as web services. This means that in principal anything that can or is linked to a computer can act as a participant in this VO model. This is of particular interest to TrustCoM as it has an impact on what we mean by “participant” and “(business) role” in a business collaboration:

Since collaborating partners are organisations who are more than just plain web services, we need to distinguish between the VO view on participants and their actual internal structure. The latter raises security implications regarding asset protection, privacy and data confidentiality as well as controlled exposure of the minimal required collaboration infrastructure. This relates to the distinction between collaboration definitions and individual business processes as detailed in the preceding section. Accordingly we need to clarify that even though we speak about (web) service providers interacting in a virtual organisation, it is really “business roles” that are realised / provided by the individual participants which again expose their functionalities as web services. From a VO-perspective there is no real difference however, whether the web service used for collaboration is just an interface to more complex executable processes or actually encompassing the business role’s entire behavioural interface for the virtual organisation.

To allow full integration into the VO lifecycle, in particular to enable autonomous discovery **according to the collaboration definition**, we must assume that information about the roles the individual entities can fulfil have been published in the enterprise network. As the functionalities are provided via web service interfaces, this process will be principally identical to the one for publishing web services.

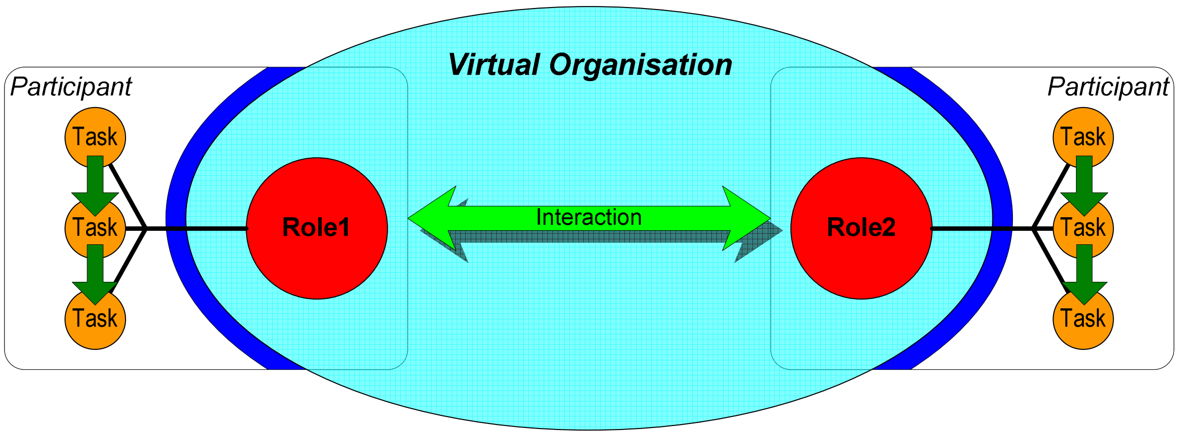


Figure 3: mapping between roles and actual tasks in the VO.

With this conceptual approach we ensure that any participant in the virtual organisation maintains full control over his/her resources and thus can enforce the respective privacy issues.

I.4 The Contract Model of TrustCoM CCLRC, NRCCL

Virtual Organisations as envisaged by TrustCoM can be regarded as the coordinated collaboration between business entities that share a common goal. From a legal perspective, the virtual organisation will normally not be considered as an organisation with legal personality, but as an instance of collaboration between the VO members. The key means to steer this collaboration is a contract or a set of contracts between the participating organisations. These contracts play a vital role in governing commercial interactions between organisations. Moreover, the contracts need to be closely linked to business processes in the e-business applications. This interplay between the legal level and the business process level is necessary in order to facilitate the joint approach towards the achievement of the common goal and to reduce inherent risks. This integration is facilitated through the TrustCoM concept of General VO Agreement (GVOA). The GVOA is a “container” of VO contracts, SLAs and policies that all partners agree to.

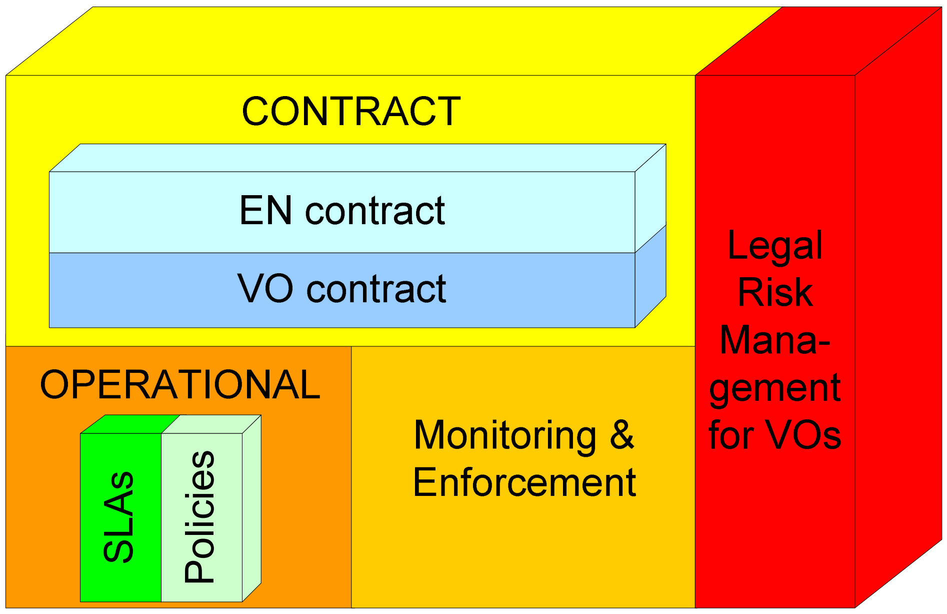


Figure 4: The Contract Model in TrustCoM

A contract is often defined as a legally binding agreement that creates an obligation to do or not to do a particular thing.[4] In the context of the TrustCoM project, our focus is on the internal legally binding agreements between VO participants.[5]

Whether agreements between (prospective) VO participants can be considered as contracts, depends on whether the parties regard them as legally binding and enforceable. A contract is usually formed by an offer and an acceptance; sources of law (national or international) provide detailed requirements on contract formation.[6]

TrustCoM has followed an approach in which two classes of VO contracts are defined:

- *VO Contracts*: contracts that express the general rules that each partner of a VO must abide to. These general rules for of collaboration constitute the legal basis for the collaboration. They define how the VO collaborates towards the achievement of the common goal and how the partners jointly work with reducing the risks of collaboration.
- *Service level agreement (SLA)*: contracts that express the specific rules that partners involved in a specific (operational) business process must abide to.

A VO contract identifies and specifies the general rules that characterise how operational business processes are to be conducted through collaboration in a VO. On the other hand an SLA describes Quality of Service (QoS) objectives for a specific service as agreed by the service provider and the service consumer.

[4] William P. Statsky, *West's legal thesaurus/dictionary,* West Publ., St. Paul 1986. One may want to add that the contracts also may contain permissions.

[5] There will also be contracts involving VO members and third parties, see e.g. Report on Consumer Protection and contracting with 3rd parties by the ALIVE IST project http://www.vive-ig.net/projects/alive/Documents/Consumer_Protection.zip.

[6] As an example, consider the United Nations Convention on Contracts for the International Sale of Goods (CISG) Article 14 I (1) and Article 18 I, even though they address goods and not services.

These contract types also need to be related to the different organisational levels of collaboration. The creation of VOs may be facilitated by an Enterprise Network (EN), which is set up as a basis for more specific collaboration in VOs. This EN will and should also be based on a contract which should include rules about the collaboration at EN level and about the creation of VOs. Hence, if there is a contract-based EN, both VO contract and SLAs may be understood within the context of the EN contract.

I.4.a EN Contract

The EN contract will be drafted by the EN founding members; it will be formulated in natural language.

A template for an EN contract is included in the Report Legal Issues in SME clusters, provided by the Legal-IST project (legal-ist.org). An EN contract should at least cover basic issues for collaboration, including

- EN structure
- EN governance (EN management structure, etc.)
- Outline of VOs (industry domain, VO management, etc.)
- IPR & confidentiality issues
- Data protection issues, if applicable
- Payment & Costs,
- Liability and insurance
- Jurisdiction & Choice of Law
- Dispute settlement
- Etc.

EN contracts will be defined by the EN, based on the types of VOs envisaged by the network, taking into account the specific needs of the industry in question and based on the requirements laid down the applicable national law. Though templates and model contracts are available, it is not possible to draft one general EN contract for all applications. There will be major differences between possible networks in various industries, services, jurisdictions, etc. The more similar the VOs in the network are, the more details may be included in the EN contract.

I.4.b VO Contracts

VO contracts may be written in natural language, but other formats may also be used. The content of VO contracts essentially depends on the specific kind of collaboration and on the relevant industry, (e.g. collaborative engineering in the aerospace industry or provisioning of eLearning services). A VO contract template in natural language was provided by ALIVE IST project (consortium agreement type of contract). More specific model contracts for different contexts are available in legal literature.[7]

Amongst the issues to be addressed by the VO contract are QoS requirements, access rights to computational resources, and trust issues (including consequences of one VO partner's trust level falling below threshold).

[7] See, e.g., Richard Morgan and Kit Burden, *Morgan and Burden on computer contracts,* 7th edition Sweet & Maxwell, London 2005.

A particular challenge in relation to VOs is the speed with which they may be expected to be formed, potentially on a time scale on the order of minutes or seconds. Creation and signing of VO contracts may thus need to be fully automatic.

The creation of VO contracts may be facilitated through the use of templates drafted e.g. at EN level. EN members from a certain industry (e.g. collaborative engineering) will normally have access to typical contract models utilized in their industry. Based on these typical contract models, VO contract templates can be drafted by the EN founding members. In cases where there are major differences between VO contracts in an EN, the EN may need to draft several different VO contract templates. Some of these templates may be very detailed, leaving only some specific matters (e.g. QoS requirements and price) for the actual contract negotiation. The degree to which the VO templates need to be adapted depends on how much the VO contracts differ from each other.

I.4.c Contracts and the VO lifecycle

The EN and VO contracts will also need to address the different phases of the VO lifecycle: Firstly, in the pre-contractual stage of the VO (identification and formation), there may be preliminary contracts (letter of intent, memorandum of understanding/preliminary contract) regulating the creation of the VO.[8] At the same time, the EN contract may include rules for the creation of VOs, e.g. regarding the selection of prospective partners, confidentiality duties, etc. Secondly, the operation as well as the evolution of the VO will follow the rules laid down in the EN and/or VO contract. Thirdly, the dissolution of the VO will need to follow the contractual rules, and VO contracts will typically include rules about the effects of termination of the VO contract.[9] A VO contract may e.g. include confidentiality duties which will prevail even after dissolution. Similarly, liability issues may need to be addressed after dissolution. Last but not least, if the VO is expected to generate results that may be IP protected, then the VO contract should address IP rights and use by VO members after dissolution. Hence, though the VO is dissolved, some contract provisions will remain valid and will require the attention of the VO partners. The contract should therefore be available for VO partners also after dissolution.

I.4.d Examples from the TrustCoM test bed scenarios

The TrustCoM test bed scenarios illustrate that there will be major differences between VO contracts in different industries: The eLearning scenario envisages that there will be a Metacampus EN contract, i.e. a rather stable contract for those participating in the marketplace. In this scenario, VO formation needs to happen in a matter of seconds or minutes as the user requests and then selects a learning path via the portal. Since the VOs only differ with respect to the learning paths, the involved LCPs and end-users, most legal issues may be covered in the EN contract. Content providers could, for example, join the EN and agree to be bound by the EN contract when registering their services in the eLearning EN. Nevertheless there is a need for a (rather operational) eLearning VO contract that governs the provision of eLearning services to one end-user, based on one learning path. Contract templates could be specified e.g. by the initial eLearning EN

[8] See, e.g. ALIVE IST Project *VE Model Contracts, Deliverable D 17a* (2002), Section 3.

[9] For further details see ibid, Section 4.6 on p. 27.

founder(s) and agreed to by each EN partner as they join the EN. This would need to be anchored in the EN contract.

In the CE scenario, VOs will differ markedly from each other: Therefore, the EN will either be a rather loose club of collaborators, or there will be a multiplicity of ENs, or the EN is centralized around the CE VO. Nevertheless, there will probably be more stable contractual relations

- between the CEVO and the Air VO, on the one hand,
- between the CE VO and a group of (potential) service providers, on the other hand.

The VO contracts between CE VO and other participants will differ based on the type of contract, e.g. outsourcing, ASP, consultancy, software licenses, combined contracts, etc. Model contracts and guidelines for the different contract types are available in legal literature.[10]

I.4.e Drafting EN and VO contracts

EN and VO contracts will be drafted based on an assessment of the planned collaboration at EN and VO level. This assessment should both cover positive aspects (what is the business objective of the EN/VO and how can it be achieved) and negative aspects (risks related to the collaboration, affecting either the common business goal or the assets of the participants).

The drafting of some elements of the EN or VO contract will be based on the business plan and strategy, on the specific needs of the industry in question and on specific requirements laid down the applicable national law. This positive assessment will take into account the envisaged VOs the VO lifecycle, the VO management structure and what in TrustCoM is referred to as the collaboration definition. The collaboration definition includes a description of the involved actors, specified as business roles, and restrictions on such actors. This information constitutes the input to define a VO contract.

Moreover, the EN or VO contract needs to take into account risks related to the collaboration. This aspect can be covered in a Legal Risk Analysis, which seeks to identify risks related to the collaboration, affecting either the common business goal or the assets of the participants. These risks may be identified and analysed in a structured way. This analysis results in a list of risks, which may be prioritized according to their likelihood and consequence value. This risk assessment serves as a basis for the drafting of rules in the EN or VO contract. Moreover, legal risk management serves as a bridge between the contract level and the operational technological level, including monitoring and enforcement. In particular, legal risk management may be utilized in order to

- Identify risks that need to be taken into account when drafting the EN or VO contract, including risks related to policies specified as described in the TrustCoM framework;
- Identify issues of high importance within the operational part of the GVOA;
- Identify risk areas that should be monitored and rules that need to be particularly enforced.

[10] Ibid.

I.5 Usage Scenarios / VO Lifecycle Phases CCLRC

Four phases of the lifecycle of a Virtual Organisation are distinguished: (1) Identification and Discovery, (2) Formation, (3) Operation & Evolution and (4) Dissolution & Termination. The main tasks to be performed during individual operations of the system for each phase are describe here as scenarios.

I.5.a Establishment of a Virtual Organisation (Identification and Formation)

The first stage on VO formation is the formation of an Enterprise Network (EN) to provide a pool of organizations who are willing to join virtual organizations. Organisations must register with an EN register which acts like a yellow pages telephone book – listing the organization and the services that they are willing to provide. One problem that is not well addressed within Trustcom yet is the motivation to join an EN in a competitive marketplace– why should an organization join one EN rather than another ? It is planned to take guidance from other IST projects that are investigating the issues of VO breeding environments in order to resolve this issue. The EN register and other EN and VO infrastructure services will be hosted by a provider. Business models are presented in other TrustCoM deliverables that show how the hosting of EN and VO services could be a profitable business in itself, probably as added value additions to basic ISP provision.

An organization which is registered as an EN member identifies a business opportunity and has the intention of creating a VO to meet it. They become the VO initiator, defining the goal of the VO, and try to discover the organizations required to make up the VO to achieve the business objective. The VO initiator will interact with a service provided by the hosts of the EN and VO infrastructure to guide him through the creation of a VO – the VO Management service, which is one of several VO services that will be introduced in the scenario. .

Given a specific business objective (provided by a customer or by the VO Management organisation itself) to be realized by a virtual organisation, the VO Management service triggers the derivation of a business processes according to a collaboration definition by contacting BP Enactment. The latter now queries (known) BP Template Repositories for collaboration definition that realise the given task. Such templates contain the next highest-level description of activities, information that is related to the roles involved in realising the processes (i.e. descriptions of the services that fulfil the individual tasks), coordination information (how the services have to interact) etc. which are passed to VO Management for partner selection. It is an open issue to be addressed by the evaluation of the TrustCoM demonstrators whether the level of description of the business objective, the collaboration definition are defined at an appropriate abstraction for the available definition of the market opportunity and the envisaged structure of the VO at this stage of the process. They may be either too abstract or too concrete, in either case the mismatch between the representations offered by the VO management services and the conceptualisation of the human VO initiator will increase the risks of the VO failing - even at this early stage,

VO Membership Manager is invoked with the collaboration definition's role related data containing information[11] about the structure of the service (operations, interface etc.), the

[11] Further information types may be added in the course of the project

quality it has to fulfil (SLA) and its trustworthiness[12]. This information is passed to the Discovery Service that for each role to be manned contacts a set of (known) repositories and returns a (sorted) list of potential organisations that meet these requirements.

Once the Membership Manager has received this list of potential participants, the SLA Negotiator on VO Management side is triggered to negotiate the actual terms with the Application Service Providers (starting with the most suitable ones), until all roles are manned. If negotiation fails to cast a specific role, i.e. if none of the respective organisations meet all requirements, the business process to be executed by that organisation and its requirements need re-evaluating.

As soon as all participants for the virtual organisation have been identified, VO Management triggers distribution of the relevant information to each of the VO members – this includes:

a) required authorisations to access other members,
b) interaction and coordination information, like what data to pass when between services
c) VO agreements and policies, as well as
d) other configuration data (contact information, notification topics etc.).

Once all participants have confirmed their configuration, the VO manager is ready to instantiate the VO and enact the overall collaboration definition.

I.5.b Normal operational work

With all VO members configured, BP Enactment starts the execution of the overall collaboration definition by triggering the first Application Service Provider(s) of the workflow and forwarding relevant execution data to it (like input values or location of data files).

Generally, the Application Service itself is responsible for triggering the execution of follow up tasks by forwarding its output data to the Application Service Provider(s) next in the overall collaboration definition (the relevant information, like which services to contact and which data to pass, has been provided during VO formation for each derived BP per role – cf. section I.5.a).

At *checkpoints* in the enacted business processes, the respective Application Service provides status information to BP Enactment, thus allowing monitoring of the overall enactment.

Execution proceeds until either failures occur (like contract breaches, destruction of services etc. – cf. sections I.5.d, I.5.e) or the business process is finalised, in which case dissolution of the VO is initialised (cf. section I.5.f).

I.5.c Dynamic addition of an organisation during operation

Not all Application Service Providers are necessarily identified during the formation phase of the virtual organisation, as some tasks may only be performed after a comparatively long

[12] Note that „trustworthiness“ as used in TrustCoM relates to „reputation“ of the respective service provider. Accordingly, services that have not yet gained such a reputation need particular treatment.

period of time and hence reservation of a service for that duration is unfeasible. Under such circumstances, the difficulty connected with the discovery process has to be considered, as some services are less common and/or are frequently occupied – assuming that the required service does exist at all.

Hence, such an approach is in particular sensible, if the required services are relatively common and only needed for short intervals.

The discovery process is either triggered directly by the need for a non-manned service arising or by a specific discovery-related activity in the collaboration definition. In the first case, the address of a non-existent service is requested from VO Management which in turn triggers the discovery process at BP Enactment, whilst in the second case the identification process reflects a separate task in the business process. Likewise, the latter case allows for discovering new services ahead of time, i.e. before they are actually needed, hence reducing potential delays in the overall execution.

Flexibility of discovery *during* the actual operation of the virtual organisation is limited as opposed to during the discovery and formation phase, since no changes in the parameters of other service providers can be accepted in order to achieve the overall goal.

Once an Application Service Provider has been identified, it is provided with the required configuration data, as described above (section I.5.a). All VO participants that need to interact with this new service are informed of the change, respectively of the addition of a new participant, by providing the contact details (including access authorisation), i.e. Endpoint Reference Address to them. This also applies to Trusted Third Parties services insofar as they interact with the Application Services (e.g. Message Brokering, cf. section II.3.b).

I.5.d Dynamic removal of an organisation during operation

Similar to adding a service provider during the operational phase, an organisation may want to free their resources again, once they are no longer needed by the VO. Accordingly, the Application Service Provider has to be removed from the virtual organisation, if so requested.

Again, the request is either raised directly (in this case by the Application Service Provider him-/herself) or indirectly by the respective entry in the overall business process. In either case, the message is forwarded to VO Management, which triggers re-configuration as follows:

As the service provider has no further rights to access other services and should not do so for security reasons, all respective access rights are revoked. In order to avoid further communication and in particular forwarding of (possibly sensitive) notifications, all references to the respective service are removed – the only communication remaining takes place between VO Management Services and the Application Service Provider.

If a price for service usage was agreed upon, billing takes place at this time – the Log may serve as a means for establishing the actual price. Similarly, the trustworthiness of the service provider is updated on basis of its performance (as maintained in the Log), i.e. the reputation gained through participation in this virtual organisation is forwarded to the Trust Maintenance Service.

Finally, the service provider is removed from the list of VO participants at VO Management side.

I.5.e Replacement of a participant by another during operation of the VO (Evolution)

During the operational phase of the VO, a particular service may need replacing, due to non-performance, contract breaching, simple "disappearance" of the service or similar reasons. In either of these cases, the overall business process is delayed as the current task can not be executed. Since the need for substitution generally arises after the actual task started execution, replacement may even cause a rollback and compensation operation in the involved BPs, as a set of tasks will (in most cases) have to start anew with the new service.

Typically, a Policy Subsystem identifies the need for a replacement as a reaction to a specific event, like e.g. contract breaching and notifies VO Management. The latter may verify the correctness of data by directly requesting information from the respective Application Service Provider.

VO Management then triggers re-configuration of the VO as described in section I.5.d (insofar as the service is still available for contacting), i.e. it removes the service to be replaced from the virtual organisation.

At the same time, VO Management triggers the Discovery service to identify a new service provider that fulfils the criteria as defined for the one to be replaced. The Application Service Provider will then be provided with the necessary information as during the dynamic addition of an organisation (cf. section I.5.c).

Once set-up accordingly, i.e. the old service removed (all tokens and related information revoked) and a new service configured according to the VO's needs, BP Enactment triggers execution of the Application Service. Since input data may have been lost during the replacement process, BP Enactment furthermore triggers the Application Service Providers representing the preceding tasks in the overall business to re-distribute their data to this new service.

Note that, similar to dynamic addition of organisations (cf. section I.5.c), circumstances like relevance of that service for the overall execution, availability of replacements etc. play a significant role in whether a service should be replaced. Data like amount of service providers initially identified for that role (during the discovery phase) may be crucial for further proceeding.

I.5.f Dissolution of the Virtual Organisation

Once the overall business process has executed its final task[13], respectively destruction of the VO is triggered by VO Management (e.g. due to grave failure), the virtual organisation may be dissolved, i.e. all partners are removed from it as described in section I.5.d.

Once all Application Service Providers have been detached, the configuration of the VO Management Services will be reset up to the point of reuse. This means that the VO

[13] Note that a virtual organisation may be maintained for more than one execution of the business process and that not all tasks are necessarily orchestrated by BPs..

Manager may decide to maintain e.g. the collaboration definition for later execution and keep a list of all service providers that performed well so that they may be contacted again.

Generally, however, we will consider the virtual organisation to be reset completely at this point, i.e. any new request to a VO Management service provider will have to start a complete new setup procedure as described in section I.5.a.

II The TrustCoM Architecture CCLRC

A system as complex as the one aimed at by the TrustCoM project may not simply be depicted within one overview diagram: besides for the mere size, the system needs to meet high expectations with respect to dynamicity (covering all four lifecycle phases [16]) and flexibility (each service provider may adapt the framework according to its own requirements and Virtual Organisations will need to be structured to meet the respective business demands) – accordingly no simple diagram structure will suffice to cover all these aspects in a way that is comprehensible for people outside of the project. We hence decided on segmenting the architecture into (1) the more conceptual overview that provides an insight into all *potential* components of TrustCoM (the “relationship model”, this chapter) and (2) a concrete usage example of this architecture in a hypothetical business case (the “deployment model” chapter III).

In the following we shall elaborate what we understand as the “relationship model” of TrustCoM:

In order to meet the business requirements identified and partially repeated in chapter I above, every service provider will need to extend his/her services with specific functionalities in a way that realises uniform message so as to maximise interoperability. In addition to this, management mechanisms need to be provided that enable the coordinated interaction of these service providers to realise the collaboration of a virtual organisation.

Since each service provider will have a different infrastructure that needs to be extended and since every virtual organisation will differ in its setup and its requirements with respect to what mechanisms need to be supported, there can be no uniform architecture - in the sense of component structure - representing the full framework capabilities. Instead we shall try to cover the **potential** of the architecture by depicting all components and interactions that are principally **possible**. Note that this implies that not all the components depicted below will be required in order to realise the TrustCoM capabilities - rather this strongly follows the service oriented architecture approach and thus each component may be regarded as an optional enhancement that may be replaced/supplemented by principally any other component that provides the same functionalities according to the relevant standards (cf. also chapter III) or even completely omitted if the respective functionality is considered unnecessary.

With such an approach, the links between the components as depicted in the diagrams below can only be regarded as **relationships**, rather than actual interactions since a replacement (if any) **need** not support the full message exchange. The details with respect to which components and transactions need to be provided in order to realise a specific functionality will be described in more detail in the “deployment model” (chapter III).

Consequently, the components of the diagrams may not be considered actual instances of services, as some of them may even be completely omitted in an actual operational VO. Instead, the components represent *classes* of components that may instantiated into objects, respectively services – their relationships may thus be regarded as a representation of their interfaces (i.e. the functionalities they expose) as well as which services *typically* would invoke these. This implies that each type of class is represented only once in the diagrams and thus that the distinction between service types that make use of these components (cf. section II.1.a) is not visible, since the components may be deployed on different types, according to usage.

II.1 From Concepts to Architecture CCLRC

In summary, the TrustCoM framework has to respect that:

(a) a service provider may not want to support all functionalities necessary to maintain a virtual organisation by itself, and/or may want to make use of already existing functionalities rather than using the TrustCoM ones. Service and functionalities hence need to be separated in a way that avoids adaptations of the respective domains as much as possible – *ideally*, a service provider will just choose the components on a “plug & play” basis, where it is completely up to him/her to “outsource” the functionality, choose own components or even skip it completely.

(b) the internal structure of a service provider is private, meaning that the TrustCoM framework will not modify or expose it. The service provider alone should decide what information is available and how it is made available – this addresses in particular monitoring related issues (as will be discussed in the SLA related sections) and the “internal” business processes (cf. section I.3).

(c) service providers in a Virtual Organisation generally do not want to *depend* on other participants, in the sense of that the respective party may manipulate them or that they would be indirectly responsible in case of failure of the other participant.

(d) accordingly, it needs to be ensures that performance is supervised so that corrective measurements can be taken in time, thus avoiding not only failure of the overall VO, but also – from the individual participants’ point of view – that responsibility for the partners that need to interact with each other does not rest on the individual. Since the participants may never formerly have worked with each other such responsibility will generally be refused.

(e) along the same line, sensitive data needs to be well protected and access to the local resources restricted to those instances that require this access.

(f) the rules and policies of the Virtual Organisation and of each individual participant need to be enforced within the VO, so as to avoid failure, data misuse, security issues etc.

(g) for each individual participant his, respectively her own policies have the utmost priority and may by no means be overridden by the VO, even if this implies that the respective provider can not participate in the collaboration.

(h) where possible, some indication of the interacting parties’ reliability should be provided.

(i) interaction takes place not only across individual organisational borders, but also between different nations, thus implying different legal situations, corporate policies and cultural backgrounds

In the following sections we want to outline how the architecture developed by the TrustCoM consortium caters for these aspects with respect to the individual lifecycle phases of a Virtual Organisation.

II.1.a Abstract Structure HLRS

TrustCoM introduces a high-level VO structure that tries to accommodate for these issues by distinguishing services according to the *type* of functionality they provide to the Virtual Organisation (cf. Figure 5). Note that this distinction has already been thoroughly discussed in previous documents (D09, ID1.1.3 etc.).

- Application Service Providers (ASP)

  Any entity that directly contributes to realising the VOs overall business objective as codified in the collaboration definition by fulfilling one or more roles in it, is acting as an Application Service Provider. These entities are Business Partners that are obliged to provide the respective services by contract and that demand payment for their contribution.

  Application Service Providers are the main participants of a VO.

- VO Management Services

  These services provide the functionalities to coordinate the interactions between the VO services so as to reach a common goal, represented as a collaboration description. Though the VO itself does not rely on the existence of a central management instance, certain requirements imply some sort of central coordination.

  The VO Manager (as a specific instance of these services) furthermore can act as the interface between customer(s) and participants of the VO – thus it represents the customer's interests inside the VO.

- Trusted Third Parties (TTP)

  Though Application Service Providers and VO Management Service Providers form the main actors in a virtual organisation and in fact would suffice for enacting collaborative workflows, additional types of services are recommended to supplement the TrustCoM specific functionalities. This relates in particular to functionalities that can not or should not be realised by the Application Service Providers – either due to the ASP wanting to "outsource" the functionality, in order to maintain privacy issues or to reduce dependencies between ASPs:

  - function outsourcing

    though generally not recommendable, an ASP may leave certain management related functionalities, including the enactment of local policies, monitoring performance and similar issues (cf. below), up to third parties that he/she trusts to perform the respective tasks. This may also involve functionalities that the ASP would like to make use of, but can not realise him-/herself, like logging etc.

  - privacy enactment

    some participants may furthermore want to remain incognito for respective collaboration parties and as such may request means of brokering interactions through a third party trusted by him/her

  - "neutral" parties

    finally, ASP and/or VO Initiator may not trust the respective other to perform certain tasks neutrally, i.e. without cheating if profit for oneself may be gained from this – this applies in particular to supervision of performance and its relationship to payment. Shifting such responsibilities to third parties trusted by both ASP and VO Initiator equally will ensure that e.g. maintenance of the performance log is performed neutrally, without preferring the one result over the other.

Likewise, TTP services maintain data that may be confidential, making them subject to VO policies, agreements and in particular security issues. Accordingly, we consider TTP services as participants of the Virtual Organisation, since their behaviour is influenced by the requirements of the individual VOs.

Notably, all TTP functionalities *may* be enacted completely by either VO Management or Application Services, thus rendering the structure of TTPs very individual to each VO and not a general requirement, but a recommendation. In summary, Trusted Third Parties take an intermediary position between VO Management as a representative of the customer and the individual Application Service Providers as the main VO participants.

- Supporting Services

As opposed to Trusted Third Parties, there will be services involved in realising the VO's capabilities that in themselves do not directly participate in the VO, i.e. that provide the same unaltered functionalities to each customer, respectively VO. As opposed to this, Supporting Services do not directly participate in the VO, yet indirectly contributes to its functioning. In general, this concerns repository-like facilities that maintain lists of services (as potential VO participants). These types of services principally already exist (e.g. UDDI) and are not, respectively only minimally influenced by a virtual organisation.

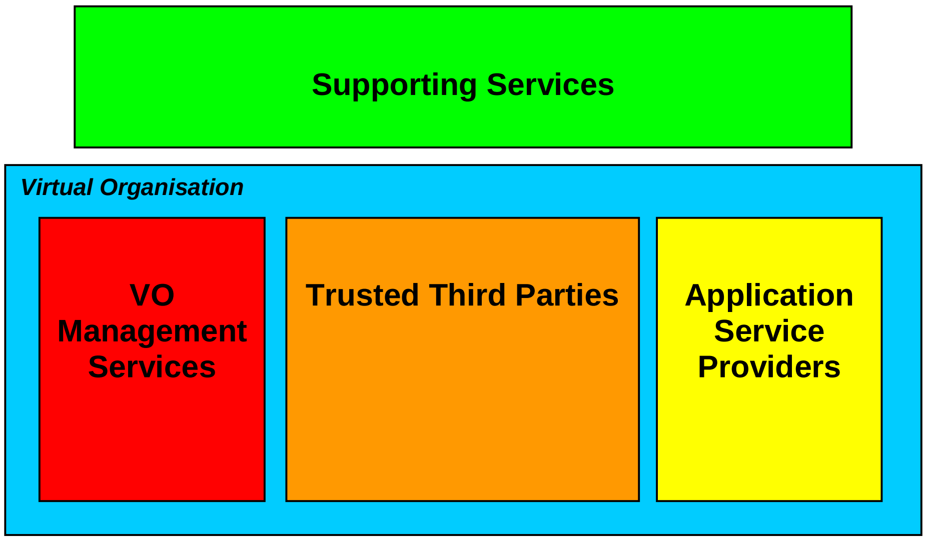


Figure 5: The service types participating in a Virtual Organisation.

According to this structure, Application Service Providers do not have to rely on other ASPs, but rather the management and sensitive functionalities are shifted to instances principally neutral to the individual participants. There is no restriction to all these functionalities actually being provided by ASP participants, i.e. any TTP or VO Management Service itself may be an ASP and the other way round.

One of the direct implications from this categorisation is that different degrees of adaptation requirements exist for the individual types of services: whilst generally an application service provider would want to make use of the full trust, contract and security features, trusted third parties will not necessary require full support, but mostly security control. Supporting services on the other hand are generally not impacted by these functionalities and will not integrate any of these features. This also applies to the communication layer – see section II.3 for more information.

II.1.b The Subsystem Segmentation

From the requirements and concepts as detailed in the previous sections, we can derive a set of functionalities that need to be provided by the TrustCoM framework that may be categorized as described in this paragraph and that reflect the individual expertise of the consortium. This categorisation is pursued throughout the project, so that individual functionalities, provided as services or components, are realised as part of the respective category, in the following also called "subsystem" of the TrustCoM framework.

With the Service Oriented Architecture approach pursued by the project, the individual components developed within the scope of the respective subsystems are principally usable in a stand-alone manner, as detailed in the following chapters. Since each subsystem reflects specific types of functionalities and hence requirements, a service provider may subsequently select individual components from each category, according to his/her needs.

Note that each of these subsystems is described in more detail in the appendix to this document.

VO Management

The VO Management component defines and stores details of each virtual organisation participating in the Virtual Organisation. It is divided into three main modules responsible for (a) lifecycle changes to the VO ("VO Lifecycle Management"), (b) maintaining the participants in the VO ("VO membership management") and (c) managing the General VO Agreement ("GVOA management"). These modules interact mainly with the SLA management component which creates and manages the detailed SLAs, and with the Business Process Enactment and Orchestration component which defines the business process of the VO, and enacts when the VO is in operation. The VO management component builds upon the data in the Enterprise Network Infrastructure when a VO manager wishes to create a VO in order to allow the business process of the VO to be defined in the BP manager along with the roles of potential organisations, it calls the EN Discovery Tool to discover candidate partners from within the Enterprise Network, calls the SLA Negotiator to negotiate with a candidate partner the details of the SLA to perform a role in the VO, then composes the legal General VO Agreement to be signed by all partners. Once the VO is created, the VO manager calls the BP Manager to enact the business processes of the VO. While the VO is in operation, the VO manager responds to evolutionary changes in the VO, as partners succeed or fail to meet deadlines, quality and other policies from the SLA, ultimately identifying replacement partners and renewing the GVOA. When the VO terminates the VO manager closes the VO down.

Business Process Enactment and Orchestration

The subsystem catering for Business Process (BP) Enactment and Orchestration provides generic, flexible services to be used in different application scenarios as well as for VO Management related purposes. This subsystem provides autonomous functionalities implementing the three phased Collaborative Business Process modelling methodology which was defined in WP2/21.

BP enactment begins with the global view/choreography of the VO business objective, the process and the roles required to achieve a set of goals, encoded in the collaboration definition. The CDL++2BPEL service component takes the collaboration definition as input and following a top-down approach, derives process views and optionally private processes from it. The latter occurs if no pre-existing private processes have to be taken into account.

A BP Management service offers runtime management methods for the BP engine. This service allows for automatic deployment of derived BPs and views, as well as their execution. Associated with engine comes a monitoring component.

On top of the operational aspects of BP creation and execution, this subsystem also takes care of Trust, Security and Contract (TSC) Management controls for BPs. Such aspects may be assigned at design and runtime as TSC extension roles to design time artefacts called TSC tasks in BPs (see Appendix, section I.2 for details).

SLA Management Services

The SLA Management subsystem provides a set of services that allow autonomous observation of individual service providers' performance and comparing these to a set of previously agreed upon quality of service parameters.

Accordingly, the subsystem needs to provide the functionalities to

a) negotiate SLA terms that meet both the customer's expectation with respect to the quality of service, as well as the service provider's capability (and intention) to maintain these.

b) monitor the performance of a specific service and/or its respective environment (like the host system's status)

c) compare the monitored information with the terms agreed upon during negotiation of the SLA.

A member of the Enterprise Network uses this subsystem to associate SLA templates with the services it may provide to an eventual VO. A potential consumer of the application service uses this subsystem to negotiate and sign an SLA with the service provider. The SLA Management subsystem assists VO Management (via the Discovery Service) in the search for services that can meet the QoS requirements of the VO.

Upon SLA violations, the SLA subsystem generates notifications that can be picked up by the Policy Subsystem in order to apply the proper adaptation policies.

Trust & Security Services

The subsystem for trust & security services provides services related to the establishment and maintenance of trust relationships with a priori unknown partners from foreign security domains.

Establishment of trust relationships is provided by Security Token Services that can issue and validate security tokens across administrative domains, and corresponding configuration management services that can be used to adapt the local security configuration to dynamic changes in the VO.

Maintenance of trust relationships is provided by a reputation management services that collects individual ratings about the prior behaviour (reputation) of Enterprise Network members, offers a combined reputation value to interested clients, and notifies registered VO members about sudden changes in this value due to recent activities. Also, a Secure Audit service provides the functionality to record custom data, for example actions performed by other partners, so that it cannot be repudiated.

Policy Control

The policy subsystem provides the means to define, deploy and enforce both access control and adaptation policies within the TrustCoM framework. Access control policies comprise both authorisation policies that define which entities are permitted to access services within the TrustCoM framework and under which constraints, and delegation policies which specify permissions on the delegation of administrative permissions. Adaptation policies (traditionally sometimes called obligation policies) are in the form of event-condition-action rules that define how the VO should adapt in response to failures, changes in the reputation or performance of its participants, security threats etc. For example, policies would typically dictate under which conditions the procedures for the removal of a member of the VO should be executed in case of repeated VO breaches or significant loss of reputation. Similarly, policies can be used to trigger reconfiguration of the service message interceptors in order to add additional handling procedures such as secure auditing. Policy control is based around two services: the policy service which receives policies from the GVOA, deploys access control policies to the Policy Decision Point (PDP), enforces adaptation policies and manages the policy life-cycle and the policy decision point which enforces the authorisation and delegation policies and responds to access control queries issued by the Policy Enforcement point which is part of the EN/VO infrastructure (cf. Appendix).

EN/VO Infrastructure

Each service provider has his/her own approach to making the functionalities of the offered service(s) available, to managing them and to support trust, security and contract managing features – if any. The EN/VO Infrastructure components provide the base functionalities to allow common access and management functionalities across all participants in a virtual organisation. This involves in particular:

a) establishing a communication layer that allows messaging and notification, and relates the additional TrustCoM functionalities (trust, security and contract) to the respective services

b) maintaining the actual locations of services and mapping handlers to them so that services are accessible even when moved

c) supporting coordinated instantiation of involved services

d) exposing functionalities of services for discovery and

e) supporting discovery over a range of service-related information (WSDL, SLA etc.)

Thus the EN/VO Infrastructure may be regarded as providing TrustCoM’s base layer.

II.2 The Relationships with respect to the individual VO lifecycle phases

In Figure 6 we provide a high-level overview over the relationships between the individual subsystems (cf. section II.1.b). As can be seen, there is a strong coupling between the subsystems with a very VO Management focused dependency – this reflects TrustCoM's approach that in order to realize secure and reliable virtual organizations, some central management instance is recommendable.

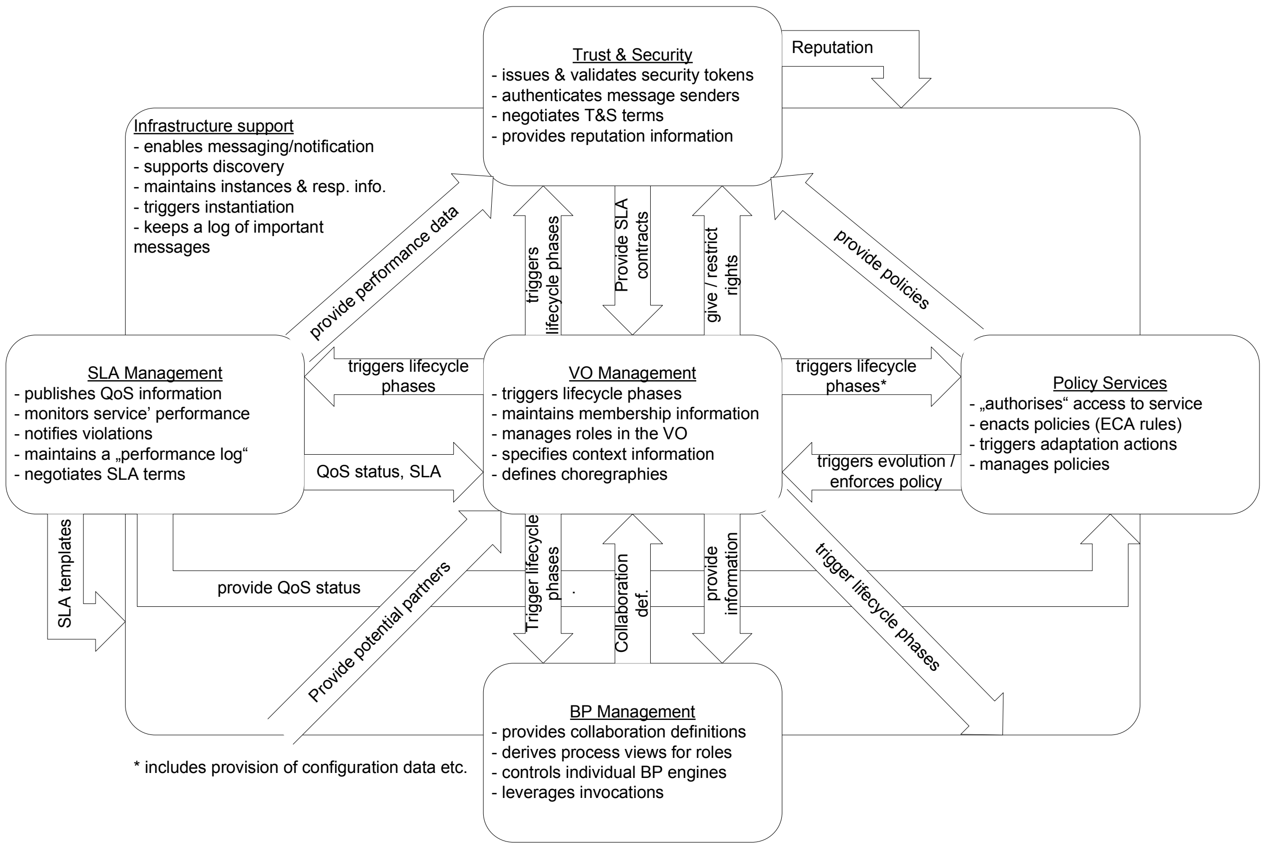

Trust & Security
- issues & validates security tokens
- authenticates message senders
- negotiates T&S terms
- provides reputation information
Reputation
Infrastructure support
- enables messaging/notification
- supports discovery
- maintains instances & resp. info.
- triggers instantiation
- keeps a log of important messages
provide performance data
triggers lifecycle phases
Provide SLA contracts
give / restrict rights
provide policies
SLA Management
- publishes QoS information
- monitors service' performance
- notifies violations
- maintains a „performance log"
- negotiates SLA terms
triggers lifecycle phases
VO Management
- triggers lifecycle phases
- maintains membership information
- manages roles in the VO
- specifies context information
- defines choregraphies
triggers lifecycle phases*
Policy Services
- „authorises" access to service
- enacts policies (ECA rules)
- triggers adaptation actions
- manages policies
QoS status, SLA
triggers evolution / enforces policy
SLA templates
provide QoS status
Provide potential partners
Trigger lifecycle phases
Collaboration def.
provide information
trigger lifecycle phases
BP Management
- provides collaboration definitions
- derives process views for roles
- controls individual BP engines
- leverages invocations
* includes provision of configuration data etc.

Figure 6: Overview over all subsystems and their interactions.

Note that the figure does not provide any details regarding the relationships of the individual components to each other: since there are many such relationships possible, the diagrams were separated according to the main lifecycle phases, respectively relating to the main VO scenarios as depicted in section I.4, so as to allow for a better overview. Accordingly, the relationships as represented in the following diagrams are restricted to lifecycle phase specific issues.

Since the EN/VO Infrastructure components offers common functionalities equally to all services in a virtual organization and partakes in most interactions, representing these relationships in the individual diagrams would just make these more confusing. Thus the relationship diagrams with respect to this subsystem were moved to its own section.

The general lifecycle model supporting VO is that organisations join an Enterprise Network with minimum commitment and expense – this phase is refered to as Preparation. An Enterprise Network member may wish to initiate a VO to respond to a market opportunity. The VO then proceeds through the following stages of the lifecycle:

1. Identification of VO members
2. Formation of the VO
3. Operation of the VO
4. Evolution of the VO
5. Dissolution of the VO

Phases 3 and 4 take place in parallel as the VO operates and evolves during that operation (also see section I.5).

II.2.a Preparation

Preparation for participation in a virtual organization is not really a lifecycle phase in itself; rather it reflects the necessary steps to take in order to participate in virtual organizations as envisaged by TrustCoM. Such steps involve mostly registration and publication processes in order to make the provided services / resources known and hence accessible. Thereby it is of no direct implication for TrustCoM whether these repositories are within Enterprise Networks, i.e. where additional requirements have to be met by the services in order to get registered, or whether these are publicly accessible, like the UDDI repository by IBM[14] and SAP[15].

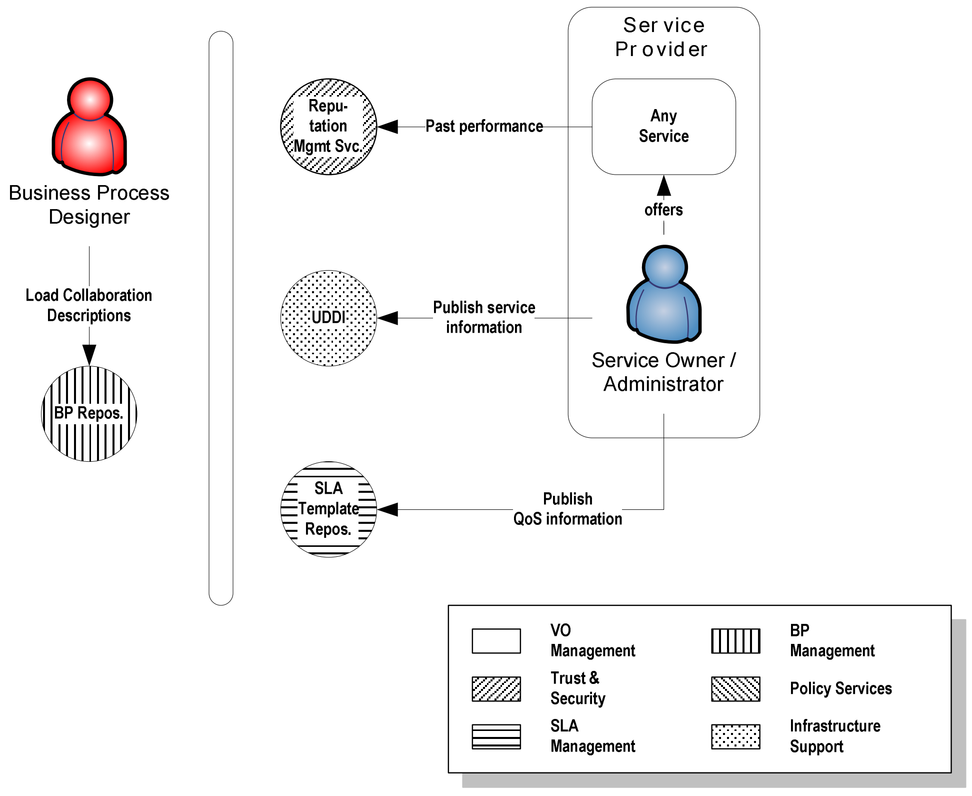


Figure 7: Components involved in the preparational processes.

Since every Enterprise Network will define its own registration conditions and steps in addition to the "typical" processes, Figure 7 depicts only the most relevant, respectively

[14] Currently discontinued, see http://www-306.ibm.com/software/solutions/webservices/uddi/

[15] http://udditest.sap.com/webdynpro/dispatcher/sap.com/tc~uddi~webui~wdp/UDDIWebUI

most recommended publication components, namely we assume that information about the service (be it an abstract application service, cf. section I.3, or supporting services, cf. section II.1.a) is published in a Service Description repository (e.g. UDDI) and that additional information regarding the quality of service it may support is available through an SLA template repository.

In addition it may be assumed that the service is registered at a reputation management service and has already achieved a reputation through past performances (cf. Appendix and section I.2) – though such information is helpful for identifying services on a trust-reputation basis (cf. below), this can not be considered a requirement as in particular SMEs and their services new to the eBusiness domain will not have earned a reputation and would hence be "invisible" to a Virtual Organisation that requires such information.

Without loss of generality, we furthermore assume that a business process designer has stored typical collaboration descriptions (cf. section I.3) in an accessible business process repository that allows users to get collaboration descriptions for a specific business goal. Notably, this process can be easily replaced with addressing such a designer directly, the customer providing the description him-/herself or similar solutions (see below).

II.2.b Identification

The identification phase is generally considered to be the first lifecycle phase of a virtual organization: on basis of the abstract goal description as provided by the VO initiator, service providers need to be identified that fulfill the requirements of the collaboration definition CD (derived from the goal description) and the additional requirements as defined by the initiator. Besides for the "application service providers" as defined by the CD, the VO will need some supporting services, as well as trusted third parties to fulfill the tasks.

Notably this process may in itself be very dynamic, since each application service provider may in itself alter the overall VO requirements, either by not fulfilling the task(s) exactly in the way proposed by the CD, by requiring outsourced or subcontracted support e.g. by additional TTP services (cf. section I.3) or related issues. Similarly the actual details of the individual participants will implicitly influence each other, as e.g. the time and budget constraints of the overall process need to be shared by, respectively distributed to all participants (cf. negotiation below).

We consider the negotiation of contracts to be part of the Identification phase, since it does not *necessarily* lead to a contract, as it may fail – in such a case, an alternative service provider needs to be identified and would hence lead to an alteration of Identification and Formation processes. Rather we follow the approach that the result of the Identification phase consists in a list of service providers that will and can definitely be considered participants of the VO once they have undergone the Formation procedures.

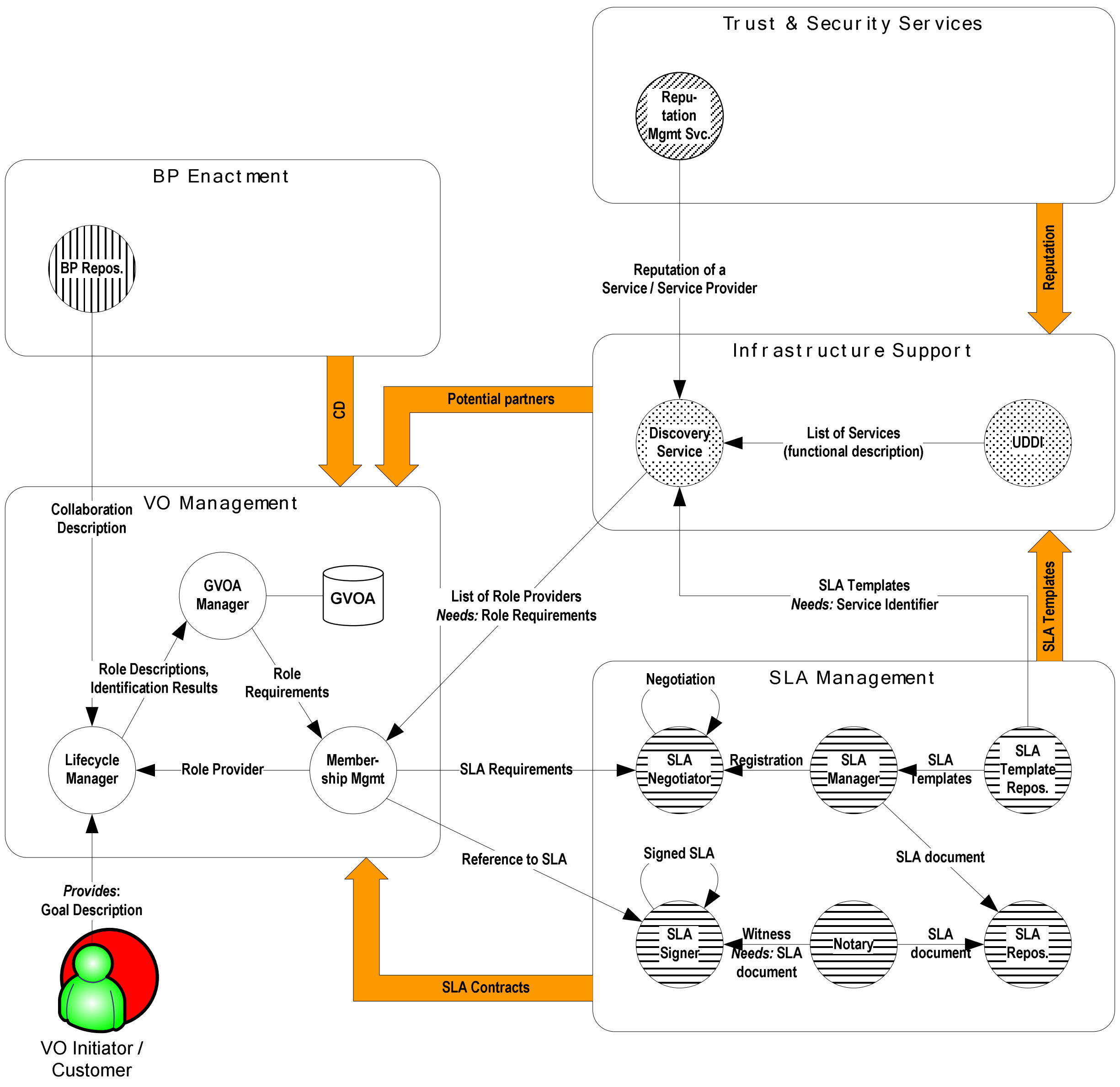


Figure 8: Relationships between components during Identification.

In Figure 8 all the components involved in realizing the Identification related processes, as well as what functionalities they provide to which other component are represented:

Without loss of generality, we assume that the instantiation of a Virtual Organisation is initiated by a customer with a particular business request ("goal description") that he/she passes to a VO Management Service provider. The VO Management related functionalities may be provided by the Customer him-/herself, if he/she desires to do so, as is e.g. the case with the CE testbed scenario. The main functionalities of VO Management in this phase relate to generating a GVOA from the collaboration description that is provided by the Business Process repository – this includes finding providers for the roles as defined in the CD and negotiating individual contracts, as argued above.

We introduce a discovery service functionality that queries a set of more or less public registries and repositories – again these could be restricted to a specific Enterprise Network – to identify the service(s) that fulfill the requirements with respect to (a) functionality, (b) quality of service and (c) reputation. The latter only if the service, respectively its provider has already built up a reputation, as noted in the previous section. If no reputation data exist, it is up to VO Management and/or customer to decide whether the risk of integrating the respective service is worth taking regarding the role it has to play in the overall business execution – in fact the respective role may already require such a low reputation threshold, that the respective decision may be taken autonomously by the VO Management services (cf. also section I.2).

II.2.c Formation

As stated above we assume that a set of service providers is available by the end of the identification phase, that

(a) fulfill all roles of the collaboration description, as well as additionally required through the Virtual Organisation requirements or by the service providers (cf. Identification above)

(b) are available at the time needed in the form (quality of service) needed, which has been ensured through the negotiation process.

In order to allow for secure communication, as well as for monitoring, enactment of VO specific policies and to enable the distributed enactment of the collaboration definition, the services participating in the virtual organization need to be configured accordingly. The main task of the Formation phase is thus to prepare the operation of a VO in a way that allows for the overall (business) requirements, as stated in section I and in D09. Notably one needs to distinguish here between (1) the configuration of the provided service (respectively the underlying system) itself, e.g. so as to meet the agreed QoS, or to actually deploy the necessary services etc., and (2) the configuration of the components related to the underlying (TrustCoM) framework, i.e. providing the monitor with information what services to supervise how, deploying the policies etc. – whilst the former are related to actually providing the service, the latter cover the aspects with respect to that particular VO "instance".

In principle each service provider is responsible for configuring his/her system so as to meet the VO requirements, and indeed *has* to do so regarding configuration of the actual service ((1) above) and *may* do so regarding the VO specific components ((2) above), in particular when choosing to replace them with own ones or omitting them (cf. section I.3). Notably the TrustCoM components and mechanisms are extensions for those service providers that can not or want not provide their own means of realizing the requirements.

The Formation phase results in a fully configured Virtual Organisation that is ready for enactment – from now on, the service providers are actual registered members of the VO and as such liable for providing the service in the agreed upon form (cf. chapter I).

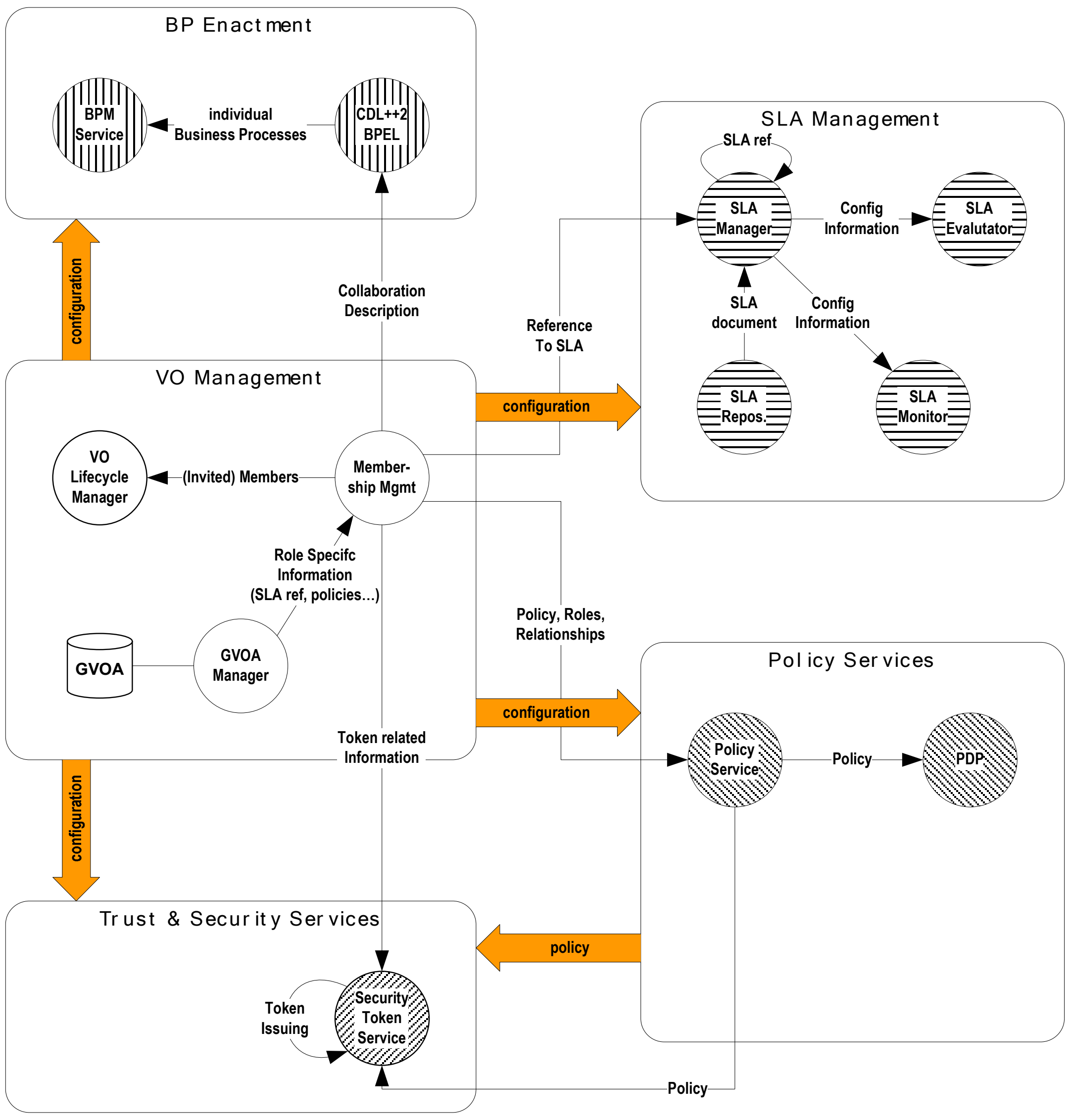


Figure 9: Relationships between components during Formation.

As can be seen from Figure 9, the main relationships during this phase consist in the distribution of VO specific information to each involved service, thus allowing the instantiation and configuration of the respective components so as to meet the according requirements. Notably configuration and instantiation is strongly related to the EN/VO Infrastructure related components that are not depicted here for reasons of space (cf. above) – please refer to section II.3 for the respective details.

In order to realise automated support for Formation, it must be assumed that each service exposes the relevant configuration capabilities, as depicted in Figure 9. There are no special requirements with respect to these capabilities, rather the components may register for specific configuration information as described in the EN/VO Infrastructure section. Accordingly not each component necessarily needs to expose its own configuration capability, but rather some such method should be available per functional subsystem (Policy services, Security support, Trust-related services etc.) that may take over responsibility for passing the necessary data to all involved components, respectively instantiating them – a particular example of this approach can be found in the SLA

Management subsystem, that uses one component (the SLA Manager) for most interactions with other logical systems (see Appendix for details).

It has to be noted that the Configuration phase is not a single definite phase, in the sense of that the processes involved will only be invoked once within the lifetime of the VO, rather just like with Identification, configuration of services may be required at various stages in the VO lifecycle:

(1) as the means to set up the VO, i.e. the second main phase of the VO.

(2) when service providers are integrated into the VO at a later stage, even when identified during the Identification phase – e.g. when a specific service is only required for a limited time and hence does not need to be configured for the whole lifetime of the VO. The actual integration time hence has to be specified in some way in the collaboration description (cf. section I.3).

(3) when service providers are replaced dynamically at runtime by other providers, i.e. involving additional identification processes, too – this is detailed in section II.2.e.

II.2.d Operation

Once a Virtual Organisation was set up according to the requirements derived from some overall (business) goal, the participants may principally start cooperation in a way correlating with the general collaboration description. Such enactment will consist in successive invocations of the individual application services, i.e. the passing and processing of data sets between each other. Business Processes realized by such Virtual Organisations are not restricted to simple data processing, as the actual roles to be fulfilled by the individual participants may be front-ends to any complicated tasks, involving human beings and any type of resources, that however communicate with other participating entities through the means of web service based message exchange.

In accordance with what has been stated in “The Business Model of TrustCoM” (section I.3), one has to distinguish between the VO’s view on the business process and the view of the individual participants: whilst the former focus on the message exchange between the service providers, but does not provide any details regarding the actual execution of the individual roles, the latter describes the details per role and intermediate interaction partners, but does (in itself) not allow insight into the overall process.

A straight-forward approach would hence foresee a central “business process engine” that triggers the actors (of the overall collaboration) corresponding to the pre-defined sequence and forwards the respective data sets accordingly. However, such an approach produces a bottleneck in messaging, would cause unnecessary delays, in particular with huge amount of data, and introduces a single-point-of-failure. Each participant in a virtual organisation will have been provided with his/her role specific information of the collaboration description during the formation phase that each participant will turn into applicable (“internal”) business processes (cf. sections I.3, II.2.c). According to the definition of the collaboration description, these role-specific parts will already contain the relevant contact information, i.e. data source and destination – thus the individual business processes will suffice for allowing “peer-to-peer” messaging.

It has to be noted here that these contact details may be considered private by the respective correspondence partner, thus requiring a trusted third party for message brokering (just as in the case of a central business process engine) – the process of identifying the interaction partner is described in more detail in the sections I.5.a, II.2.b, as well as in the Appendix, sections I.1 and I.6.d. Note furthermore that the contact information may change during the execution of the respective provider's service, which will require updating of the details – see the description of the Evolution phase below.

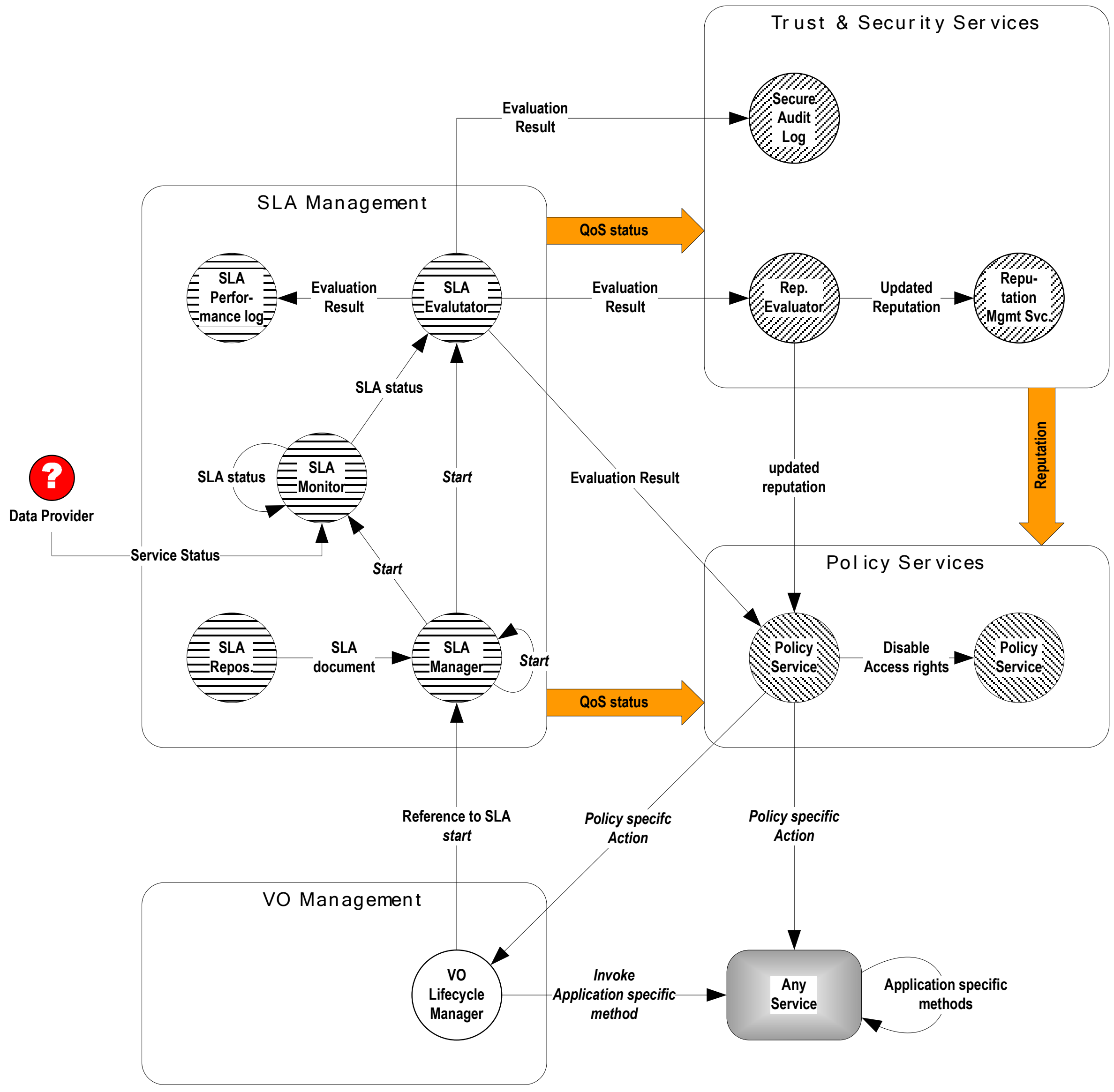


Figure 10: Relationships between components and services during Operation.

Besides for the relationships between the enacting participants, i.e. the application services and the involved supporting services, respectively trusted third parties, the QoS monitoring plays an important role during the Operation of a virtual organisation (see Figure 10):

Services that are subject to QoS terms will be constantly supervised during their enactment with regards to the SLAs agreed upon during the Identification of the respective service(s) (cf. sections I.4, II.2.b and the Appendix, section I.3). This data is gathered through local data providers that in their form and distribution fully depend upon the local infrastructure of that service provider – as such information may be considered private by the respective entity, these means are completely up to the service provider. Hence the respective information does not need to be published and is thus hidden from all other participants (cf. section I.3), since only the converted (based on the SLA parameters) data is of interest to the VO. Notably this conversion allows for “neutralisation” of data, e.g. by converting specific performance parameters into neutral percentage information that is meaningful for the consumer. From the perspective of TrustCoM, the importance lies not in *how* the data is gathered, as long as it is provided in a way that is computational – TrustCoM provides best-use recommendations for these issues, e.g. by integrating the Ganglia and WMI tools, which are not prescriptive however.

The current status of the service provider with respect to the negotiated SLA may be distributed to different interested parties – besides for the customer, in particular policy and reputation related services that require this information for taking SLA related actions (cf. Evolution, section II.2.e), for updating the reputation and for other such purposes.

Policies are the main basis for taking decisions in a Virtual Organisation – not only for triggering the evolution actions on basis of SLA violations, but also for specifying the access right restrictions (cf. EN/VO Infrastructure description, section II.3) and describing the general event-condition-actions in a VO (cf. Appendix, section I.5). Generally, smooth operation of a virtual organisation does not require taking actions that depend on specific events that take place, given that the overall collaboration description is fully specified, so that the actions defined in a policy will mostly relate to triggering Evolution processes (cf. section II.2.e). In relationship with the overall collaboration definition, policies may also be exploited to steer the overall progress depending on environment conditions, like e.g. changes in the market demand – however, without loss of generality, this may considered Evolution, since it entails the reconfiguration of the Virtual Organisation (see below).

II.2.e Evolution

Within its lifetime, the participants and configuration of a Virtual Organisation will most likely be subject to multiple changes, i.e. service providers may be replaced, security settings altered, the business goal redefined etc. Though this is part of any “normal” operation of a VO, we consider it a (sub)phase of its own as it will generally lead to partial repetition of Identification, Formation and Dissolution processes.

The actual causes invoking such an Evolution are various and may actually change between different VOs, as they may be (co)defined by the initiator and the Collaboration Description. Besides for the individual ones, some common triggers may be identified that are recommended to be considered in a Virtual Organisation:

- SLA Violations

  Generally, violating the SLA contracts by not meeting specific QoS related parameters, or – more generally – by not providing the performance as agreed upon during negotiation, will lead to some form of compensation to be provided by the violating party, like paying a fine. However, repeated violations or severe “contract” breaches may lead to complete replacement of the respective service/provider, which implies dissolution (for the specific partner), potential re-identification of service providers (in case the alternatives were not maintained during the initial Identification phase), new negotiation and re-formation.

Whether that member will actually *be* replaced depends on a number of factors relating to the overall goals of the VO – as such, e.g. low time-constraints may be a relevant factor for maintaining even mal-performing parties, since a replacement may delay the overall process too much. In all cases, *availability* of alternative providers will play an important role.

- Reputation drop

  Since business entities will provide their services to more than one customer (or here the virtual organisation), their performance in different relationships will feed back on their reputation (given that they are registered at some reputation management service in that respective business relationship). Accordingly, the TrustCoM VO will analyse updates in the participating parties' reputation and take according actions once the reputation drops lower than the overall requirements allow. Low reputation of a provider implies an increased risk of their misbehaving with respect to performance, security issues and similar aspects, depending on the "type" of reputation (cf. section I.2) and should hence be circumvent by the virtual organisation as much as possible – this may imply increasing the security thresholds, lowering access rights, etc. up to the point of replacing the service (cf. SLA Violations above).

- Changing Location / EPR

  Resources may change on behalf of the service provider, e.g. by changing the address of the respective machine, by moving the resource to a different machine etc. Such changes generally imply modifications of the contact specific information alone (see EN/VO Infrastructure, section II.3) – however, more radical resource changes, e.g. moving a resource to a non-EC state may imply changes on the security aspects of the VO. We must assume that such restrictions, respectively consequences are defined in the GVOA and hence the VO Policies.

- Non-responding Participants

  Participants, in particular services that fail to respond within a given time to VO specific requests (invocations) need to be considered unavailable so as not to delay the overall processing of the VO for too long. The actual timeout delay will vary between individual VOs and even between participants, depending on how time critical the provided service is – as such, e.g. a frequently used calculation web service may be more time-critical than a simple file backup service.

  Non-responding parties will generally have to be replaced, as it must be assumed that the service is down for good and hence can not take over the respective task(s) again.

- Unassigned Roles

  It may be the case that a role of the collaboration description is not assigned right from the beginning, since it is not required for the whole duration of a Virtual Organisation. In such a case, the (potential) Identification and integration (Formation) of the role provider is considered Evolution.

- Lacking Role Providers

  As the identification attempts (when replacing a member or when assigning a role during operation) may not necessarily lead to actual results, i.e. if no suitable service provider for a specific role can be identified, a collaboration may have to be reconfigured completely. This may range from re-negotiation of individual terms up to designing a new collaboration description – this issue is discussed up to some degree in the section on Identification (II.2.b) above.

- Security Violations

  Repeated Intrusion attempts, like repeated unsuccessful authentication or endeavours to access restricted resources, may indicate severe attempts to breach the security of the Virtual Organisation. Thus such attempts will require a reconfiguration increasing the security thresholds, in particular logging of the invocations and their sources, and may possibly even result in changing the contact points to hinder further attempts. As such efforts may also be initiated from *within* the VO (“malperforming” partners), counter-measurements, in particular fines and potential removing of the member, need to be enacted.

- Changing Environmental Conditions / Customer Request

  Since the Virtual Organisation is created to meet a specific, potentially temporary business objective, such as covering a market niche, changes in the environment (e.g. market niche being sufficiently covered by other enterprises) may lead to externally triggered reconfiguration of the VO (either by customer, VO Management or specific VO policies designed for such occasions). This does not necessarily imply termination of the virtual organisation, as it may be possible to compensate for the changes by adapting the collaboration description, e.g. to fill a similar, yet less covered market niche.

Notably, changes in the structure and/or the configuration of a virtual organisation may have an impact on other services and even the whole business process, depending on how well the new structure/configuration meets the overall requirements: since for example a replacement may need to be found urgently, in order to keep the time constraints, the new service may exceed the financial budget thus requiring renegotiation with other, more flexible providers. In the worst case, appropriate replacements for a service may not exist thus necessitating changes in the overall collaboration description in order to find alternative means to realise the business goals. In any case, such aspects need to be taken into account when writing the event-condition-action rules that determine under what circumstances a service should be replaced, or “milder” consequences will be taken (see also Identification, section II.2.b).

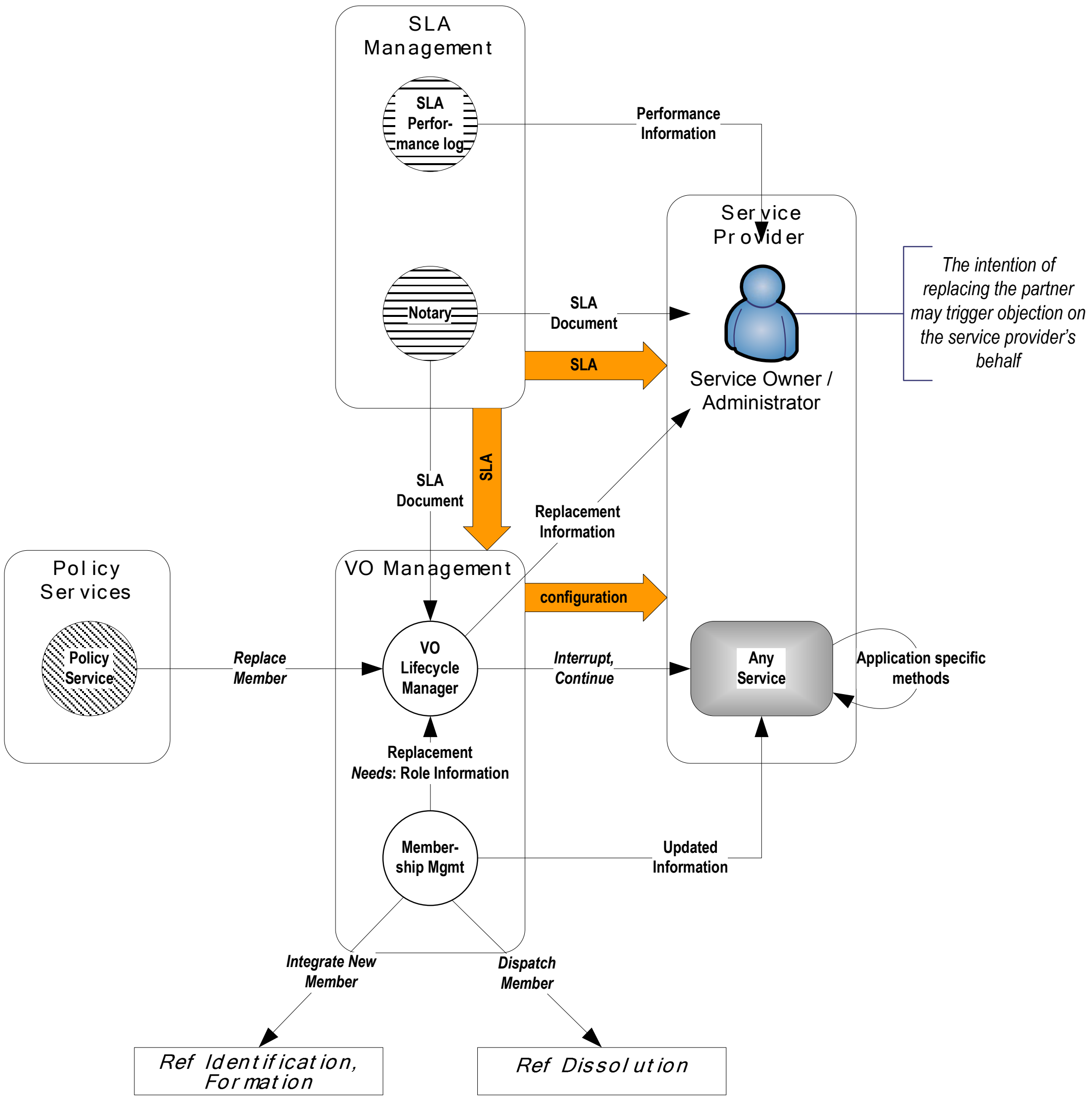


Figure 11: Relationships between components, services and service providers during Evolution.

Figure 11 shows how a policy induced replacement of a specific member relates to other processes in the Virtual Organisation – though focussing here on replacement, the relationships would be similar for a reconfiguration of the participants, yet without making use of the additional steps for actually dispatching the member.

In particular with respect to "severe" measurements, like dispatching a specific party due to SLA violations, i.e. even though the service is still existent, TrustCoM needs to take potential errors on behalf of the VO services into account. This includes failures on behalf of the monitoring and evaluation components, divergence of the SLA etc. The SLA performance log, as well as the reference SLA document stored at the Notary will provide additional information for identifying the source and "justification" of the replacement action. During this process, execution with respect to the party in question needs to be interrupted to avoid failure and reduce the risk of misbehaviour of e.g. "doubtful" participants, i.e. when the respective reputation has dropped below a critical threshold. Such interruption also involves the temporary restriction of all access rights and making security tokens invalid for the time being.

In particular, the relationships are being identical to the ones provided for the Identification, Formation and Dissolution phases of the Virtual Organisation (described in the according sections), whereas the respective processes only differ with respect to how many services are affected by the Evolution: whilst Dissolution will only affect the party to be dispatched (in case of replacement), Identification will also be involved when a non-assigned role needs to be manned or contracts need to be re-negotiated. Finally Formation includes all the necessary reconfiguration steps that involve in most cases *all* participants for adjusting access information, providing updated security tokens etc. (cf. section II.3).

With successful reconfiguration, enactment of the overall collaboration may continue, though potential “rollbacks” need to be taken into account when the execution-state by the respective provider gets lost, respectively can not be taken over, or the execution can otherwise not simply continue from the time of interruption, e.g. due to slight changes in the means of generating the data between the new and the replaced service (cf. appendix, section I.2)

II.2.f Dissolution

The Dissolution phase marks the end of the VO lifecycle, though not necessarily for the whole Virtual Organisation, but potentially only for individual participants that are not required anymore or that will be dispatched, respectively replaced due to violations or similar issues (cf. Evolution above). The two processes differ only slightly since dissolution of the whole VO is conceptually similar to dispatching all its members.

For each member that is to be dispatched the respective execution needs to be halted and, in particular for participants with low trustworthiness, their capabilities of accessing other services and/or data needs to be restricted, so as to reduce the risk of the respective entity inflicting potential damage upon other participants in the Virtual Organizations, or even the whole execution. This implies that all other members are informed of the respective changes in time to avoid problems with executing the process when interactions with the (to be) dispatched entities are required.

Dispatched members will also want to go through the process of auditing, where it is ensured that the entities will get paid for the service they have provided. In addition to this, we consider reputation (and thus trustworthiness) of participants with respect to their performance in VOs an important issue for supplementing security aspects and reducing the overall risks of execution failure – thus auditing for TrustCoM involves assessing the respective providers performance with respect to SLAs and other policy violations, insofar as they are monitored by the Virtual Organization (cf. section II.2.d). Also refer to section I.4.c for details about dissolution with respect to the VO contract.

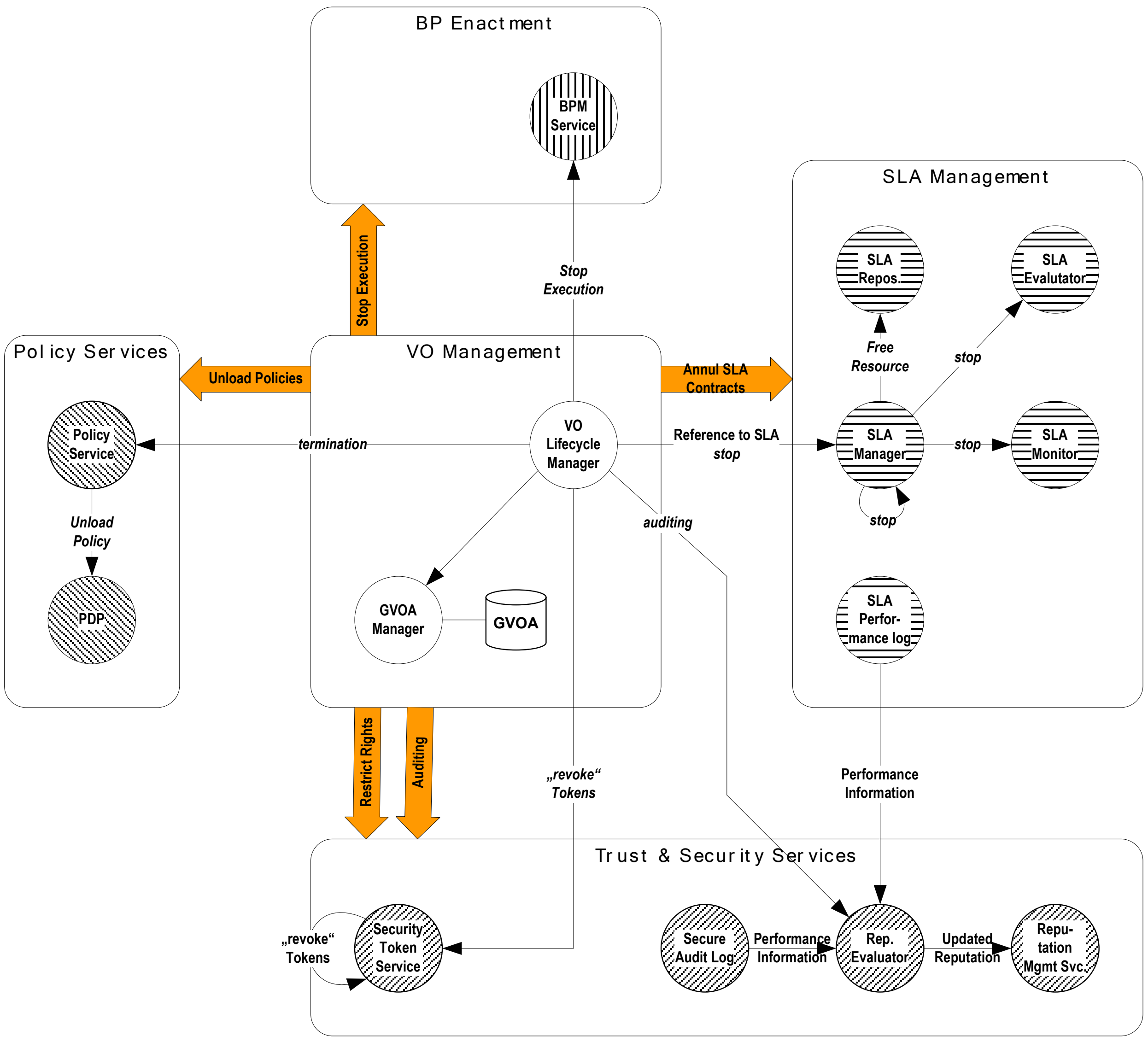


Figure 12: Relationships between components during Dissolution.

Functionally, the Dissolution stops all active business processes of the according service(s) and revokes all security tokens and policies that implicitly define the access rights of the respective service – revocation of such access rights here means that all participants in the virtual organisation are instructed to not accept the respective tokens any more. Furthermore, the SLA contracts with that respective service(s) are annulled and all SLA management related services stopped, since the monitored data is no longer valid and would cause a lot of unjustified violation messages.

Since TrustCoM is not examining means of financial auditing, as already enough systems for this purpose exist, we only focus on the trust-related aspects of auditing, yet the setup of the framework is flexible enough to allow integration of most SOA based auditing systems. In order to assess the performance of the respective participant, in particular the SLA Performance log is assessed for a history of violations (and "fulfilments") produced by the provider. The VO-"local" Reputation Evaluators will be capable of converting the SLA performance information into trust values meaningful for the more global Reputation “repositories” that can be accessed by other interested parties in the Enterprise Network or through other means (cf. sections II.2.a, II.2.b).

Notably, any changes to the configuration will imply changes on all members *involved* with the respective party, as restricting access rights, revoking tokens etc. really requires reconfiguration of all related entities, as detailed below in the EN/VO Infrastructure section. All changes are maintained in the GVOA.

II.3 The Relationships in the underlying EN/VO Infrastructure

As has been noted before, the EN/VO Infrastructure related processes are behaving somewhat different from the aforescribed components, since they build the underlying basis for uniform messaging, accessibility and coordination of interactions. As such, they participate in some way in most interactions so as to enact their functionalities upon the messages and are principally invisible for the service providers. These main functionalities span support for interactions across enterprise borders (notifications, messaging, logging), as well as deployment and management of service and component instances (service instantiator, service instance registry and information repositories). Notably discovery support (discovery services and additional repositories) has already been discussed in section II.2.b of this document, due to its strong usage in that phase and this will not be repeated here, even though these functionalities logically belong to the EN/VO Infrastructure.

From a functional point of view the EN/VO Infrastructure extends the (virtual) services exposed by the service providers with VO capabilities that will allow the entity to make use of the functionalities summarised above. To realise these extensions in a virtual organisation, the respective counterparts on VO Management level are required, so that the diagrams below give no indication of the component distribution across participants and management instances – for such information refer to chapter III.

Though the EN/VO Infrastructure follows the overall VO lifecycle, we distinguish here only 3 phases, namely Setup, Messaging and Evolution since the actual usage of the related components overlap with the overall phases. As such, e.g. the messaging capabilities described below will already be partially used during Formation, whilst Evolution captures also aspects of Dissolution. This way we avoid repeating functionalities in different diagrams – since the phases on this level are generally not explicitly triggered but implicitly invoked through processes on the service level (section II.2), this overlap of phases does not causes functional problems.

II.3.a Setup (Formation)

Most services in a Virtual Organisation will need to be stateful or at least individually configured for the respective requirements, e.g. when the service is subject to QoS parameters. As such, these services will require instantiation and configuration before they can be used – though the instantiation details will be different between service providers and may even be private, relevant configuration details and at least a trigger indicating when the instances are required need to be distributed correctly. Also, TrustCoM specific components implicitly follow the instantiation procedures.

Since these instances may change during enactment of the virtual organisation, generally due to some Evolution processes (cf. section II.2.e), it is not sensible for services to *primarily* interact with the exact Endpoint References (short EPR) since every change would require updating all related information, like e.g. the business process. In order to avoid this problem, TrustCoM deals with individual interaction partners on the basis of *handlers* that are resolved by the message interceptor (cf. Messaging below).

Besides for direct interaction, much information distribution during operation of the VO takes place as topic-based notifications, thus informing (a set of) interested parties of specific events that take place in the Virtual Organisation. Types of events are distinguished by "topics" in order to reduce amount of messaging and to allow subscribers to pre-select only those event that are of interest to them. Even for notifications we see the requirement of privacy, so that some messages may not be received by all participants.

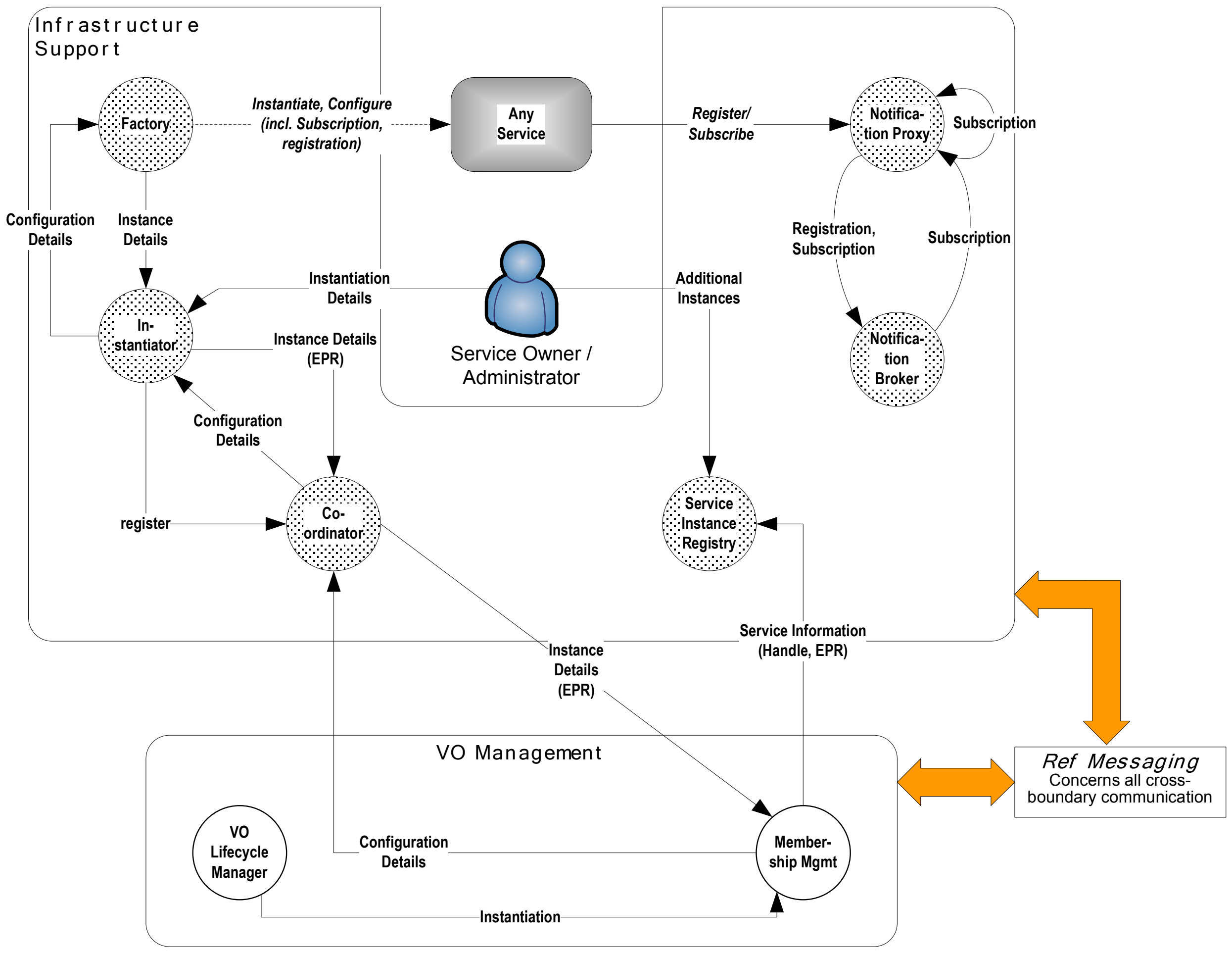


Figure 13: EN/VO Infrastructure interactions to set up a VO.

In Figure 13 we depict the processes, respectively relationships that partake in setting up the components for these functionalities: the instantiation process and according distribution of configuration details proceeds in a coordinated way so as to avoid that instances are required before they have been instantiated. The detailed information of these instances is provided to the so-called "Service Instance Registry" which takes over responsibility for resolving the aforementioned service handlers (see Messaging below for details).

Configuration details and potentially additional information from the administrator will furthermore convey details with respect to what notifications need to be provided, respectively received by the individual services, thus triggering the subscription and registration processes at the notification related components (please refer to Messaging below for more information). As such, a service provider may add individual, local endpoints to the service instance registry, thus redirecting specific invocation calls to (only locally known) service instances or even for enhancing messaging between local messages (cf. below).

II.3.b Messaging (mostly Operation)

In order to enact the functionalities of TrustCoM upon interactions between participants, respectively services in the Virtual Organisation, the messages need to be enhanced and verified accordingly. In theory, all outgoing messages of the actual service (be it application service, TTP or VO Management), should be enhanced in a way that allows uniform understanding within the VO, including identification of the actual endpoint from the handler (cf. above). To complement this, all ingoing messages should be verified with respect to access rights and authentication of the sender.

To realise these capabilities, all participants need to support some kind of message enhancement / verification system as a kind of “frontend” or gateway to the actual service(s). This gateway acts as the actual contact point for interacting with the local services, thus allowing local redirection of messages, that is not visible from outside of the respective service’s domain.

As can be seen from Figure 14 and Figure 15, the processes behind message reception and message sending are principally identical, even though the real functionality, in the sense of purpose, of these mechanisms slightly alters:

Sending Messages

As mentioned, sent messages should principally be extended by the VO specific requirements thus allowing uniform interactions. An additional focus rests on resolving the service handlers that are actually used for sending to valid endpoint references.

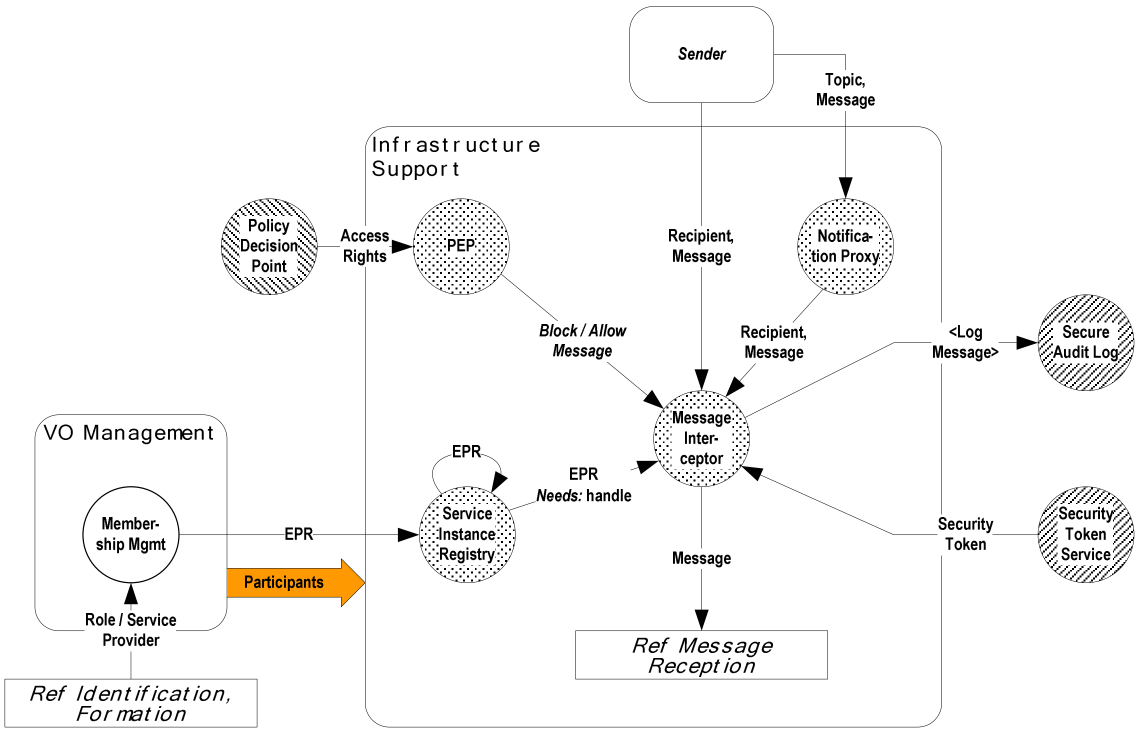


Figure 14: Processes involved in *sending* messages.

As Figure 14 shows, XML documents may either be sent as notifications or as “normal” SOAP messages – the main difference being the recipient, which could be principally any subscriber for notifications but only the referenced endpoint (represented as a handle) for plain SOAP messaging. In case of notifications, the Notification Proxy will provide all the actual interested parties that potentially receive the message (given that they have been subscribed during Setup as described above). All messages may now (1) be verified for whether they are allowed to be shipped to the recipient and (2) be extended by a security token to authenticate the sender within the VO – note that the Security Token Service and the Policy Enforcement Point (PEP) are described in more detail in chapter II.2.

The actual Endpoint References for individual recipients are provided by the Service Instance Registry, which may need to query the VO Membership Management if the respective handle is unknown. This may be due to the fact that the respective participant has not yet been assigned and hence no instance or contact point exists for it – in such a case, trying to access it would need to interrupt the process and trigger Identification and Formation of that respective role. Notably the identified EPR may not be the one of the actual recipient, if message brokering is desired for hiding the true identity of a service from either recipient or sender, e.g. if the sender of a message is not allowed to know the true location of the recipient for privacy reasons. In such a case the recipient as detailed below will consist of an intermediary service that acts as a Broker forwarding the message to the desired endpoint, potentially eliminating information about the sender by replacing the EPR with the handler again.

Receiving Messages

The recipient of a message, be it Broker or actual destination point, will want to ensure that the message was sent by an actual member of the Virtual Organisation and that he/she has the right to access the resource provided by it, thus minimizing the risk of data theft and other potential misuses.

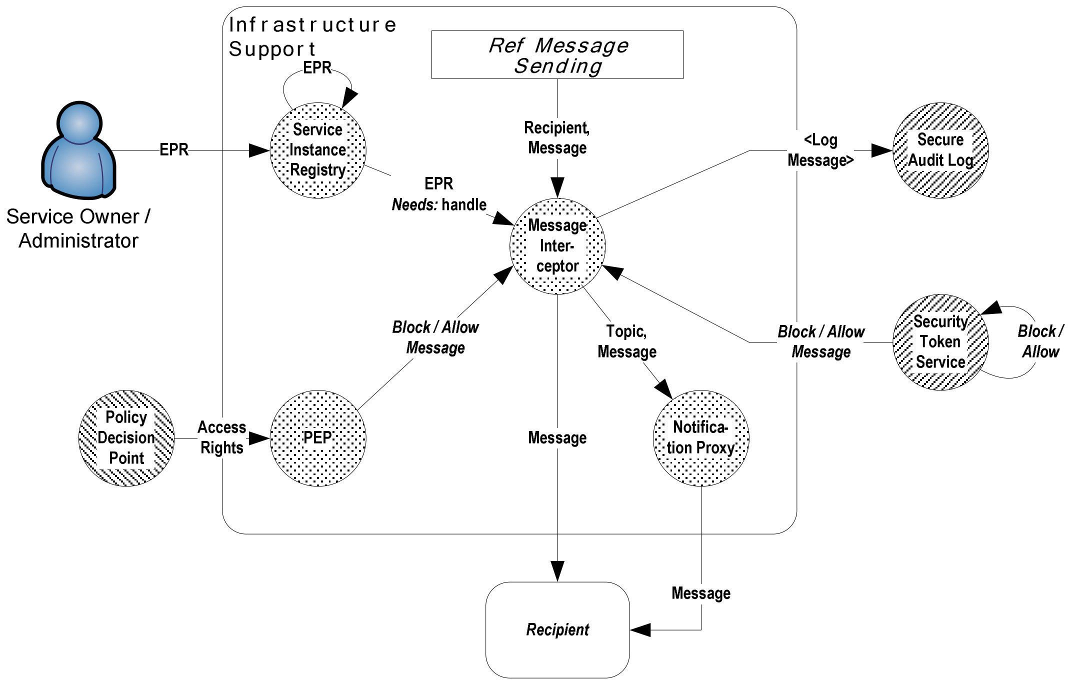


Figure 15: Processes involved in *receiving* messages.

As such, Policy Enforcement Point and Security Token Service will be queried to (1) authenticate the sender and (2) verify its access rights, potentially leading to the message being dismissed and a potential security violation being logged for later reference.

It has already been noted that the frontend as provided by the Message Interceptor does not necessarily reflect the real structure of the participant's domain – thus the actual recipient may need to be identified first on basis of the additional instance information provided by the service administrator (cf. Setup above). Again, the actual EPR may be missing, which in this case requires the interaction of the Service Administrator, since the domain-internal structure is principally unknown to the VO. Note that in case of message brokering the message would leave the domain again, thus starting the processes for sending messages as describe above again. Note also that notification messages may be distributed to different Endpoints within the domain, thus requiring the Notification Proxy as the actual destination of the message.

II.3.c Reconfiguration (Evolution)

With replacing or just dispatching a participant in a Virtual Organisation, it has to be ensured that the respective entity can no longer access resources in the VO, so as to avoid potential misuse of data (cf. discussion in section II.2.e). This implies not only that that service provider can no longer query resources, but also that other providers do not forward information, e.g. as part of a business process to that entity. Since the evolution process may involve several steps, including potential objections on behalf of the member to be dispatched (section II.2.e), preliminary restrictions need to be activated immediately, as the overlap must be considered a potential security threat – these restrictions may be deactivated again, once it turns out that the Evolution procedure is not valid.

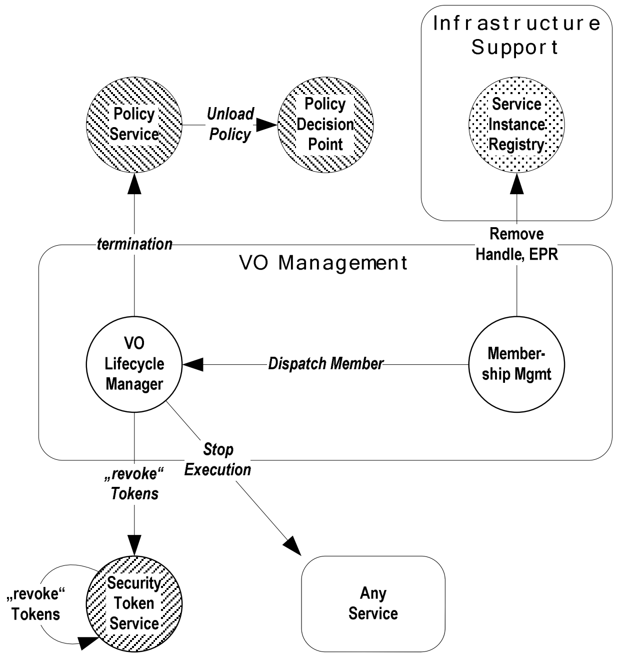


Figure 16: Evolution, respectively dissolution relationships in EN/VO Infrastructure components.

Most of the required processes take place on service level (cf. sections II.2.e, II.2.f), like revoking security tokens, unloading policies and interrupting the execution of the business process – this implicitly blocks all messaging attempts due to the lacking access rights and invalid authentication tokens. In addition to this, the contact information for that respective service provider will be removed to render all information passing *to* that entity impossible.

If the entity will be replaced rather than dispatched, the updated contact details will replace this obsolete data during the repeated Setup phase, as described above.

III Towards a Deployment Model

It has been discussed in the previous chapter (“Four phases of the lifecycle of a Virtual Organisation are distinguished: (1) Identification and Discovery, (2) Formation, (3) Operation & Evolution and (4) Dissolution & Termination. The main tasks to be performed during individual operations of the system for each phase are describe here as scenarios.

III.1.a Establishment of a Virtual Organisation (Identification and Formation)

The first stage on VO formation is the formation of an Enterprise Network (EN) to provide a pool of organizations who are willing to join virtual organizations. Organisations must register with an EN register which acts like a yellow pages telephone book – listing the organization and the services that they are willing to provide. One problem that is not well addressed within Trustcom yet is the motivation to join an EN in a competitive marketplace– why should an organization join one EN rather than another ? It is planned to take guidance from other IST projects that are investigating the issues of VO breeding environments in order to resolve this issue. The EN register and other EN and VO infrastructure services will be hosted by a provider. Business models are presented in other TrustCoM deliverables that show how the hosting of EN and VO services could be a profitable business in itself, probably as added value additions to basic ISP provision.

An organization which is registered as an EN member identifies a business opportunity and has the intention of creating a VO to meet it. They become the VO initiator, defining the goal of the VO, and try to discover the organizations required to make up the VO to achieve the business objective. The VO initiator will interact with a service provided by the hosts of the EN and VO infrastructure to guide him through the creation of a VO – the VO Management service, which is one of several VO services that will be introduced in the scenario. .

Given a specific business objective (provided by a customer or by the VO Management organisation itself) to be realized by a virtual organisation, the VO Management service triggers the derivation of a business processes according to a collaboration definition by contacting BP Enactment. The latter now queries (known) BP Template Repositories for collaboration definition that realise the given task. Such templates contain the next highest-level description of activities, information that is related to the roles involved in realising the processes (i.e. descriptions of the services that fulfil the individual tasks), coordination information (how the services have to interact) etc. which are passed to VO Management for partner selection. It is an open issue to be addressed by the evaluation of the TrustCoM demonstrators whether the level of description of the business objective, the collaboration definition are defined at an appropriate abstraction for the available definition of the market opportunity and the envisaged structure of the VO at this stage of the process. They may be either too abstract or too concrete, in either case the mismatch between the representations offered by the VO management services and the conceptualisation of the human VO initiator will increase the risks of the VO failing - even at this early stage,

VO Membership Manager is invoked with the collaboration definition’s role related data containing information about the structure of the service (operations, interface etc.), the quality it has to fulfil (SLA) and its trustworthiness. This information is passed to the

Discovery Service that for each role to be manned contacts a set of (known) repositories and returns a (sorted) list of potential organisations that meet these requirements.

Once the Membership Manager has received this list of potential participants, the SLA Negotiator on VO Management side is triggered to negotiate the actual terms with the Application Service Providers (starting with the most suitable ones), until all roles are manned. If negotiation fails to cast a specific role, i.e. if none of the respective organisations meet all requirements, the business process to be executed by that organisation and its requirements need re-evaluating.

As soon as all participants for the virtual organisation have been identified, VO Management triggers distribution of the relevant information to each of the VO members – this includes:

e) required authorisations to access other members,

f) interaction and coordination information, like what data to pass when between services

g) VO agreements and policies, as well as

h) other configuration data (contact information, notification topics etc.).

Once all participants have confirmed their configuration, the VO manager is ready to instantiate the VO and enact the overall collaboration definition.

III.1.b Normal operational work

With all VO members configured, BP Enactment starts the execution of the overall collaboration definition by triggering the first Application Service Provider(s) of the workflow and forwarding relevant execution data to it (like input values or location of data files).

Generally, the Application Service itself is responsible for triggering the execution of follow up tasks by forwarding its output data to the Application Service Provider(s) next in the overall collaboration definition (the relevant information, like which services to contact and which data to pass, has been provided during VO formation for each derived BP per role – cf. section I.5.a).

At *checkpoints* in the enacted business processes, the respective Application Service provides status information to BP Enactment, thus allowing monitoring of the overall enactment.

Execution proceeds until either failures occur (like contract breaches, destruction of services etc. – cf. sections I.5.d, I.5.e) or the business process is finalised, in which case dissolution of the VO is initialised (cf. section I.5.f).

III.1.c Dynamic addition of an organisation during operation

Not all Application Service Providers are necessarily identified during the formation phase of the virtual organisation, as some tasks may only be performed after a comparatively long period of time and hence reservation of a service for that duration is unfeasible. Under such circumstances, the difficulty connected with the discovery process has to be considered, as some services are less common and/or are frequently occupied – assuming that the required service does exist at all.

Hence, such an approach is in particular sensible, if the required services are relatively common and only needed for short intervals.

The discovery process is either triggered directly by the need for a non-manned service arising or by a specific discovery-related activity in the collaboration definition. In the first case, the address of a non-existent service is requested from VO Management which in turn triggers the discovery process at BP Enactment, whilst in the second case the identification process reflects a separate task in the business process. Likewise, the latter case allows for discovering new services ahead of time, i.e. before they are actually needed, hence reducing potential delays in the overall execution.

Flexibility of discovery *during* the actual operation of the virtual organisation is limited as opposed to during the discovery and formation phase, since no changes in the parameters of other service providers can be accepted in order to achieve the overall goal.

Once an Application Service Provider has been identified, it is provided with the required configuration data, as described above (section I.5.a). All VO participants that need to interact with this new service are informed of the change, respectively of the addition of a new participant, by providing the contact details (including access authorisation), i.e. Endpoint Reference Address to them. This also applies to Trusted Third Parties services insofar as they interact with the Application Services (e.g. Message Brokering, cf. section II.3.b).

III.1.d Dynamic removal of an organisation during operation

Similar to adding a service provider during the operational phase, an organisation may want to free their resources again, once they are no longer needed by the VO. Accordingly, the Application Service Provider has to be removed from the virtual organisation, if so requested.

Again, the request is either raised directly (in this case by the Application Service Provider him-/herself) or indirectly by the respective entry in the overall business process. In either case, the message is forwarded to VO Management, which triggers re-configuration as follows:

As the service provider has no further rights to access other services and should not do so for security reasons, all respective access rights are revoked. In order to avoid further communication and in particular forwarding of (possibly sensitive) notifications, all references to the respective service are removed – the only communication remaining takes place between VO Management Services and the Application Service Provider.

If a price for service usage was agreed upon, billing takes place at this time – the Log may serve as a means for establishing the actual price. Similarly, the trustworthiness of the service provider is updated on basis of its performance (as maintained in the Log), i.e. the reputation gained through participation in this virtual organisation is forwarded to the Trust Maintenance Service.

Finally, the service provider is removed from the list of VO participants at VO Management side.

III.1.e Replacement of a participant by another during operation of the VO (Evolution)

During the operational phase of the VO, a particular service may need replacing, due to non-performance, contract breaching, simple “disappearance” of the service or similar reasons. In either of these cases, the overall business process is delayed as the current task can not be executed. Since the need for substitution generally arises after the actual task started execution, replacement may even cause a rollback and compensation operation in the involved BPs, as a set of tasks will (in most cases) have to start anew with the new service.

Typically, a Policy Subsystem identifies the need for a replacement as a reaction to a specific event, like e.g. contract breaching and notifies VO Management. The latter may verify the correctness of data by directly requesting information from the respective Application Service Provider.

VO Management then triggers re-configuration of the VO as described in section I.5.d (insofar as the service is still available for contacting), i.e. it removes the service to be replaced from the virtual organisation.

At the same time, VO Management triggers the Discovery service to identify a new service provider that fulfils the criteria as defined for the one to be replaced. The Application Service Provider will then be provided with the necessary information as during the dynamic addition of an organisation (cf. section I.5.c).

Once set-up accordingly, i.e. the old service removed (all tokens and related information revoked) and a new service configured according to the VO’s needs, BP Enactment triggers execution of the Application Service. Since input data may have been lost during the replacement process, BP Enactment furthermore triggers the Application Service Providers representing the preceding tasks in the overall business to re-distribute their data to this new service.

Note that, similar to dynamic addition of organisations (cf. section I.5.c), circumstances like relevance of that service for the overall execution, availability of replacements etc. play a significant role in whether a service should be replaced. Data like amount of service providers initially identified for that role (during the discovery phase) may be crucial for further proceeding.

III.1.f Dissolution of the Virtual Organisation

Once the overall business process has executed its final task, respectively destruction of the VO is triggered by VO Management (e.g. due to grave failure), the virtual organisation may be dissolved, i.e. all partners are removed from it as described in section I.5.d.

Once all Application Service Providers have been detached, the configuration of the VO Management Services will be reset up to the point of reuse. This means that the VO Manager may decide to maintain e.g. the collaboration definition for later execution and keep a list of all service providers that performed well so that they may be contacted again.

Generally, however, we will consider the virtual organisation to be reset completely at this point, i.e. any new request to a VO Management service provider will have to start a complete new setup procedure as described in section I.5.a.

The TrustCoM Architecture") that the relationship model as presented does not provide insight into the actual distribution of components and services across a "real" Virtual Organisation, as there is no distinction made between the service "types" the components *could* belong to. Furthermore, the relationship model does not convey details about the actual usage of the components, since it mainly represents what kind of information *can* be provided and which components typically use them, not in what order these transactions would proceed to realise the functionalities.

This chapter tries to redeem these shortcomings by exemplifying the usage of TrustCoM architecture in the context of an actual (though simplified) usage scenario, that is roughly aligned to the Collaborative Engineering scenario (cf. D10, D20) – such a description provides the lacking usage details at the expense of restricting the architecture to a specific "type" of Virtual Organisation. The view will be restricted to a very limited use case with only two participants, in order to maintain a maximum of overview. Even though this model will detail what components are typically deployed on which systems, the main focus is not on the deployment procedures, as typically conveyed by this type of diagrams, but rather on the interactions in a fully deployed VO system.

III.2 Deployment Overview (One possible VO Instance)

In the following descriptions we assume that a customer needs to reach a business goal that can be realised with two main application service providers (roles, cf. section I.3). As described below, it must be assumed that such services exist and are available within some form of Enterprise Network. It does not matter here whether the customer deploys the according components of his/her own, or uses an existing VO Management Service Provider with the capabilities to instantiate a Virtual Organisation.

We furthermore assume that each Application Service Provider participating deploys the full TrustCoM support in his/her infrastructure, i.e. without integrating proprietary systems to replace individual components, like e.g. the security token service. As such, Trusted Third Party functionalities focus on supporting in particular management related tasks and do not need to supplement ASP capabilities in themselves (cf. below). Overall, the scenario is very basic with respect to functionalities and requirements towards the VO, we even omit issues like message brokering as we initially focus on conveying the main aspects rather than the full complexity. As noted, this may change with future versions of this document.

Similar to the relationship model above (chapter II.2), the deployment overview needs to be segmented into several "views" so as to not unnecessarily complicate the diagrams – in particular we distinguish between[16] (1) the "Service Level view" and (2) the more detailed "Component Level view":

III.2.a Service Level view

The "Service Level view" is the most high-level view on the Virtual Organisation described above and as such provides an overview over the interacting systems rather than the actual transactions between the individual components (section III.3). From this point of view every interaction partner exposing its own interfaces, i.e. each individual *service* in the

[16] Note that the naming of this distinction differs slightly from previous version of this document

VO is considered a component in the diagram below – this reflects the actual view from individual participants of the VO on other entities which generally expose this one interface for interaction. Without loss of generality, each invocation on this level may be regarded as a web service call – notably the interfaces really represent “virtual endpoints”, as detailed further in section III.3.a below.

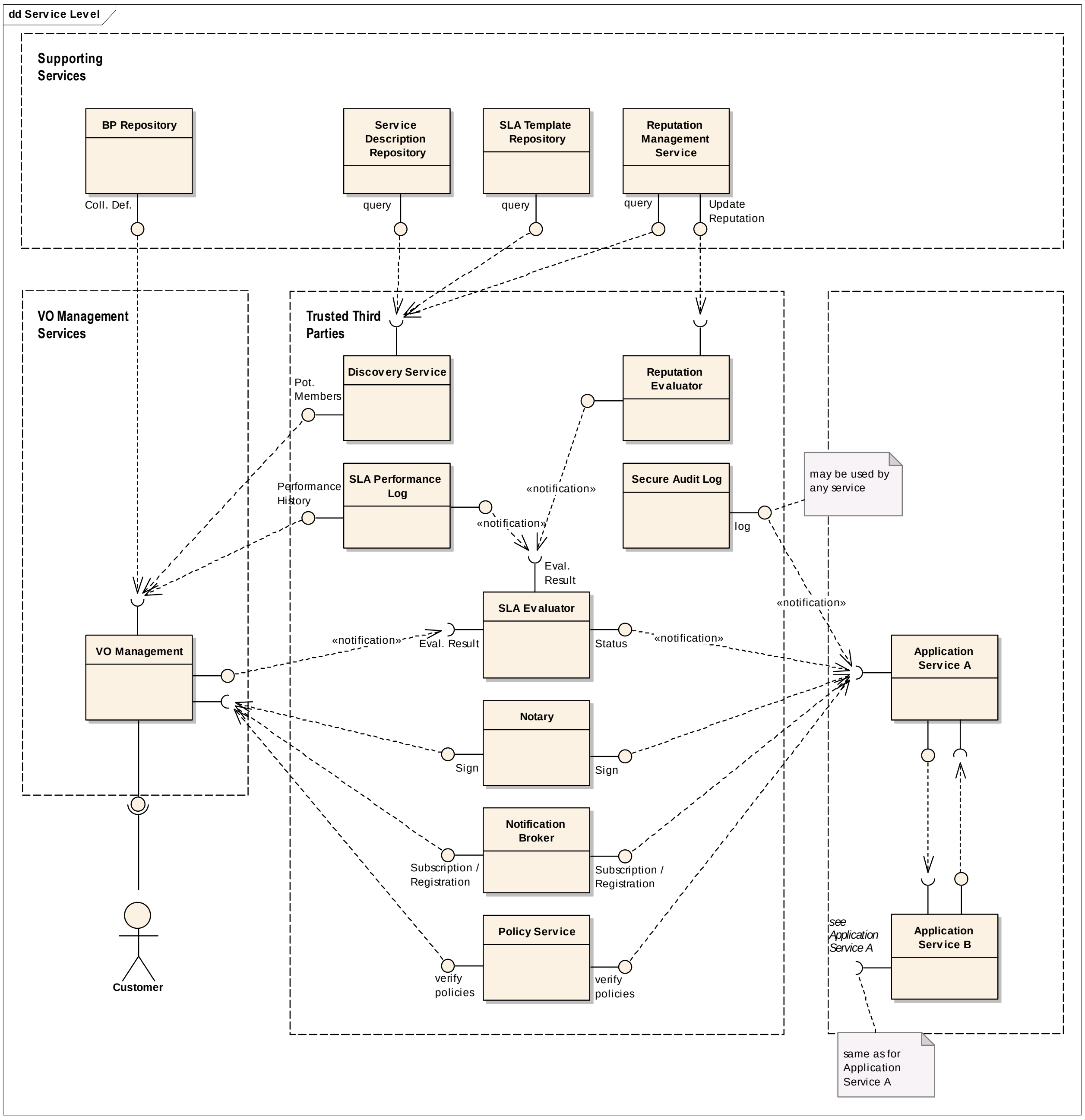


Figure 17: High-level view of the services involved in (this) Virtual Organisation.

Figure 17 provides an overview over these main services (represented as classes) and the interaction capabilities (interfaces) they typically provide to other services (arrows). The services have been grouped according to their type of functionality, as described in chapter II.1.a, namely Supporting Services, Trusted Third Parties, Application Services and finally

VO Management. The details of the actual interactions between participating services will be outlined in section III.3 with respect to the individual VO lifecycle phases' requirements.

Note that each of the services depicted in the diagram are assumed to provide, respectively integrate the VO specific enhancements, as detailed in the following section.

III.2.b Component Level view

As discussed in chapter II and section I.3, we assume that the actual participants in a Virtual Organisation can be represented as individual services that fulfil a specific role in the overall Collaboration Description. This does not imply that there is a direct mapping between role and actual tasks to be executed to perform this role, i.e. a service in the sense of a VO member does not necessarily comply to just one actual resource that fulfils the respective actions, but rather that it maps to a business process of its own triggered by the invocations and involving any amount of actual resources – these details are and in fact should be of no interest to other participants in the VO, including the VO Manager, as well as the customer, as long as the individual infrastructure, its according business processes, the contracting model do not clash with the overall policies of the virtual organisation, like e.g. that subcontracting is not acceptable.

We hence enhance the functionalities of this virtual endpoint with as little impact on the actual infrastructure of the service provider as possible. As already explained above (cf. sections II.1, II.2), there are different requirement with respect to "VO enhancements" for the individual types of services (see section II.1.b), so that typically each type will deploy different components. In the following we represent – for each type – the most typical deployment for the given usage scenario. Note that such deployment is not prescriptive, but only one example of how a working system could be realised on the given architecture.

Supporting Parties

As discussed, supporting parties are "outside" of the Virtual Organisation and as such do not experience any alterations through TrustCoM – in fact one of the most important issues to observe from a technical stance is that the existing and recommended services (mostly repositories) can be used. TrustCoM caters for this through the use of web service standards and respective recommendations.

Trusted Third Parties

Though Trusted Third Parties are in the first instance quite "generic" services (logs, notary, notification broker – cf. Figure 17), they nonetheless are VO specific: as discussed above (section II.1.b), TTP services are considered "trusted" since they may have to maintain sensitive data for the participants and keep them private. Accordingly TTP services need to observe *at least* the security rules and the access rights of the Virtual Organisation. Thus these services require an enhanced infrastructure, similar to the full setup of the Application Services (cf. Figure 18).

Since TTPs may interact with other services for querying data or triggering actions, they will also have to be provided with the necessary contact information that allows identification of the EPRs of the respective interaction partners, or at least usage of message brokers to convey the relevant documents. Note that Notification is also a valid form of interaction for Trusted Third Parties. Being stateful themselves, at least with respect to maintaining the configuration parameters of the VO, these services furthermore need to provide some form of instantiation and support the means for registering the according instances in the virtual organisation.

For the given usage scenario, TTP services are not themselves subject to Service Level Agreements and their tasks are “atomic” in the sense of that they do not enact business process for providing the required functionalities.

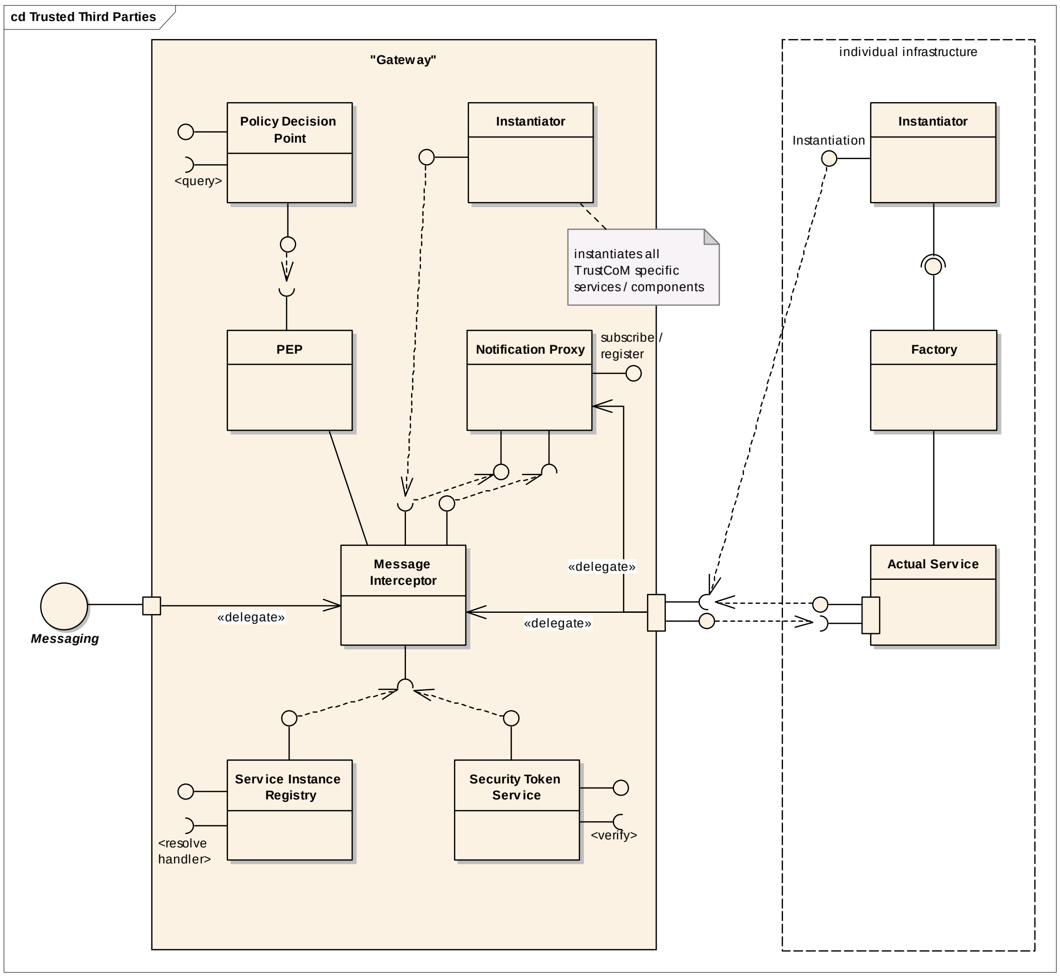


Figure 18: Typical components structure for Trusted Third Parties.

As with any other participant, the TrustCoM framework tries to influence the infrastructure of the TTP service provider as little as possible: the aforementioned message gateway thus needs to enact the authentication and access rights issues for incoming transactions, and needs to enhance outgoing messages with the necessary identification tokens within the VO, as well as identifying the actual EPR of the recipient (or at least of an intermediary).

Notably, the structure of this messaging gateway is principally identical for all service providers participating in the Virtual Organisation (cf. details below). It will also be noted that more interfaces are depicted in all the diagrams of this section, than are actually explained or interactions are defined for: most of these either relate to interactions with other services, or have been omitted – if they do not contribute to the better understanding of the service – in order to maintain a more clear diagram structure. All these aspects will be elaborated in more detail with respect to the lifecycle phases in chapter III.3.

Application Services

Application Services are the main contributors to a VO's business goal(s) and as such require the most complete configuration and setup, thus realising the requirements identified by TrustCoM (cf. chapter I). An Application Service as provided to a Virtual Organisation consists in principle of any number of resources, services and (human) workers that are aggregated and directed by a business process to expose a specific functionality (the "role"). The individual infrastructure, distribution of tasks, as well as the business process details are principally completely up to the service provider, as long as they comply with the overall VO requirements. The aggregated functionality is exposed as a single (web) service with no information about how the exposed methods map to internal processes.

Though the TrustCoM framework provides support for enacting a business process, as well as guiding the instantiation and configuration processes, these means are not imperative and may be replaced by any other approach the service provider prefers, as long as the interaction and the monitoring are guaranteed.

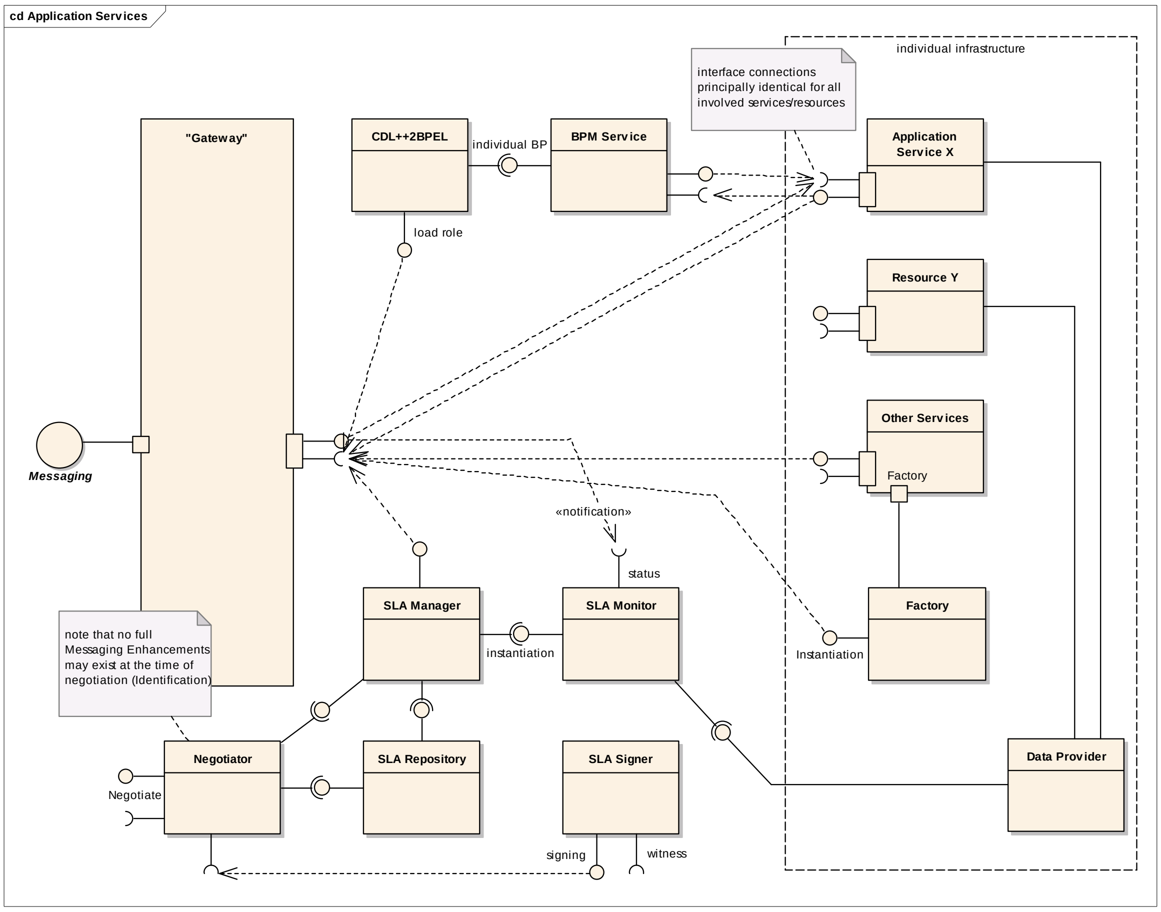


Figure 19: Typical components structure for Application Service Providers.

Figure 19 shows a typical component structure as required for Application Service Providers in the given scenario: besides for (1) the gateway structure like the one for the TTP services, an Application Service would generally show support (2) for Service Level Agreements, and (3) for the enactment of business processes.

(1) Though the gateway structure is principally identical to the one used by a Trusted Third Party, an ASP may want to exploit the functionalities for redirecting specific messages to the appropriate endpoint by enhancing the service instance registry with details about local instances so that individual invocations may be mapped to different endpoints. Since the service instance registry is a local component, the respective information is not exposed to other services and kept private.

(2) Application Services are principally subject to Service Level Agreements, i.e. have to maintain a specific quality of service as negotiated with the customer. From the VO's point of view, this requires in particular the capability of providing information about the current performance related status. Since neither customer nor service provider may trust each other to perform *neutral* evaluation of the QoS data, we place evaluation here at a Trusted Third Party side – note that a service provider may want to make use of local evaluation for management purpose, which is not depicted here, however. As has been noted before (cf. section I.3), the issue of the private infrastructure implies that the structure for performance measuring is equally unknown to the VO – likewise the data provisioning systems may be individually configured to meet the service provider's infrastructure and the QoS requirements (cf. Appendix for details).

(3) As an application service provider exposes aggregated services that may be realised through complicated business processes – we propose a "CDL++2BPEL" component that may be used for turning the Collaboration Definition based role description into a business process that can be enacted by the provider.

VO Management

The VO Management related service(s) take a particular role in the overall organisation and enactment of Virtual Organisations by managing and maintaining the participants, supervising the main processes and steering the lifecycle phases. Its main goal is to represent the customer's interest with the additional enhancement and capabilities to realise them. As such, the VO Management service differs from other participants in the Virtual Organisation, since it needs to ensure that all members can interact with each other and observe the overall and specific requirements, including policies, access rights and QoS definitions. Notably, this does *not* imply that the VO Management service itself needs to be capable of "understanding" all the related information, e.g. evaluation of SLA status information or interpret policy requests, but that the respective mechanisms are catered for in the VO and that the consequences are enforced accordingly.

These management functionalities may be realised through one or multiple services, either realised directly by the customer or through some intermediary providing the means to host VO Management services, i.e. acting as a "normal" service provider for these particular types of functionalities.

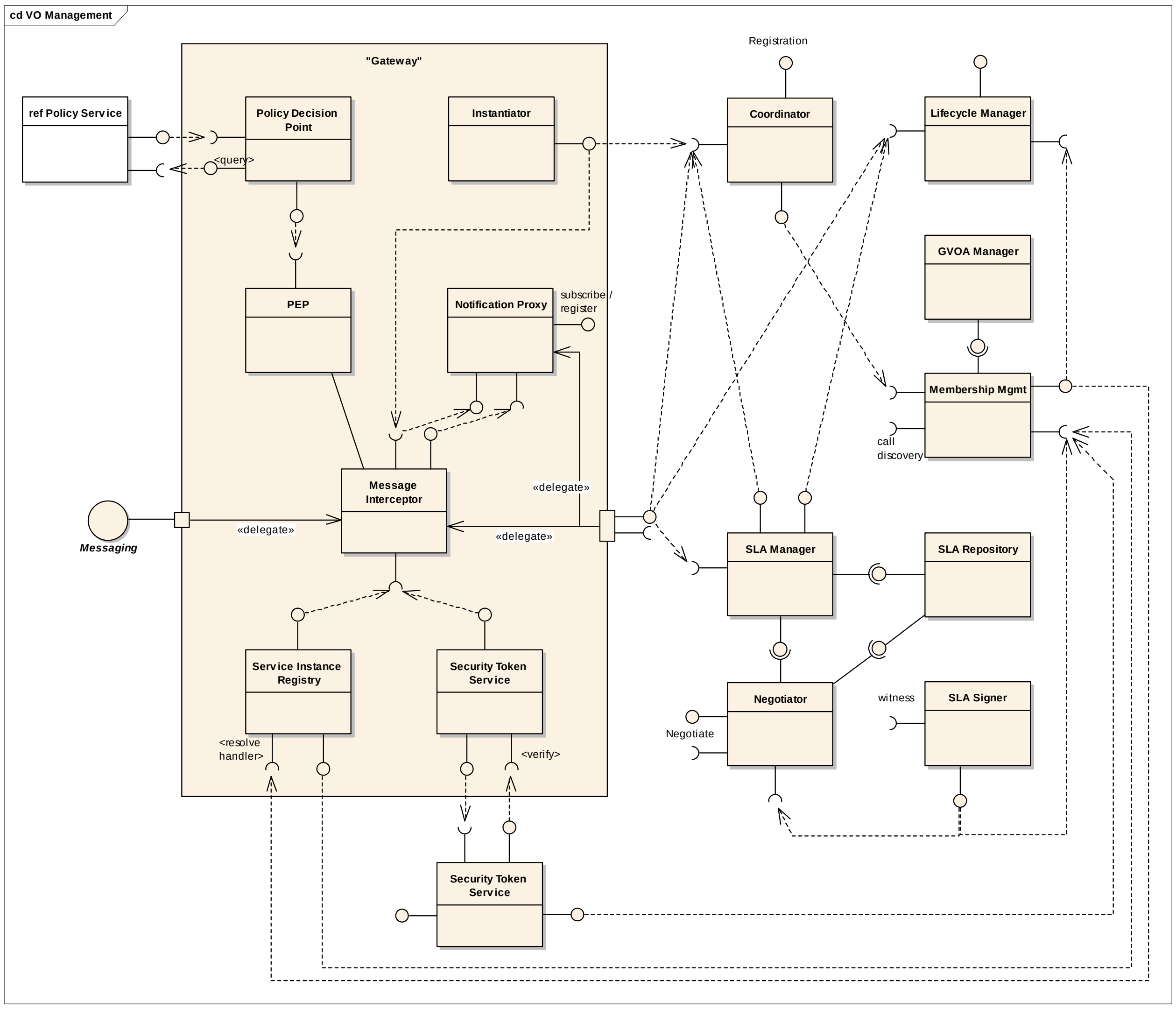


Figure 20: Typical structure of a VO Management Service.

In its position, VO Management is the first member of a Virtual Organisation and the one service responsible for identifying the required participants for reaching the business goals as defined by the customer (VO Initiator). Likewise, it needs to provide the means for turning the goal description into a collaboration description that includes details about the roles that need to be manned for enactment. We do not presume, however, that such a service can realise this process which requires detailed knowledge about business processes, but that it uses supporting services for this task – rather, VO Management will take this elaborated description and extract from it the individual role definitions, including such requirements as QoS parameters and actual task descriptions. The service will furthermore take consequences for adapting requirements, respectively collaboration description, depending on the identified and available participants, i.e. whether all roles can be manned and whether the overall requirements can be fulfilled by it.

In order to build the virtual organisation, the VO Management needs also be capable of (a) defining and (b) distributing the specific configuration details, as well as supporting the instantiation processes. Configuration details in particular comprise VO specific identification tokens, access rights and communication information (handlers of parties, as well as actual endpoints), so that interaction between the parties that need to exchange data is possible through one way or another – as has been noted before, this may involve message brokering, dynamic interaction parties etc. Please also refer to the Appendix for details.

The full description of the VO as realised by VO Management is maintained in the so-called General VO Agreement (GVOA) that relates legal and electronic contracts – see section I.4.

III.3 A Potential VO Instance

With the structures depicted above, it is in principle possible to build up a virtual organisation that, though simplified, shows almost all of the capabilities of TrustCoM, besides for the ones requiring large groups of participants – this involves mostly the enactment of widely distributed collaboration workflows, more complicated message brokering issues and similar, i.e. capabilities that are reflected in smaller VOs, too.

The simplified VO model we use for demonstration purposes here consists mainly, as already outlined above, in two Application Service Providers, i.e. the underlying collaboration definition involves two main roles, and several Trusted Third Party services, Supporting Services, as well as the VO Management service. As such, the collaboration may pursue a variety of purposes and we align our scenario slightly to the Collaborative Engineering testbed (cf. D20):

A customer wants to enhance an existing design of an airplane (a boat, a car) with multimedia entertainment capabilities that may conflict with the existing on-board systems and hence may require reconfiguration. Without loss of generality we assume that the design data is transported directly, thus not requiring storage provisioning services (though realistically more feasible). The collaboration definition will thus involve a multimedia designer role and a designing and testing role for the on-board systems. The details of the requirements (budget, QoS, policies etc.) will be omitted here for the sake of simplicity and as they are discussed in more detail in the testbed related documents like D20 and D21. For demonstration purposes we simply assume that a QoS requirement consists in providing progress reports every day and finishing the complete design after a period of three months.

The collaboration itself is quite straight-forward and involves the customer / VO Management providing the design of the on-board systems to the systems tester and the details of the multimedia requirements to the multimedia designer. The latter will develop an initial design and provide it to the system tester which will analyse the data and try to redesign the on-board system to avoid conflicts, potentially requiring another redesign from the multimedia provider. The result of the collaboration will consist in an enhanced design for the on-board system integrating multimedia capabilities.

We will assume that all application service providers provide the fully configured system as detailed in chapters II and III.2.b, and that the supporting and TTP services required for privacy, discovery and similar issues are identical to the ones listed in section III.2.a.

In the following the main interactions of the services and components with respect to the individual phases of a VO will be roughly outlined – since the processes have already been discussed in previous sections and since the actual interactions of each subsystem are detailed in the Appendix to this document, we will not provide thorough discussions of the respective interaction diagrams. Note that this section is still under development and may show slight inconsistencies with respect to the messaging details.

III.3.a Gateway Functionalities

Similar to the argumentation in section II.3, we will make a distinction between the gateway related functionalities and the actual interactions between services, respectively components to reduce the complexity of the diagrams: since the gateway structure will behave principally identical for all services and communication, the respective processes need not be repeated every time.

As opposed to section II.3, the aspects of the gateway mechanisms will be presented first to allow for better understanding of the direct communication between services as will be described in the lifecycle-specific sections following these elaborations.

Instantiation (Figure 21)

Though the gateway itself is not directly enabling instantiation of service-specific components and resources, it nonetheless requires instantiation itself: besides for being a unique representation of the service provided, it furthermore requires stateful information for maintaining the access rights, security tokens etc., i.e. the configuration details.

Furthermore, the EN/VO Infrastructure related instantiation & deployment capabilities allow for support of coordinated instantiation of the provided resources, in as far as desired by the service owner. When triggered by the VO Management to instantiate and configure the related components, the registered components' and services' factories (or similar) will be triggered with the required information, such as an SLA (or at least a reference to identify the SLA document).

The endpoints of the actual instances will be fed back to VO Management to fill the membership details (that will later provide the information for the Service Instance Registry) – note that this concerns only endpoint references that are required for messaging and not considered inaccessible (in the sense of not directly exposed) by the service owner.

(Re)Configuration (Figure 22)

To use the gateway, it needs to be configured with information related to access rights, individual policies, security tokens and the handler-instance mapping – changing these details during Evolution (cf. section II.2.e) is principally identical to the interactions depicted below and shall not be elaborated additionally.

Besides for providing, respectively updating the information details, the configuration will also involve notification related subscription and registration, i.e. the steps required for registering the individual components / services either as consumers of events related to a specific topic, or as producer of such events, so that others may subscribe to it. Note that for both the handler-instance mapping and the subscription, the endpoint known to the individual services may not be identical to the actual source, respectively destination of the interactions, if a broker acts as an intermediary transport medium (see also sections II.3.b and II.2.d).

The service owner is principally capable of extending the Service Instance Registry with further mapping details for purposes of communication internal of the service provider's domain or to redirect individual invocations to specific instances (cf. section II.3.b). He/she may also make use of the policy subsystem to enforce local policies, either by adding them to the local policy decision point or by registering his/her own at the PEP in addition to the existing one.

Messaging (Figure 23, Figure 24, Figure 25)

The main task of the gateway consists in intercepting in- and outgoing messages to authenticate the sender, to verify his/her access rights of the sender, to enact individual policies, to redirect messages to the endpoint (using a handler) and to expose the individual service's & components' functionalities as (enhanced) web methods:

In principal, three different processes have to be considered in order to realise these capabilities: interception of incoming messages (Figure 23), of outgoing messages (Figure 24), and messaging via notifications (Figure 25). Note that notifications are rather a means of managing destinations and sources, than a substitute for the gateway capabilities, i.e. messages generated by the proxy will be dealt by the gateway like any other messages.

III.3.b Main Functionalities

The following paragraphs will comprise the interactions between services and their subsystems with respect to the individual lifecycle phases. Again, the details regarding the individual subsystems are provided in the Appendix to this document.

Note that these diagrams omit the gateway related processes as they are identical for all cross boundary communication (cf. above).

Identification (Figure 26)

In order to establish a Virtual Organisation, a collaboration description needs to be derived from the goal definition provided by the customer / VO Initiator: Such a collaboration description will have to include (1) information about the functionalities that participants have to provide, i.e. their "roles" (basically a WSDL), (2) the requirements of the roles with respect to (a) quality of service, (b) policy & security settings and (c) reputation. These requirements will be derived from the constraints, as provided by the customer, by distributing them sensibly across the roles – however, we consider this a task of the business expert "behind" the BP Repository, as this is outside the scope of TrustCoM.

The actual discovery processes consist in a series of queries to different repositories and registries. Note that we consider negotiation part of identification as this will determine whether the resources are available in that form (cf. section II.2.b).

Formation of a Virtual Organisation (Figure 27)

Participants in the actual Virtual Organisation need to be invited, instantiated and configured so as to meet the according requirements, as well as to (legally) bind them to the VO (cf. section I.4). The respective processes comprise in particular triggering the individual participants with the relevant instantiation and configuration details – this is mostly "gateway" related and thus discussed in section III.3.a.

Once all participants' services and components have been instantiated and the respective instance details gathered, the Service Instance Registries of all members will be updated to enable interaction with the respective endpoints, insofar as allowed (cf. discussion in chapter II.2.d).

Enactment of the Collaboration – Operation (Figure 28)

From the service level point of view, the operation of the Virtual Organisation consists mostly in the data exchange between services in a way that pursues the overall collaboration descriptions. The diagram in Figure 28 does not comprise the full process as outlined above, but only the main processes: distribution of initial data, starting the execution of individual services and alternating invocations.

Since the services may be (or in this case: are) subject to quality of service agreements, supervision of the performances is a relevant aspect of VO operation. Performance of a system may have an impact on the configuration of the VO (e.g. Evolution, increasing security settings etc.) and in particular on the respective service's reputation. Notably, the reputation of a participant may trigger reconfiguration of the VO, too.

Reconfiguration of the Virtual Organisation – Evolution (Figure 29)

Services that violate the requirements and/or agreements within the Virtual Organisation may trigger a reconfiguration process that will consist in minor changes, like increasing the security settings, or even in replacing, respectively dispatching the service in question from the VO.

Note that dynamically adding or dispatching a member due to other issues, like on request by the party, or that the resource is no longer required, is also considered evolution.

The Evolution processes are really a mixture of dissolution and identification / formation, but with the addition that the respective participant may want to object to being replaced, which will lead to a series of validation steps, verifying whether the replacement is justified.

Termination of the Collaboration – Dissolution (Figure 30)

At the end of the execution, be it of a single participant (either due to evolution or due to being no longer required) or of the whole Virtual Organisation, the respective (collaboration) relationships need to be dissolved in order to avoid security issues. This covers in particular the removal of the respective Endpoint from the Service Instance Registry, deleting the access rights and declaring the according Security Token as invalid, to make further resource access impossible.

Furthermore, all (SLA) contracts will be terminated and auditing of the respective service initiated – this may in principal involve financial auditing, which, however, is not represented in the diagram.

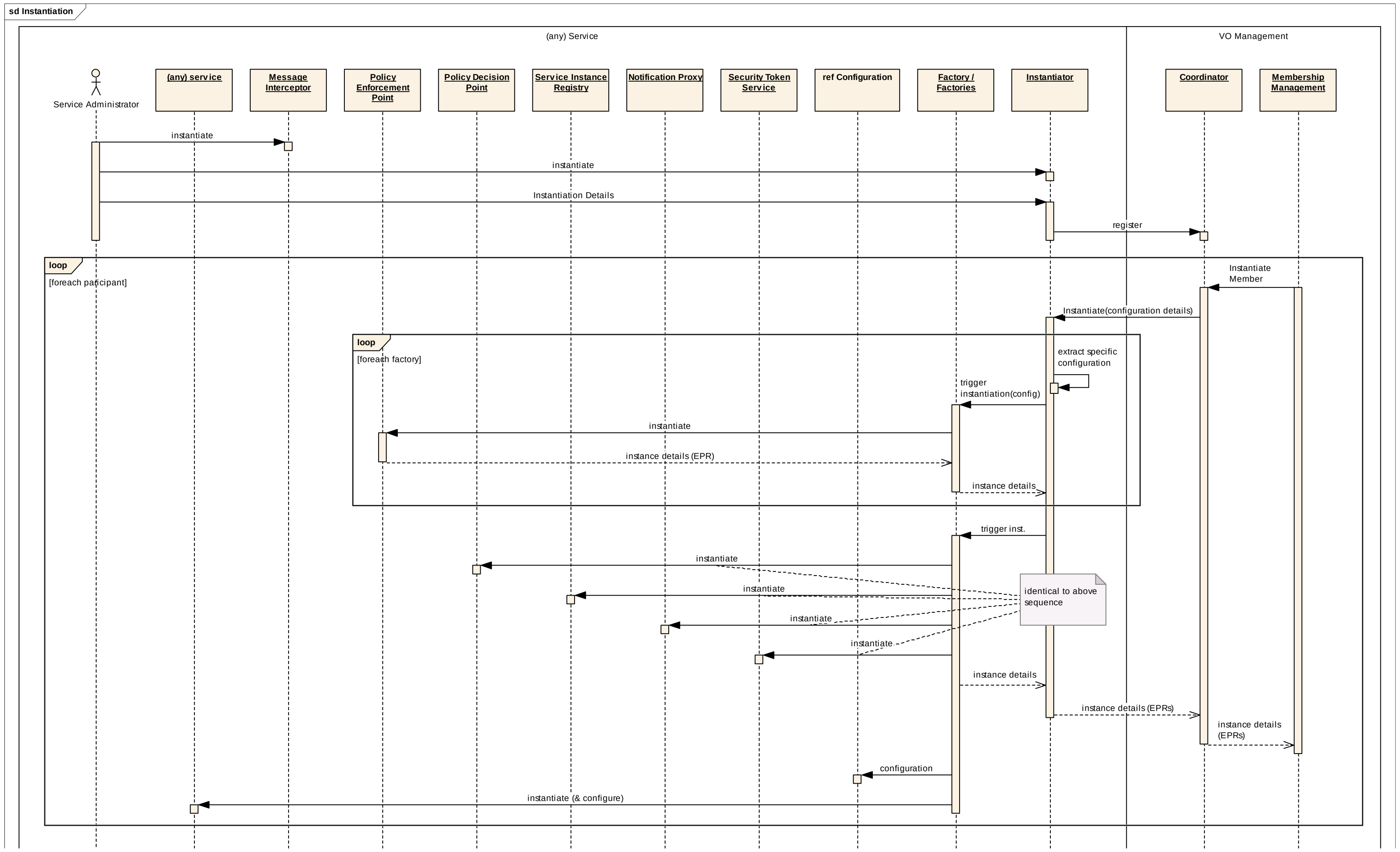


Figure 21: Message exchange during Instantiation of the gateway.

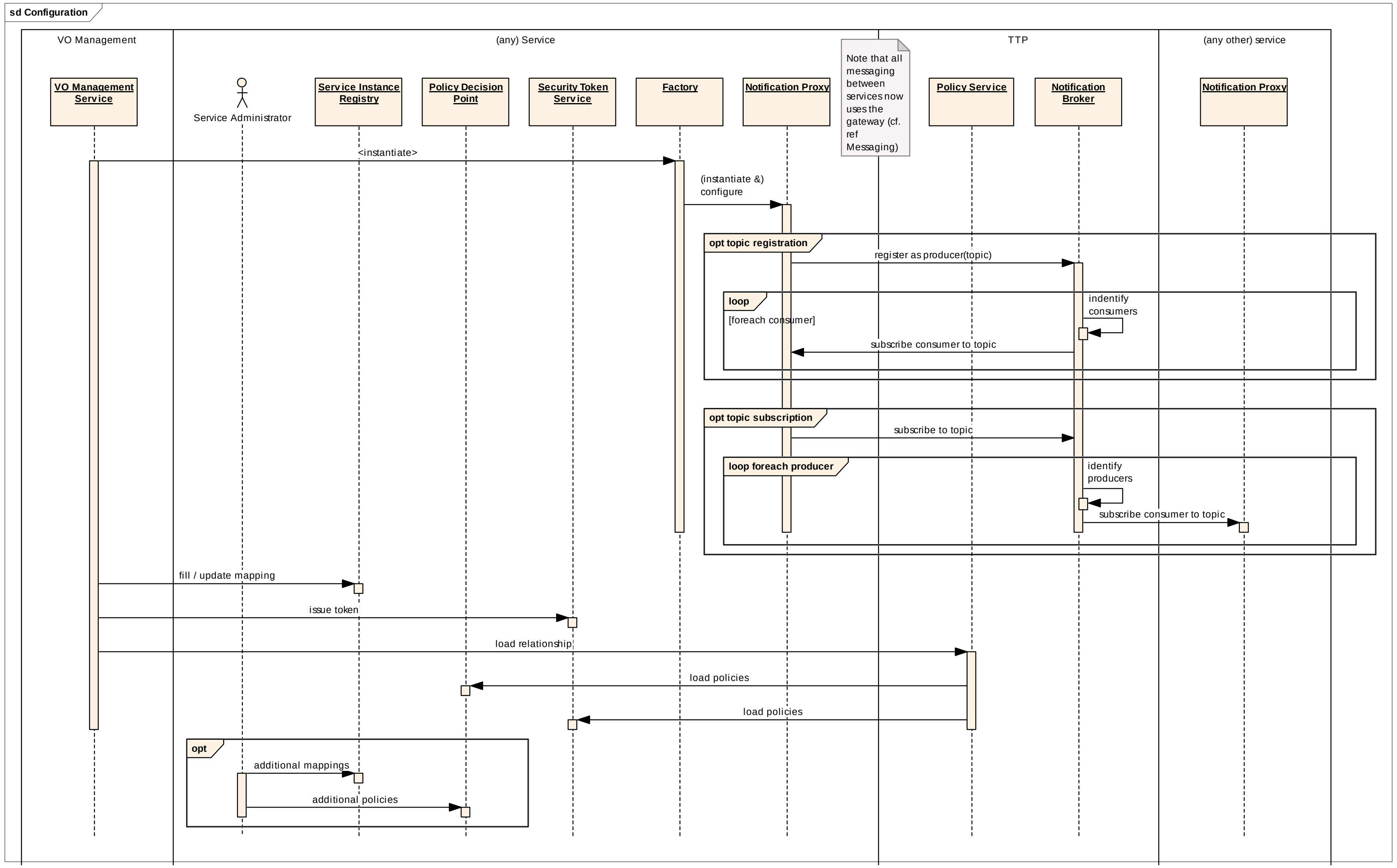


Figure 22: Message exchange during (re)configuration of the gateway.

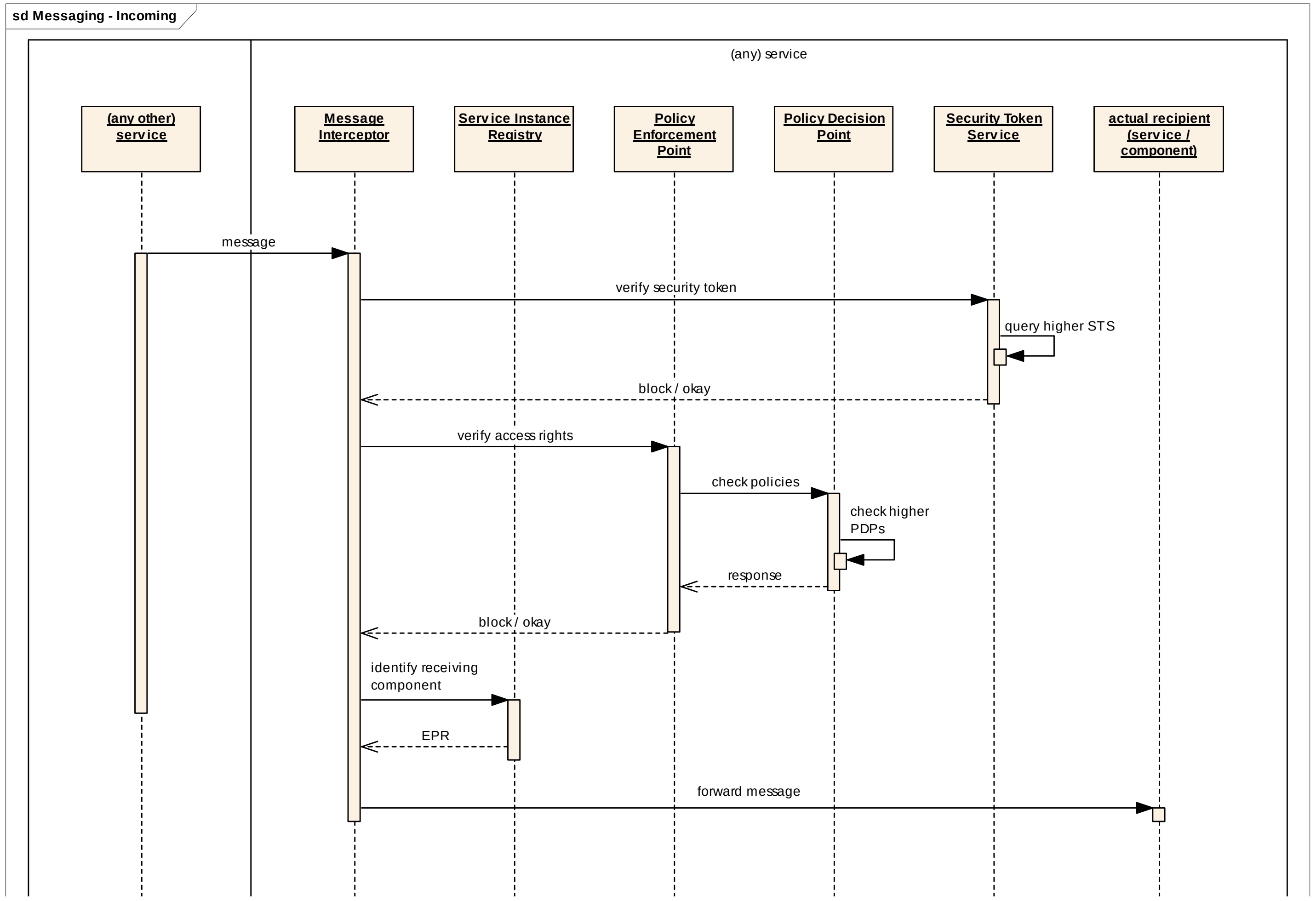


Figure 23: Message interception for incoming messages.

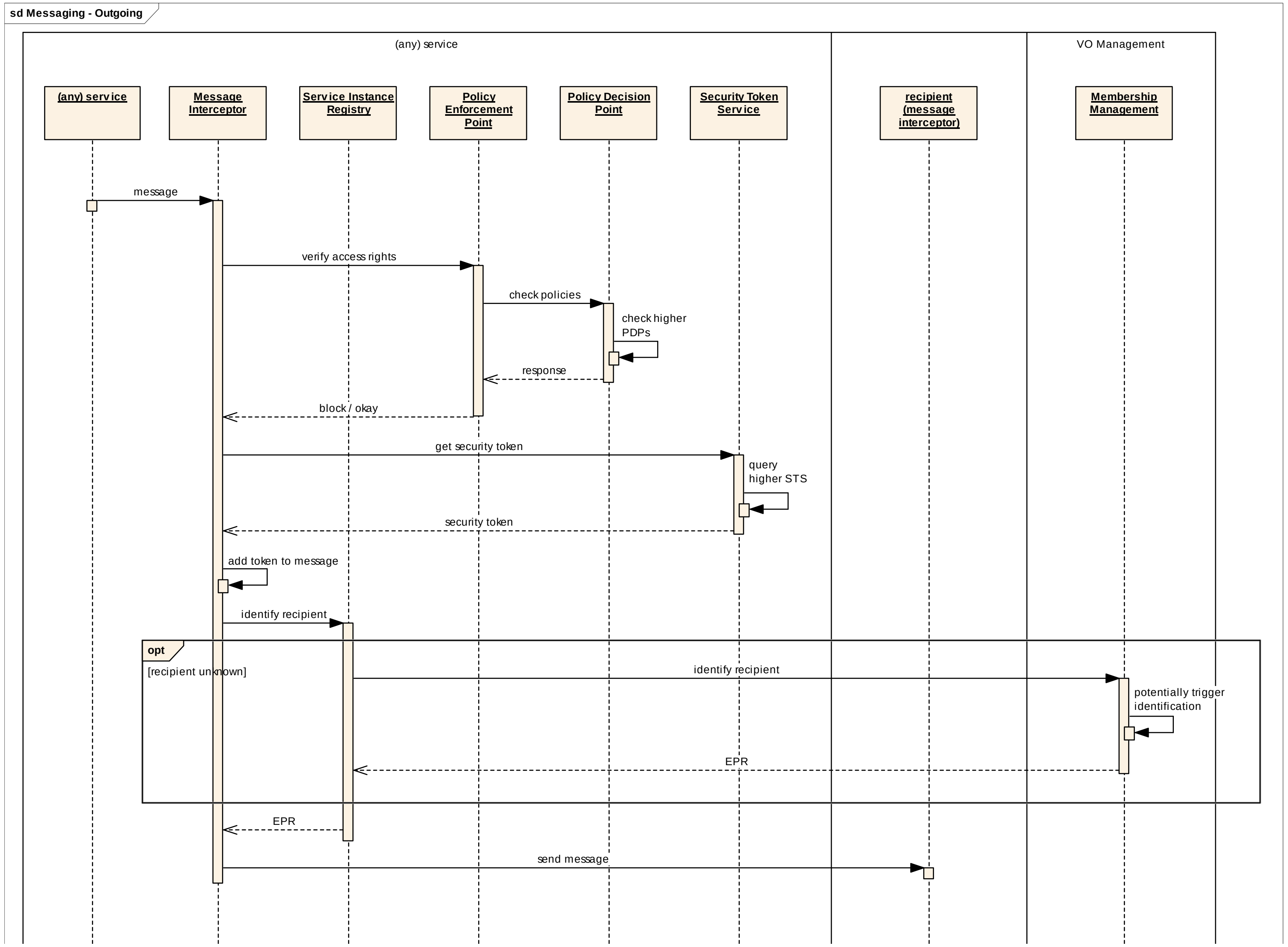


Figure 24: Message interception for outgoing messages.

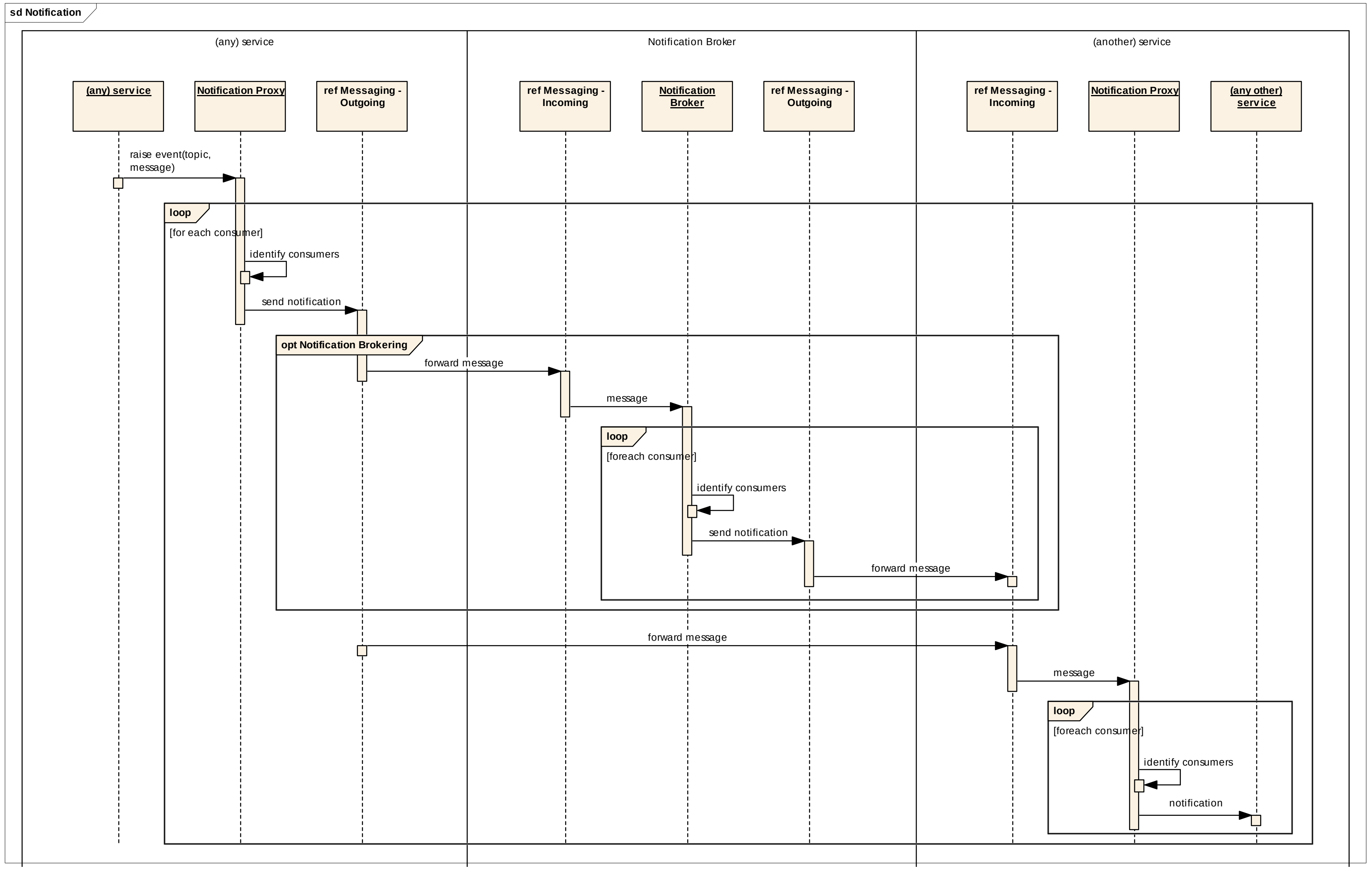


Figure 25: Notification specific interactions.

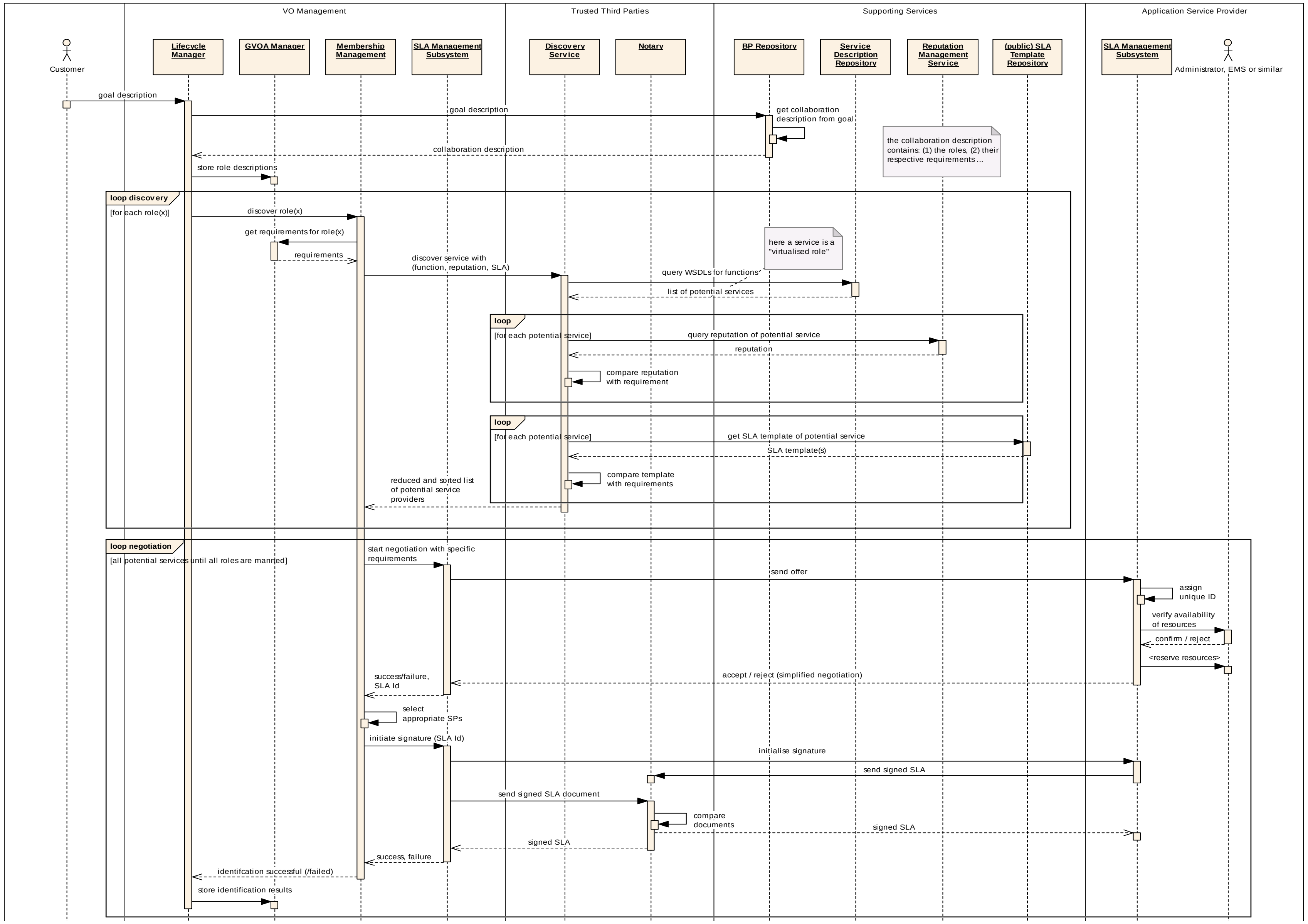

VO Management
Trusted Third Parties
Supporting Services
Application Service Provider
Customer
Lifecycle Manager
GVOA Manager
Membership Management
SLA Management Subsystem
Discovery Service
Notary
BP Repository
Service Description Repository
Reputation Management Service
(public) SLA Template Repository
SLA Management Subsystem
Administrator, EMS or similar
goal description
goal description
get collaboration description from goal
the collaboration description contains: (1) the roles, (2) their respective requirements ...
collaboration description
store role descriptions
loop discovery
[for each role(x)]
discover role(x)
get requirements for role(x)
requirements
here a service is a "virtualised role"
discover service with (function, reputation, SLA)
query WSDLs for functions
list of potential services
loop
[for each potential service]
query reputation of potential service
reputation
compare reputation with requirement
loop
[for each potential service]
get SLA template of potential service
SLA template(s)
compare template with requirements
reduced and sorted list of potential service providers
loop negotiation
[all potential services until all roles are manned]
start negotiation with specific requirements
send offer
assign unique ID
verify availability of resources
confirm / reject
<reserve resources>
accept / reject (simplified negotiation)
success/failure, SLA Id
select appropriate SPs
initiate signature (SLA Id)
initialise signature
send signed SLA
send signed SLA document
compare documents
signed SLA
signed SLA
success, failure
identifcation successful (/failed)
store identification results

Figure 26: Interactions during Identification.

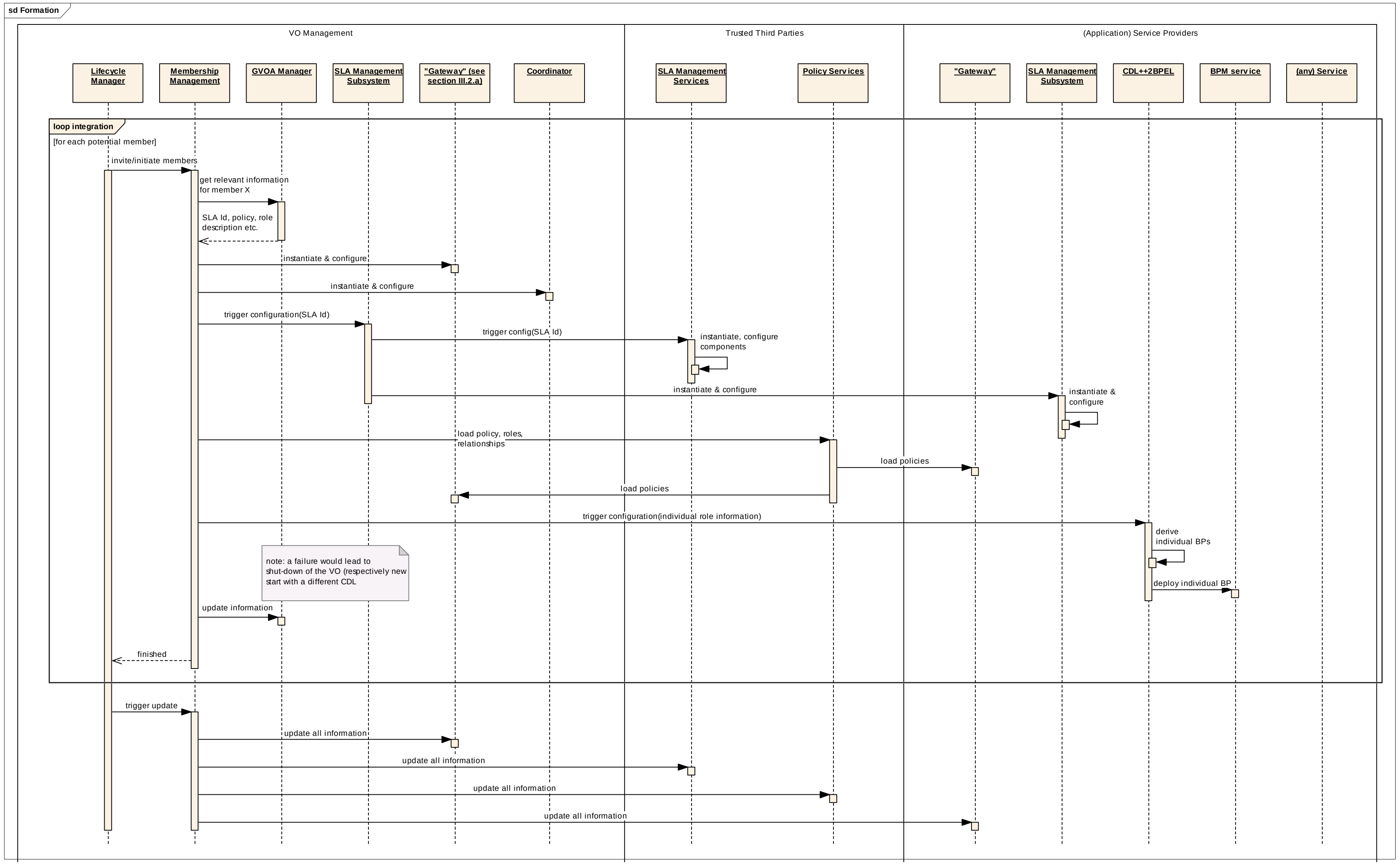


Figure 27: Interactions during Formation.

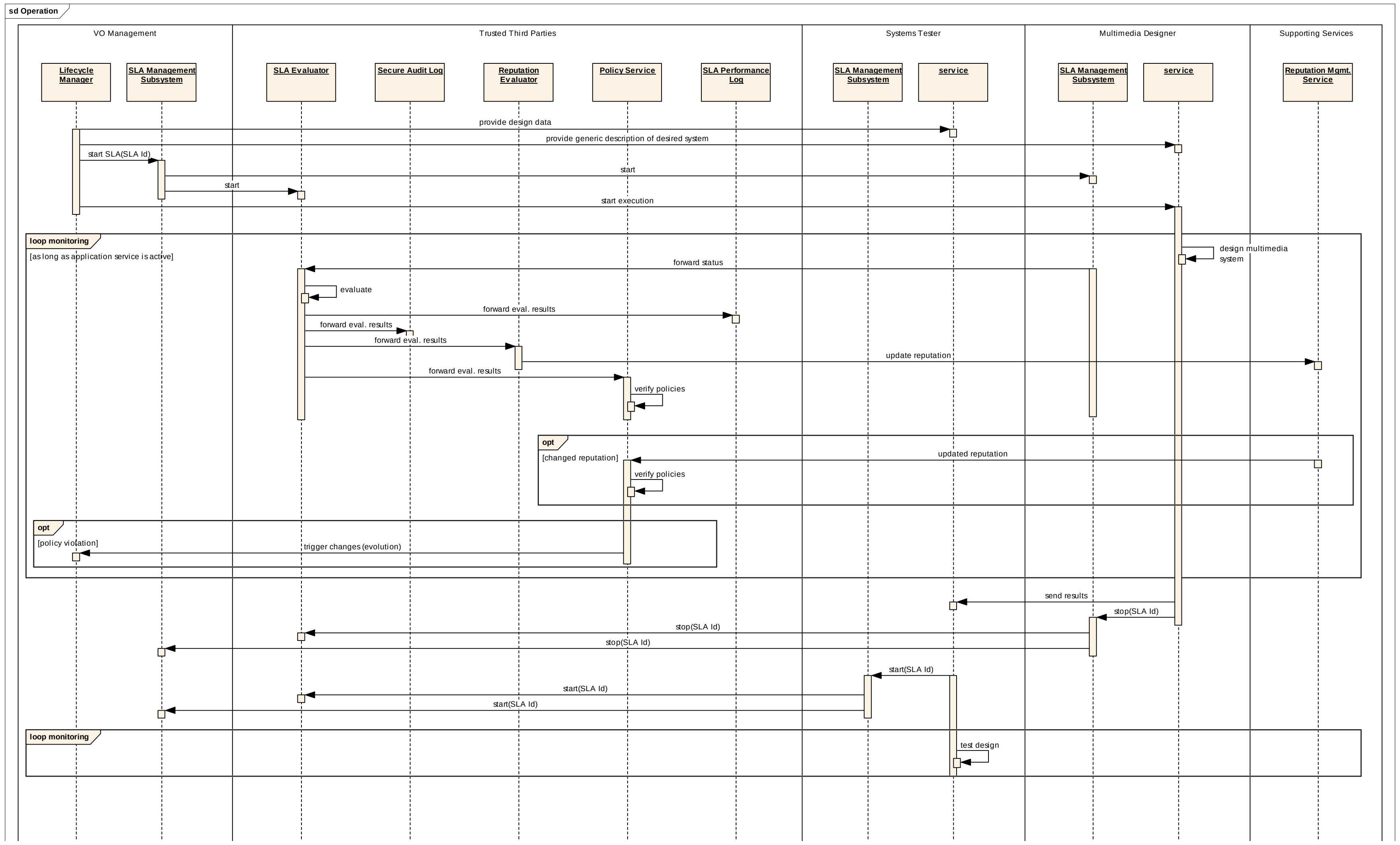


Figure 28: Typical interactions during Operation.

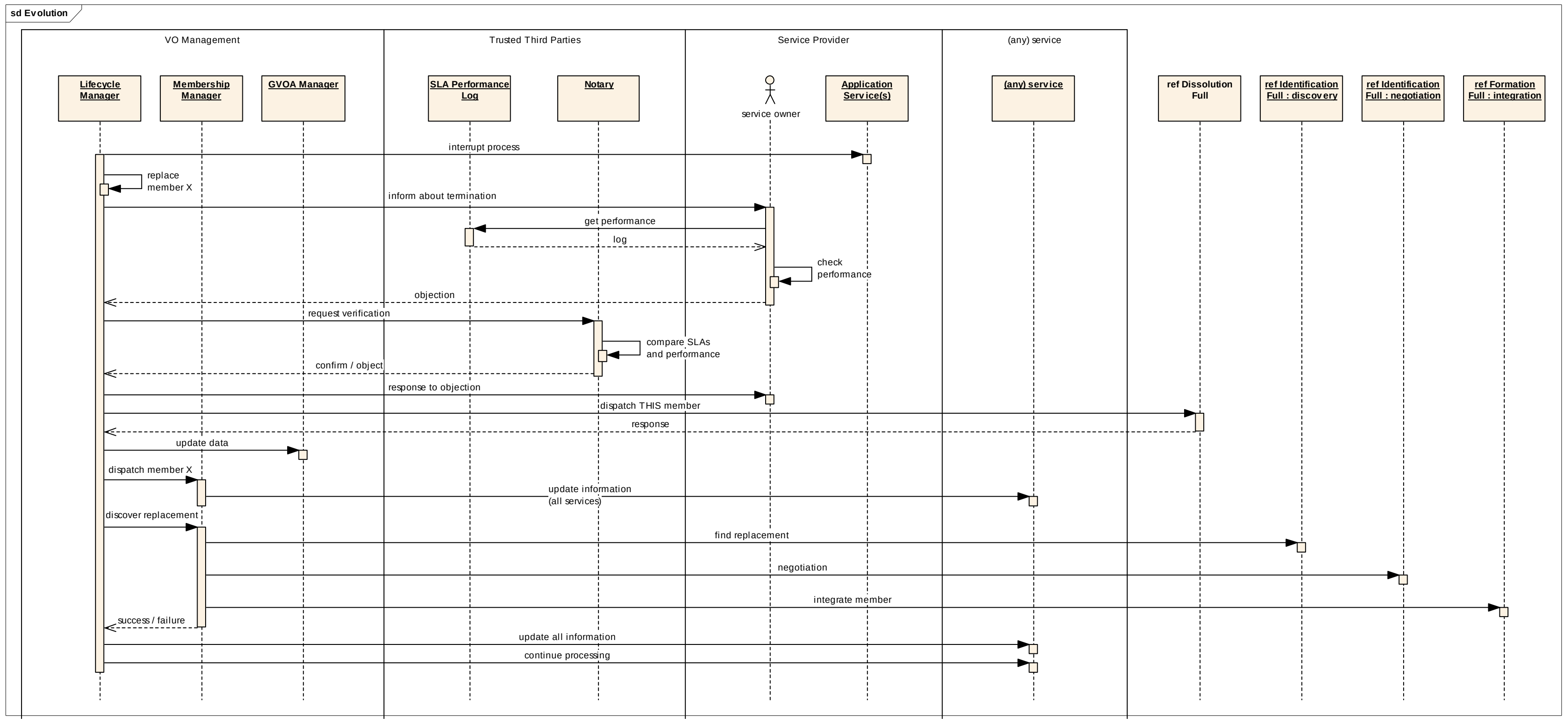


Figure 29: Overview over the Interactions during Evolution.

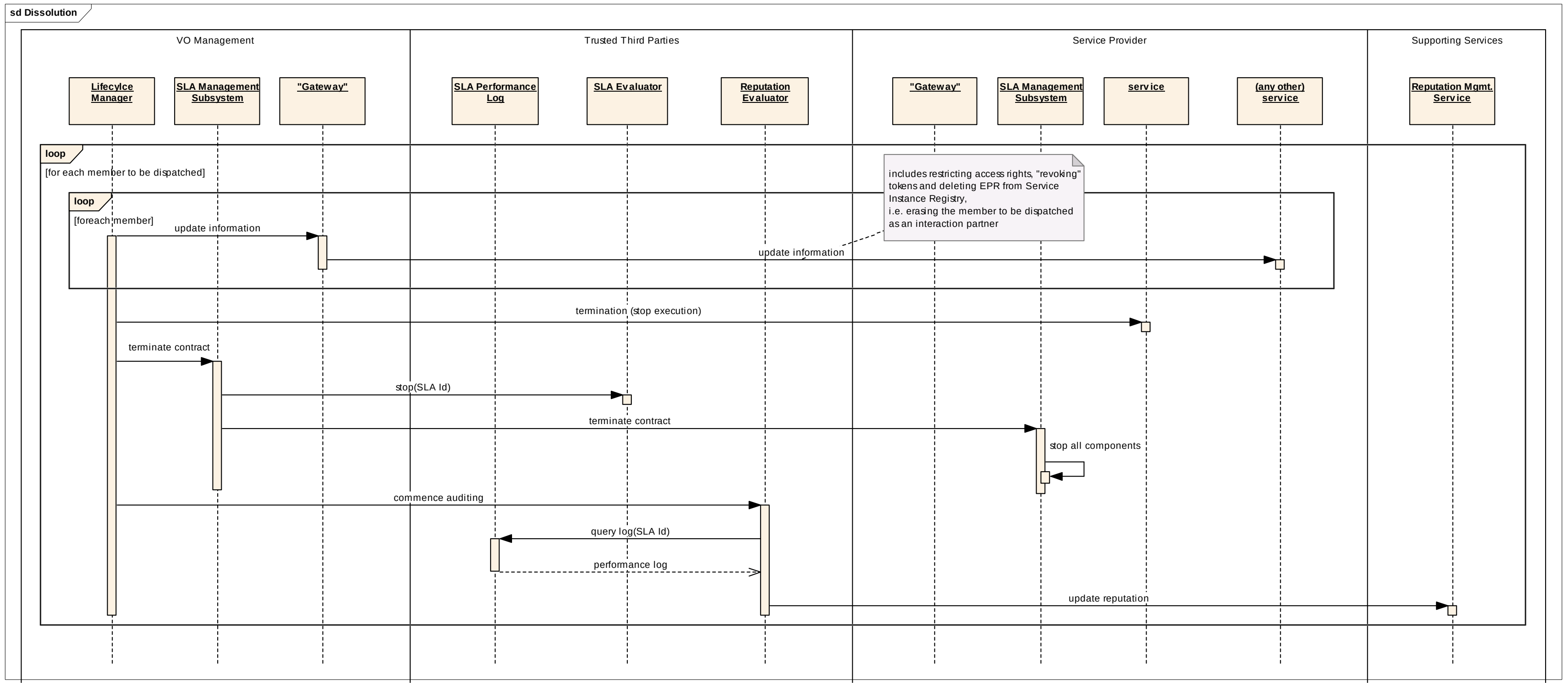


Figure 30: Overview over the Interactions during Dissolution.

IV Profiles EMIC, CCLRC

TrustCoM will focus on, and expects to have its main impact with respect to standardisation in the creation of profiles. A profile identifies how different specifications should be used together to support complex applications. This specifically applies to (but is not limited to) interoperable web services. If individual web services standards are metaphorically seen as pieces of a jigsaw puzzle, that each capture some autonomous functionality, then profiles can be seen as recommended designs of jigsaws and “best practice” guidelines that support work towards implementing comprehensive and potentially complex business functions. Profiles are created in response to the ever-growing number of interrelated specifications, all at different version levels and different stages of development and adoption, and often with conflicting requirements. Profiles integrate and refine dominant web services standard specifications by resolving potential conflicts between them, constraining their extensibility options where necessary, and exploiting their complementarity and composability characteristics.

Specific emphasis goes to the potential of creating TrustCoM profiles that integrate existing standards within and across the different areas. The project will concentrate on integration profiles, bringing together the isolated subsystem developments; while we have refined the potential standardisation contributions within each specific TrustCoM research and development area, the most immediate result of the TrustCoM standardisation activity is expected to be in the integration of existing standards across the different areas.

The specification of the Trustcom Framework for implementation in software draws upon many open specifications for three reasons:

- to transparently show how it operates in order to build trust in it as a technology;
- to ease implementation by anybody who wishes to do so;
- to improve the probability that the technology will interoperate between a wide range of platforms.

Consequently, there are many combinations of open specifications that could be the subject of profiles. In order to have an impact, only a small set of specifications have been selected as the basis of profiles which are both likely to be adopted, and where the project has mature input resulting from significant experience. These are:

- WSLA – to revive the structural detail required to specify SLA’s lost in the development focus on WS-Agreement
- WS-Trust – to refine the interaction of WS-Trust with other specifications
- WS-CDL – to demonstrate the integration of choreography and orchestration as methods of co-ordination of distributed business processes.
- XACML – to introduce delegation into the security specification
- EDA-Policies – to refine the policy representation as used in Trustcom

Each of these will be described below as a proposal for wider adoption.

IV.1 WSLA SICS

IV.1.a Introduction

This section defines a profile for the use of the Web Service Level Agreement (WSLA) specification to describe service level agreements (SLAs) within a TrustCoM Virtual Organization.

Background

The SLA technology analysis performed in the state-of-the-art evaluation and the experience accumulated so far in the project has resulted in the selection of WSLA as the main SLA specification formalism to be used within TrustCoM. Although the specification seems at times over-engineered, it is rich enough to match the needs of SLAs within the TrustCoM framework, naturally matching the distribution of tasks and responsibilities in the SLA Management subsystem as discussed in D9 (Architecture Deliverable).

An alternative specification, WS-Agreement, has so far been perceived as lacking enough structure (possibly resulting from an excessive zeal for generality). Nonetheless, future work regarding the creation and deployment of agreements within the TrustCoM Framework will be much influenced by the work on WS-Agreements.

Two design decisions affect considerably the way the WSLA is used in the Framework. First, loosely coupled components are to make as much use as possible of the notification mechanisms supported by the EN/VO infrastructure. An agreement must therefore be explicit about the way these mechanisms are to be used during SLA management. Second, application and supporting services are to be virtualized as VO Resources before they can be shared in a VO. The virtualization mechanisms, also provided by the EN/VO infrastructure, introduce a level of indirection in the representation of service addresses which has to be taken into account when describing services and their QoS requirements/guarantees.

Summary

The *Web Service Level Agreement specification* (WSLA) from IBM has been chosen as service level agreements description language. WSLA has a rich set of elements that is suitable for describing the distribution of tasks and responsibilities in the SLA Management subsystem although some minor modifications have been necessary.

Namespaces

For this profile, the namespace prefixes are defined as follows:

```
xmlns:wsla=”http://www.ibm.com/wsla”
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:uddi=”urn:uddi-org:api_v3”
xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xmlns:slaEval=”http://www.sics.se/Trustcom/SLAEvaluator”
```

Scope

The present version addresses only full SLAs. Neither SLA Templates nor WSLA SDI (Service Deployment Information) are covered by this profile.

SLA Identification

An SLA is identified by an URI of type `xsd:anyURI` returned by the SLA Manager when called to configure the monitoring service. The identifier attribute `name` of the root element of WSLA service level agreement, `wsla:SLA`, is not used.

Parties

All party descriptions in WSLA are instances of a subtype of the abstract type `wsla:PartyType` and therefore contain the sub-elements `<Contact>` and `<Action>` and the attribute `name`. Neither of the two elements serves the purpose of properly identifying the party as a VO partner. The name attribute should be used for such a purpose and therefore it should be of the type reserved by TrustCoM for VO Partner IDs, which is the UDDI Business Entity identifier type `uddi:businessKey`[17]. This is a change of the WSLA specification where the original type was `xsd:string`.

```
          <xsd:complexType name="PartyType" abstract="true">
               <xsd:sequence>
                    <xsd:element name="Contact"
                                type="wsla:ContactInformationType"
                                minOccurs="0"/>
                    <xsd:element name="Action"
                                type="wsla:ActionDescriptionType"
                                minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
               <xsd:attribute name="name" type="uddi:businessKey"/>
          </xsd:complexType>
```

It has been considered, but so far ruled as unnecessary, to add a further attribute to the `wsla:PartyType` corresponding to the handle of the manageability interface implemented by the party's *VOVirtualizationNode* (cf. EN/VO Infrastructure). This handle should be recovered from an index service provided by the EN/VO infrastructure.

Signatory Parties

In WSLA, an SLA relates the consumer to the provider of a service. These two parties are expected to sign the agreement and therefore are described with elements of type `wsla:SignatoryPartyType`, an extension of `wsla:PartyType`.

In the TrustCoM Framework, the identity of a signatory party is used to assign responsibilities, but in principle it is not expected that it directly implements any SLA Management action. Therefore this profile discourages the use of the `<Action>` element for signatory parties.

Supporting Parties

The WSLA specification defines three types of supporting parties (i.e. contributors to the execution of the SLA that are neither consumers nor providers): MeasurementService, ConditionEvaluationService and ManagementService.

Measurement Service. Corresponds both to the simple and aggregating monitors in the conceptual model (cf. deliverable D16 – Conceptual Models).

[17] http://uddi.org/schema/uddi_v3.xsd

1. When a `<Metric>` element of type `wsla:MetricType` contains a `<MeasuringDirective>` sub-element, then its <Source> sub-element specifies the id of a Simple Monitor. The following example tells us that supporting party “YMeasurement” is responsible for producing the integer value of a MeasurementDirective of type tc:StatusRequest (to be defined).

Example

```
          <Metric name=”MeasuredStatus” type=”integer” unit=””>
               <Source>YMeasurement</Source>
               <MeasurementDirective xsi:type=”tc:StatusRequest”
                                     resultType=”integer”>
                    <EndpointReference>
                         …An element of type wsa:EndpointReference…
                    </EndpointReference>
               </MeasurementDirective>
          </Metric>
```

Observe that the directive identifies the endpoint reference (EPR) that shall be used to query this monitor. The EPR may point to the monitoring interface implemented by a VO Virtual Node.

2. When a `<Metric>` element of type `wsla:MetricType` contains a `<Function>` sub-element, then its <Source> sub-element specifies the id of an Aggregating Monitor. The following example tells us that supporting party “HLRS2” is responsible for producing the double integer value that results of dividing the result of metric clock_speed by (100 – process_Cpu_Load).

Example

```
          <Metric name="performance_metric" type="double" unit="GHz">
               <Source>HLRS2</Source>
               <Function xsi:type="wsla:Divide" resultType="double">
                    <Operand>
                         <Metric>clock_Speed</Metric>
                    </Operand>
                    <Operand>
                         <Function xsi:type="wsla:Minus" resultType="double">
                              <Operand>
                                   <LongScalar>100</LongScalar>
                              </Operand>
                              <Operand>
                                   <Metric>process_Cpu_Load</Metric>
                              </Operand>
                         </Function>
                    </Operand>
               </Function>
          </Metric>
```

Observe that the example does not clarify how party HLRS2 is expected to make the metric value available to other SLAM components. In the case of the Condition Evaluation service this is done by the use of element `<SLAParameter>` as explained below.

3. Some Monitors (i.e. MeasurementServices) are also producers of SLA Parameters. A `SLAParameter` contains based on a sub-element `SLAParameter/Metric` indicating how the parameter is defined. The party responsible for providing the parameter is indicated by the sub-element `SLAParameter//Source`. The sub-element `SLAParameter//Pull` may be used to define which parties are allowed to pull the SLA Parameter from the monitor (by invoking its `GetSLAParameterValue` operation). The sub-element `SLAParameter//Push`, if not empty, indicates that:

1. The Monitor is expected to produce notifications according to the metric schedule.
2. The SLA Management subsystem shall subscribe the list of parties in the <`Push`> element to receive those notifications.
3. A Monitor publishes notifications using the simple topic dialect for topics. A topic is formed in the following way in order to identify a SLA Parameter uniquely:

   { *value of* `SLAParameter//`<`Source`> } *value of* `SLAParameter/@name`

   For instance, if element `SLAParameter//`<`Source`> has value "`http:/example.com:monitor1`" (a URI) and the SLA parameter attribute `name` has value "`averageResponseTime`" then the topic is: "{http:/example.com:monitor1}averageResponseTime".
4. A notification shall contain a message of type `slaEval:SLAParameterWrapperType` that contains element `ArrayOfSLAParameters`, which in turn contains an array of `SLAParameter` elements of type `slaEval:SLAParameterType.` The type `slaEval:SLAParameterType` is based on type `wsla:SLAParameterType` in order to be compatible with the WSLA specification. Thus it has element `value` of type `xsd:double` and has attribute `name` of type `xsd:string`, attribute `type` of type `xsd:string` and attribute `unit` of type `xsd:string.`

Condition Evaluation Service. There is a one to one relationship between this type of supporting party and SLAEvaluators (see D16 – Conceptual Models).

An SLAEvaluator is responsible for notifying the violation of a set of service level objectives (see

Obligations)

Management Service. The approach of the TrustCoM Framework establishes an SLA Management infrastructure as part of the constitution of the VO, regulated by the GVOA, so it is unnatural to let specific SLAs define how they are to be managed. For this reason, the present profile deprecates the use of ManagementServices.

Obligations

ActionGuarantees shall only be used to encode obligations on the SLAEvaluator and the Monitors.

An SLAEvaluator is responsible for the evaluation of a Service Level Objective (SLO) if it is listed in `/SLA/Obligations/ActionGuarantee/Obliged`. In such a case, the SLO in question is given in the `Expression//ServiceLevelObjective` sub-element of the `ActionGuarantee`.

Example

```
          <Obligations>
               <ServiceLevelObjective name=”g1”>
                    …
               </ServiceLevelObjective>
               <ActionGuarantee name="g2">
                    <Obliged>SICS</Obliged>
                    <Expression>
                         <Predicate xsi:type="wsla:Violation">
                              <ServiceLevelObjective>g1</ServiceLevelObjective>
                         </Predicate>
                    </Expression>
                    <EvaluationEvent>NewValue</EvaluationEvent>
                    </QualifiedAction>
                    <ExecutionModality>Always</ExecutionModality>
               </ActionGuarantee>
          </Obligations>
```

In this example, party SICS provides an SLAEvaluator that computes SLO g1. Whenever this SLO is violated, the SLAEvaluator is obliged to send a notification. The notification message is published with a simple topic "{http://wsrf.notification.de}SLA_violation". The message contains the identifiers of the violating SLA Parameters, the identifier of the SLO (qualifying the SLO name using the SLA name) and the violating partner (see discussion on topics for notifications emitted by MeasurementServices).

In case the SLO is fulfilled, a similar message is sent with topic "{http://wsrf.notification.de}SLA_fulfilment".

IV.2 WS-Trust & SAML EMIC, ETH & UoK

This section describes a WS-Trust and a SAML token profile for virtual organizations as implemented in the FP6 TrustCoM project. The purpose of this document is to specify how web service components communicate with security token services (STS) to request an STS to issue and validate ‘cross-organizational’ security tokens.

This chapter does not intend to present a final profile, nor does it intend to present a mandatory profile for use outside the “scoped federations” context as implemented in the FP6 TrustCoM project.

We differentiate between two types of security tokens:

- Organization-internal security tokens and

- Cross-organizational security tokens.

This differentiation is necessary because each VO partner organization may use arbitrary security tokens inside the organization’s own network, so that a standardization and unification of these types is not possible. For example, one VO partner organization may solely use Kerberos tokens to authenticate and protect messages inside the company’s network, whereas other VO partners may use username/password or X.509 certificates inside their organization. Even in scenarios where all organizations use long-term tokens such as X.509 certificates, it may not be possible to use these tokens cross-organizationally, because the companies may not have a common root of trust (e.g., no X.509 cross-certification).

For the above reasons, it is necessary to agree on a common format of cross-organizational security tokens. Inside TrustCoM, we agreed to use SAML assertions as security tokens.

The objective is to draft a profile in which all the parameters are clearly justified, and correspond to a concept from the framework. The current draft is not fully there yet, primarily because the profile originally suggested uses symmetric encryption (whereas other SAML profiles with which this should be consistent use asymmetric encryption) and it also contained parameters whose purpose is not immediately obvious.

It is important that it can be clearly seen how the profiles and their parameters fit the framework, and what the relationship between profile parameters and framework elements/concepts are.

Here is a summary of what has been agreed so far:

i. The WS-Trust profile will send SAML attribute assertions

ii. The parameters to be used from SAML attribute assertions are

a. the issuer field is mandatory and contains the name of the issuer of the attribute assertion/security token.

b. advice is optional and probably wont be used

c. the signature field is optional and isnt needed when X.509 ACs are passed as the attributes, or when symmetric encryption is used

d. conditions are optional, but when present will contain the validity time of the attribute assertions (notBefore and notOnOrAfter)

e. the subject statement holds the name of the entity that the attributes are being assigned to

f. the set of attributes contain the attributes being assigned to the subject

Whereas the following issues are still outstanding and not agreed so far:

i. how symmetric tokens and tickets are encoded

ii. how obligations are encoded

iii. how delegation permission is encoded

iv. how "no assertion" is incoded

These will be addressed in the next development cycle of six months before the next release of V3 of this Framework.

IV.2.a Namespaces and supported specifications

Inside this document, the namespace prefixes are defined as follows:

```
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"

xmlns:emic="http://www.microsoft.com/emic/SAFe/"
xmlns:emicfrc="http://www.microsoft.com/emic/SAFe/#FederationRestrictions"
xmlns:emicfpi="http://www.microsoft.com/emic/SAFe/#FederationPartners"
xmlns:wstx="http://www.microsoft.com/emic/SAFe/#WSTrustExtensions"
```

IV.2.b WS-Trust

This profile is based on the WS-Trust specification from February 2005 (http://msdn.microsoft.com/ws/2005/02/ws-trust/).

Issuance Binding Profile

For requesting a new cross-organizational security token, we use the "Issuance Binding" as defined by the WS-Trust specification from February 2005.

- wst:TokenType

The WS-Trust token type for cross-organizational SAML assertions is defined as follows:

```
<wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1</wst:TokenType>
```

- wsp:AppliesTo

In this profile, the requestor of a security token MUST specify a wsp:AppliesTo element as part of the wst:RequestSecurityToken. This element may have the following components:

wsp:AppliesTo/wsa:EndpointReference/wsa:Address (MAY)

> The URI of the web service where the token will be used.

wsp:AppliesTo/wsa:EndpointReference/wsa:Action (MAY)

> The action that is invoked on the web service where the token will be used.

wsp:AppliesTo/wsa:EndpointReference/wsa:ReferenceProperties/emic:FederationUUID (MUST)

> The FederationUUID is an identifier of the VO inside which the token will be used. We expect that an issue request for a cross-organizational token MUST contain a VO identifier (such as a FederationUUID). That is necessary because the STS must be able to lookup whether the requesting client has available claims for that particular VO.
>
> In this profile document, we defined an own format for a VO identifier. The model would allow to use other types of identifiers with equivalent functionality, for example from UDDI space.

wsp:AppliesTo/wsa:EndpointReference/wsa:ReferenceProperties/emicfpi:FederationPartnerIdentifier (SHOULD)

> That federation partner identifier is an identifier of the VO partner organization that performs token validation for the service. Such a partner identifier could be a long-term credential of the partner's STS (such as an X.509 certificate or a reference to a certificate), a UDDI business entity key or some other unique identifier.
>
> The STS needs the federation partner identifier for different purposes: In a symmetric-key based (Kerberos-like) model, the STS requires that information to determine the service's organization's security token (key), so that the STS can include a session key inside the cross-organizational token. In addition, the partner identifier may be used for client-side security decisions.

- wst:RequestedSecurityToken

The wst:RequestedSecurityToken MUST contain a cross-organizational saml:Assertion element.

- wst:RequestedProofToken

The wst:RequestedProofToken SHOULD contain the private or secret key material associated with the saml:Assertion. In the current "scoped federations" prototype, the wst:RequestedProofToken contains an xenc:EncryptedKey element. The xenc:EncryptedKey contains a symmetric key encrypted for the requestor of the token, i.e., the key is encrypted under the client's organization-internal key.

Validation Binding Profile

The current prototype adopts the WS-Federation "U-model". To validate an existing cross-organizational security token at the service side, we use the "Validation Binding" as defined by the WS-Trust specification.

- wst:TokenType

The WS-Trust token type for validation SAML assertions is defined as follows:

```
<wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1</wst:TokenType>
```

- wsp:AppliesTo

See Issuance Binding Profile. For token validation, the service MUST provide wsa:Address and wsa:Action elements in the wst:RequestSecurityToken/wsp:AppliesTo.

- wstx:ValidateTarget

The wstx:ValidateTarget element refers to the target of validation. The wstx:ValidateTarget element MUST contain the cross-organizational saml:Assertion that should be validated.

- wst:RequestedSecurityToken

The wst:RequestedSecurityToken MUST contain a saml:Assertion that contains the validation results.

- wst:RequestedProofToken

The wst:RequestedProofToken SHOULD contain the public or secret key material with which the service can verify the signature of the received message as well as decrypt the received message. In the current "scoped federations" prototype, the wst:RequestedProofToken contains an xenc:EncryptedKey element. The xenc:EncryptedKey contains the symmetric key associated with the SAML token, now re-encrypted for the service.

- wst:Status

The wst:Status element MUST be included in the RSTR as specified by WS-Trust. The predefined URIs, as specified in WS-Trust, are used in the current prototype.

IV.2.c SAML Assertion Profile

This profile is based on the SAML 1.1 Assertion specification (http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf) and the Web Services Security SAML Token Profile 1.1 (http://www.oasis-open.org/committees/download.php/15256/Web%20Services%20Security%20SAML%20Token%20Profile-11.pdf).

SAML cross-organizational token

The cross-organizational security token is a SAML 1.1 saml:Assertion. The saml:Assertion MUST include saml:Conditions, saml:AttributeStatement, and ds:Signature elements.

- saml:Assertion

The @Issuer attribute SHOULD contain the URI of the issuing STS.

- saml:Conditions

In addition to the @NotBefore and @NotOnOrAfter attributes which MUST be included, the saml:Conditions element MUST include a emicfrc:FederationRestrictionCondition.

emicfrc:FederationRestrictionCondition (MUST)

> The FederationRestrictionCondition defines the federation scope in which the cross-organizational SAML assertion can be used. Validation in other scopes must fail.

- saml:AttributeStatement

The saml:Assertion MUST contain exactly one saml:AttributeStatement. That saml:AttributeStatement element MUST contain one saml:Subject and a saml:Attribute element.

saml:Subject (MUST)

> The subject is the owner of the token and is identified by a saml:SubjectConfirmation/saml:ConfirmationMethod urn:oasis:names:tc:SAML:1.0:cm:holder-of-key as specified in the WSS SAML Token Profile 1.1.

> The key is included in a ds:KeyInfo element which contains an xenc:EncryptedKey with a symmetric key encrypted for the receiving VO partner organization.

saml:Attribute Claims (MUST)

The AttributeName is “Claims” and the AttributeNamespace is "http://schemas.xmlsoap.org/ws/2005/02/trust". The saml:AttributeValue element MUST contain a wst:Claims element.

The wst:Claims element contains the claims that the client possesses in the particular VO. These claims may be xacml11 attributes.

- ds:Signature

The Signature MUST contain exactly one ds:Reference referencing the saml:Assertion/@AssertionID attribute. This Reference MUST have exactly two transforms:

1. The first transform is “Enveloped Signature” (http://www.w3.org/2000/09/xmldsig#enveloped-signature)
2. The second transform is “Exclusive XML Canonicalization without Comments” (http://www.w3.org/2001/10/xml-exc-c14n#)

To support cross-organizational validation of the signature of the token, the KeyInfo element MAY contain various references to the signing certificate of the issuing STS, including a wsse:SecurityTokenReference/wsse:KeyIdentifier, a wsse:SecurityTokenReference/wsse:Embedded, or a emicfpi:FederationPartnerIdentifier.

SAML validation token

The validation response is a SAML 1.1 saml:Assertion. The saml:Assertion MUST include saml:Conditions, saml:AttributeStatement, and ds:Signature elements. In addition, a saml:Advice SHOULD be included.

- saml:Assertion

The Issuer attribute SHOULD contain the URI of the validating STS.

- saml:Conditions

In addition to the NotBefore and NotOnOrAfter attributes which MUST be included, the saml:Conditions element MUST include a emicfrc:FederationRestrictionCondition.

emicfrc:FederationRestrictionCondition (MUST)

The federation scope in (and only in) which this SAML assertion is to be considered.

- saml:Advice

The saml:Advice SHOULD contain the original cross-organizational saml:Assertion that has been validated.

- saml:AttributeStatement

The saml:AttributeStatement element MUST include a saml:Subject and at least one saml:Attribute element.

saml:Subject (MUST)

The subject is the owner of the original cross-organizational token that is validated, and is identified by a saml:SubjectConfirmation/saml:ConfirmationMethod urn:oasis:names:tc:SAML:1.0:cm:holder-of-key as specified in the WSS SAML Token Profile 1.1.

The key is included in a ds:KeyInfo element which contains a wst:BinarySecret with a cleartext symmetric key (this assumes that the RSTR is properly protected!), or an xenc:EncryptedKey with a symmetric key encrypted for the receiving VO partner organization.

saml:Attribute FederationPartnerIdentifier (MAY)

The AttributeName is “FederationPartnerIdentifier” and the AttributeNamespace is "http://www.microsoft.com/emic/SAFe/#FederationPartners". This attribute MAY be included to explicitly indicate to the service the VO partner organization the service request is originating from. If this attribute is present, the saml:AttributeValue element MUST contain an emicfpi:FederationPartnerIdentifier element.

saml:Attribute Status (MUST)

The AttributeName is “Status” and the AttributeNamespace is "http://schemas.xmlsoap.org/ws/2005/02/trust". This attribute MUST be included to indicate the result of the security token validation. The saml:AttributeValue element MUST contain a wst:Status with one of the predefined wst:Code status codes.

saml:Attribute Claims (SHOULD)

The AttributeName is "Claims" and the AttributeNamespace is "http://schemas.xmlsoap.org/ws/2005/02/trust". This attribute SHOULD be included to pass the validated (and possibly transformed) claims to the service. If this attribute is present, the saml:AttributeValue element MUST contain a wst:Claims element.

A policy enforcement point (PEP) may forward these validated claims to a policy decision point (PDP) to support the policy decision.

saml:Attribute ValidationMessage (MAY)

The AttributeName is "ValidationMessage" and the AttributeNamespace is "urn:string". This attribute MAY be included to pass a human-readable validation result message to the service.

IV.2.d Custom elements

The following custom namespace prefixes are defined in the current “scoped federations” prototype in TrustCoM:

```
xmlns:emic="http://www.microsoft.com/emic/SAFe/"
xmlns:emicfpi="http://www.microsoft.com/emic/SAFe/#FederationPartners"
xmlns:emicfrc="http://www.microsoft.com/emic/SAFe/#FederationRestrictions"
```

- emic:FederationUUID

The emic:FederationUUID represents a universal and unique identifier for the federation scope.

- emicfpi:FederationPartnerIdentifier

The emic:FederationPartnerIdentifier identifies a VO partner organization. A partner organization can be identified in various ways as indicated in the Type attribute.

- X509SubjectName Type

X509Data/X509SubjectName (MUST)

    The X.509 DN of the certificate of the issuing STS of the partner.

- emicfrc:FederationRestrictionCondition

The emicfrc:FederationRestrictionCondition is a custom SAML condition which intends to indicate the “scope” within which the SAML cross-organizational or validation token MUST be considered.

- wsp:AppliesTo

The FederationRestrictionCondition MUST contain a wsp:AppliesTo element.

wsp:AppliesTo/wsa:EndpointReference/wsa:Address (SHOULD)

    The URI of the web service that is invoked.

wsp:AppliesTo/wsa:EndpointReference/wsa:Action (SHOULD)

    The action that is invoked on the web service.

wsp:AppliesTo/wsa:EndpointReference/wsa:ReferenceProperties/emic:FederationUUID (MUST)

The VO identifier inside which the assertion can be used.

Role semantics

In the first prototype, we used a self-defined role claim with proprietary semantics to represent roles. In TrustCoM, we will use XACML 1.1 attribute values to convey role information.

IV.3 WSCDL SAP

IV.3.a Overview

This section defines a profile for the use of W3C's Web Service Choreography Description Language (WS-CDL) specification to describe the business process modelling aspects of collaboration definitions within a TrustCoM Virtual Organization.

Background

The technology analysis on collaborative business processes performed in the state-of-the-art evaluation and the experience accumulated so far in the project has resulted in the selection of WS-CDL as the business process specification to be used for the holistic view on collaborative business processes within TrustCoM, i.e., the single view on the collaborative process that includes the activities at and interactions between all involved parties. Although much critique has been issued against the specification [10], [11], [12] and it is not yet a standard, the advantages over other available choreography models outweigh the issues. WS-CDL matches the needs for collaborative business processes within the TrustCoM framework, due to the following reasons: It specifies the control flow over interactions and local activities between multiple roles from a high-level perspective, and is conceptually close enough to single-party business process languages to be matched with them. A choreography language that allows for the modeling of complex interaction patterns would mostly be good for design, not for execution as a business process, because its execution should include executable business processes as well as more flexible programming models and human interaction, e.g., for distinctive choice points with a high economical impact.

Most other choreography languages state single-partner processes and connect them, at the cost of hard legibility and high risk of incoherency. WS-CDL always offers a combined view on all partners' activities, making it much easier to realize and observe coherence in the various parties' behavior

Alternative specifications include WSCI, WSCL, and BPSS:

- WSCI[18] is also a specification by W3C, which allows the definition of choreographies by extending WSDL interfaces to express business process semantics over the web

[18] WSCI: Web Service Choreography Interface

service operations and connecting such extended WSDLs to form a choreography. There are multiple points to note here: WSCI is more a web service technology than a business process technology. Its most natural use would be to connect existing web services, thus suggesting a bottom-up approach – instead of the here-anticipated top-down approach. The distribution of the choreography specification over multiple documents does not feature a global view on the collaborative business process as a whole, which is supposedly very helpful for consistence and coherency in the understanding of the overall control and data flow. Last, the level of detail is fairly high: on the choreography level, the exact WSDL interfaces of each partner do not yet have to be present.

- ebXML[19] is a business collaboration framework, which offers related mechanisms to specify collaborations between partners. The focus here is rather on the business level with its functional and legal implications and not process integration and execution.

Still, an ideal choreography language is not available yet. Potentially, ongoing and future developments will strongly influence the choice for a choreography language in future implementations of the TrustCoM architecture.

Summary

The *Web Service Choreography Description Language (WS-CDL)* [8] is the main effort of W3C's WS Choreography Working Group. Still being on the way of becoming a standard, it offers the most promising, currently available way to describe business processes for multiparty collaborations from a high-level perspective. Similar to an abstract BPEL process, a choreography in WS-CDL only describes the externally observable aspects of a collaborative business process. It is important to keep in mind that a choreography is not meant for execution, but resembles a design artifact.

Scope

In TrustCoM, WS-CDL is used to model the collaborative business process (CBP) spanning all members of a VO and describing the interplay of their local activities and communication during the operation phase of a VO. This description is given from the high-level perspective of the whole VO with an emphasis on interactions, omitting the details about internal implementations of business services. In other words: While many components in the TrustCoM framework deal with the administrative aspects of the cooperation between the VO members, the choreography describes the actual work to be performed by the VO and how the members align their efforts.

Due to the current usage of WS-CDL, which is to generate WS-CDL code from UML diagrams via the UML2CDL service, and to generate BPEL code from the CDL via CDL2BPEL, WS-CDL could in principle be replaced with moderate effort.

IV.3.b WS-CDL Language Elements and Representation

One or many choreographies form a cdl:package. Exactly one of them is marked as the "root choreography", and thus is the starting point for a package. Having its roots in the Pi-

[19] ebXML: Electronic Business using eXtensible Markup Language , see http://www.ebxml.org/

calculus, a choreography in CDL describes the control flow around basic activities through structuring activities. A choreography can have variables, exception handlers, and finalizers, which define communication and the like at the end of a choreography. Due to the point of view taken by CDL, there are only few basic activities, with the interaction as the center piece, since the focus of choreographies is to describe the how and when of communication. All basic activities, conditional expressions, and variables can be defined for only a subset (sometimes of size one) of the available roles.
In CDL, the concept used for referring to one of the parties is always the role type (or, in short, the role). A party that wants to participate in a choreography can be required to play multiple roles by specifying a cdl:participant subsuming these roles. Note that each role can belong to zero or one participant. Also, a role can be defined to show more than one behavior. Each behavior can be refined in a WSDL document, and, if it is not, has no deeper meaning for the details of the choreography. However, both, a WSDL and a CDL document, describe the behavioral interface of entities, although a choreography includes far more information. Thus, our impression is that the redundancy in providing an additional WSDL per role behavior alongside with a choreography yields no significant advantage. Note, that CDL has a closed-world assumption, meaning that interactions are always bilateral between two roles specified in the choreography.

Example

```
      <roleType name="AnalysisPartner">
            <behavior name="Analyzer"/>
      </roleType>
      <roleType name="StoragePartner">
            <behavior name="StorageProvider"/>
      </roleType>
```

In the above code snippet from a WS-CDL package in Collaborative Engineering, two role types are defined: the *AnalysisPartner* and the *StoragePartner*, each showing a single behavior with no assigned WSDL interface. The choreography corresponds to the UML Activity Diagram in Figure 31.

WS-CDL Activity Elements

Starting with the structuring ones, the list of activities is shown below.

- **sequence** - Sequential order of activities.
- **parallel** - Parallel execution of activities.
- **workUnit** - As the most unusual structuring activity, the workUnit specifies conditions under which an enclosed activity is executed or repeated. Its guard condition is similar to an if-condition in standard programming languages, and can contain various XPath expressions or CDL supplied functions. The guard can be evaluated either immediately or deferred (e.g., when a variable becomes available) by setting the block attribute to true or false, respectively. Furthermore, the repeat condition states if a workUnit is considered for execution again after completion.
- **choice** - Exclusive branching: at most one of the enclosed activities (which may itself be a structuring activity) is to be performed. A cdl:choice is intended to contain workUnits as children, with a guard condition. If there are non-workUnit children in a

choice, the branching condition is said to be non-observable or not relevant at the choreography.

- **interaction** - Used for communication between two roles. In data exchanges the submitted variables are specified. Timeout conditions can be defined directly in an interaction, as well as assignments with reference to the data exchanges. If an interaction's "align" attribute is true, transactionality for an interaction is enabled, in the sense that the interaction only shows effect if the involved roles have the mutual understanding that the interaction completed successfully.

  Example

  ```
  <interaction name="getRawDataReq" operation="getRawDataOp"
    channelVariable="chVarGet">
    <participate relationshipType="AnalysisStorageRel"
      fromRoleTypeRef="AnalysisPartner"
      toRoleTypeRef="StoragePartner"/>
    <exchange name="exRawDataAddr" informationType="uriType"
      action="request">
      <send variable="cdl:getVariable('varRawDataAddr_Ana','','')"/>
      <receive variable="cdl:getVariable('varRawDataAddr_Sto','','')"/>
    </exchange>
  </interaction>
  ```

  This code example shows the information exchange between the AnalysisPartner (AP) and StoragePartner (SP) from the first CDL code example. The Web service operation 'getRawDataReq' at SP is called by AP. The information exchanged is the raw data's address, available in the variable 'varRawDataAddr_Ana' at the AP and stored in the variable 'varRawDataAddr_Sto' at SP after the transmission.

- **noAction** - Explicit "no operation" for a specified role. The respective party must remain idle.
- **silentAction** - Partner-internal action, whose details are of no interest to the choreography as a whole. The comment, by default in natural language, specifies what a partner is assumed to do at that instant, e.g., "analysis of aircraft antenna".
- **assign** - Variable value modification. Can be used to trigger exceptions.
- **perform** - Execution of another choreography. With CDL's binding mechanism, variable values from the outer choreography can be carried over to the inner choreography.

The link between WS-CDL and the Pi-calculus is strong, and also becomes apparent in the availability of channels in CDL. There, channel variables are of a channel type, which allows the definition of identity and reference tokens, restrictions on the channel usage, and the receiving role at the end of a channel. However, the way channels can be used in CDL as well as certain activities and more allow for several points of critique. This critique is subject to [10], [11], [12] and shortly summarized below.

Graphical Notation

UML activity diagrams offer a good visualization for choreographies, as justified in [9]. Where common business process modeling languages deal with only one party per process, in a choreography there are always multiple roles. The distinction between activities of the various roles is achieved by using a swim-lane (large, rectangular boxes) per partner. In contrast to WS-CDL, UML activity diagrams do not know a single activity for the interaction as a whole, so each cdl:interaction is represented by a pair of send and receive activities.

Summary of Critique Against WS-CDL

The main points of critique in [10] (p.16-18) are: the not explicitly stated link to a formalism as the Pi-calculus on the one hand, and the conceptual limits of linking WS-CDL to WSDL, WSDL-MEPs[20], and WSBPEL on the other hand; the not anticipated runtime selection of participants; the restriction to binary interactions; the dissimilarity of the sets of control flow constructs of WS-CDL and WSBPEL with respect to the fact that WSBPEL is the most promising orchestration language; and the discrepancy of WS-CDL being a design-level language and having no graphical representation. These are all very good points and - since they are deeply positioned in the concepts of the language - question the future of WS-CDL as a whole.

In WS-CDL, communication (cdl:interaction) is always bilateral, and built-in transactionality is restricted to the guaranteed mutual agreement of single variable values at one point in time. Therefore, WS-CDL most likely is unable to express the majority of the 15 "Service Interaction Patterns" from [11]. It thus seems not suitable for modeling related use cases, like a broad request for proposals with unknown outcome.

Also, the redundancy in certain WS-CDL elements makes writing a choreography with a general-purpose editor inconvenient. For instance, an attribute whose content has to be a variable, still needs to use the cdl:getVariable function.

IV.3.c Annotation of Trust, Security, and Contract (TSC) Tasks

As an augmentation of WS-CDL documents a conceptual model for a collaborative business process security concept was introduced in D16, the TrustCoM conceptual models V1, and is further refined in the Appendix. The goal of this concept is to inject security controls where required into the role specific executable public/private business processes. To achieve this, the collaboration definition activities and interactions are annotated with so-called TSC Extension Roles. This concept serves its purpose if, at collaboration definition modelling time, it is at least known, that a TSC control has to be enforced at a specific interaction in collaboration. This is realised by adding an empty TSC Extension Role only containing the header data, the specific role can be deployed at runtime by the BPM service.

IV.4 XACML SICS

XACML is an OASIS standard for access control policies. This document describes how XACML is used in TrustCoM. The aim is to define a common method of applying XACML in order to provide for interoperability and easy to use guidelines which save time and effort for the TrustCoM partners.

[20] WSDL 2.0 Message Exchange Patterns

TrustCoM uses XACML 1.1[21] with the delegation extensions developed at SICS[22].

XACML is based on the concept of attributes. Subjects and resources are defined in terms of their attributes, for instance the role of a user is an attribute of the subject and the name of a service is an attribute of the resource. Policies are written in terms of these attributes and the attributes of the subject and resource that is being accessed are made available to the PDP, which can then calculate whether the access should be permitted or not.

One important part of this profile is to define which attributes are available for policy writers to refer to in their policies. Another part of this profile makes recommendations on the overall structure of policies and how the delegation features fit in the overall picture of TrustCoM.

XACML itself does not define any kind of transport formats. This profile defines how policies are enveloped in a signed transport format for secure distribution.

IV.4.a Attributes

In the TrustCoM PDP there are two sources of attributes. The PEP will add attributes to the request it sends to the PDP. These attributes concern the access entities, that is the subject, resource, action and environment. In addition to this, the PDP will get attributes from the tokens that policies have been signed with. These attributes concern the issuers of policies and are used to verify that the policies have been issued in an authorized manner.

The attributes that the PEP fills in the request can be divided into two categories: application independent attributes and application specific attributes. The application independent attributes are derived from the SOAP header of the service invocation that is under access control and the WS-Trust token from the SOAP message. The application specific attributes may be based on content from the SOAP body.

IV.4.a.1 Attributes based on the SOAP header

The following attributes are derived from the SOAP header.

Description	Address of the invoked service. Value of <wsa:To> element.

[21] XACML 1.1: The full standard is available at http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf. For a brief and easy to understand overview see http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

[22] http://www.sics.se/isl/pbr/xacml/XACML-delegation.doc

XACML request section	Resource
Attribute id	urn:oasis:names:tc:xacml:1.0:resource:resource-id
Value	Content of /soap:Envelope/soap:Header/wsa:To element
Type	http://www.w3.org/2001/XMLSchema#anyURI

Description	Value of <wsa:Action> element
XACML request section	Action
Attribute id	urn:oasis:names:tc:xacml:1.0:action:action-id
Value	content of /soap:Envelope/soap:Header/wsa:Action element
Type	http://www.w3.org/2001/XMLSchema#anyURI

The XML fragments below show how an example SOAP header translates to XACML attributes in the request.

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <soap:Header>
<wsa:Action>http://tempuri.org/RepositoryMngSoap/getContentsByProjectRequest</ws
a:Action>
    <wsa:MessageID>urn:uuid:a1543b3d-451c-458c-b528-8b6e67df00d5</wsa:MessageID>
      <wsa:ReplyTo>
<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</ws
a:Address>
    </wsa:ReplyTo>
  <wsa:To>http://localhost:3998/SP_WS/RepositoryMng.asmx</wsa:To>
  </soap:Header>
  <soap:Body>...</soap:Body>
</soap:Envelope>
```

```
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context">
  <Subject>...</Subject>
  <Resource>
    ...
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:
resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>http://localhost:3998/SP_WS/
RepositoryMng.asmx</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    ...
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>http://tempuri.org/RepositoryMngSoap/
getContentsByProjectRequest</AttributeValue>
    </Attribute>
  </Action>
  </Request>
```

IV.4.a.2 Attributes from WS-Trust tokens

Every claim in the token translates into one attribute of the issuer. Currently we are using mock tokens since the STS from EMIC does not support asymmetric cryptography yet. This specification will be updated once support for real tokens becomes available.

The content of the AttributeId element becomes the attribute id in the XACML attribute and the content of the AttributeValue element becomes the value of the XACML attribute. The two XML fragments below show how a token translates to XACML attributes:

```
<MockToken>
  <Claim>
    <AttributeId>group</AttributeId>
    <AttributeValue>administrator</AttributeValue>
  </Claim>
  <Claim>
    <AttributeId>hair_color</AttributeId>
    <AttributeValue>pink</AttributeValue>
  </Claim>
</MockToken>
```

```
<Attribute
  AttributeId="group"
  DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>administrator</AttributeValue>
</Attribute>
<Attribute
  AttributeId="hair_color"
  DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>pink</AttributeValue>
</Attribute>
```

Currently all attributes derived from WS-Trust tokens are of string type. (This could be changed if there is need for other types of attributes.) TrustCoM does not use the “issuer” attribute of XACML attributes.

It is the responsibility of the PEP to verify that the token is valid and trusted when it comes to the attributes that the PEP includes in the request. When it comes to the attributes of policy issuers, it is the responsibility of the PDP to validate the tokens.

IV.4.a.3 Application specific attributes

Access control and delegation policies, as processed by the PDP, may refer to application specific attributes. For the case where the values of these attributes are to be recovered from the bodies of SOAP messages (corresponding to service invocations and responses), this profile suggests two solutions: (1) the PEP can forward the whole body of the message to the PDP, thus letting the PDP extract attribute values using XPath expressions specified in its policies; or (2) the PEP examines the message body itself, extracts attribute values and places them in the authorization request.

Alternative (1) may result in excessive communication costs, depending on the size of the SOAP message bodies, but has the advantage over (2) that the PEP does not need to be configured with application-specific information. Otherwise, in case (2), the Policy Service, i.e. the service that uploads policies to the PDP, could also be in charge of configuring the PEP with information on how to extract attributes from the message bodies.

IV.4.b Policies

Delegation

When a service is deployed, a root policy must be installed in the PDP which will serve the new service. The root policy should contain a full delegation right for the owner of the service. The access policies will be created by having the service owner issue them. The root policy is not modified during normal operations. If the policies need to be changed, new signed policies may be added or removed. This way daily administration can be decentralized as needed by means of the delegation model. However, the current scenarios do not cover access policy reconfiguration, so this issue has not been explored in detail.

The right to delegate is expressed by means of a condition on delegation chains. A delegation chain is a special attribute in the environment section of the request which specifies whether the request is a request for access to a resource or a request for verifying the authority of a policy issuer, with the contents of the chain detailing the whole chain of issuers leading to the final access permission. By writing conditions on the delegation chain we can differentiate between rights to issue policies (administrative rights) and access rights, and also specify limits on further delegation of administrative rights[23]. For TrustCoM we limit the policies to three kinds: access policies, administrative policies which do not allow further delegation and administrative policies which allow further delegation.

Access policies

An access policy shall contain a condition which constrains the delegation chain to only access requests. The XML fragment below shows what this kind of condition looks like. Notice the empty delegation constraint, which means that the condition will match only requests with an empty delegation chain. Requests with an empty delegation chain are access requests.

```
<Condition FunctionId="urn:FIXME:function:delegationMatch">
```

[23] See http://www.sics.se/isl/pbr/xacml/XACML-delegation.doc.

```
  <AttributeValue
    DataType="urn:FIXME:data-type:delegationConstraint">
  </AttributeValue>
  <Apply
    FunctionId="urn:FIXME:function:delegationChain-one-and-only">
    <EnvironmentAttributeDesignator
      AttributeId="urn:FIXME:environment:delegationChain"
      DataType="urn:FIXME:data-type:delegationChain"/>
  </Apply>
</Condition>
```

Administrative policies without further delegation

An administrative policy without further delegation shall contain a condition on the delegation chain with a single step which specifies the required attributes of the subject of the administrative right.

The XML fragment below shows an example of a condition which allows anyone in the administrator group to issue access policies. This condition does not allow creation of administrative rights.

```
<Condition FunctionId="urn:FIXME:function:delegationMatch">
  <AttributeValue
    DataType="urn:FIXME:data-type:delegationConstraint">
    <ConstraintStep>
      <SubjectMatch
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string"
          >administrator</AttributeValue>
        <SubjectAttributeDesignator
          AttributeId="group"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
    </ConstraintStep>
  </AttributeValue>
  <Apply
    FunctionId="urn:FIXME:function:delegationChain-one-and-only">
    <EnvironmentAttributeDesignator
      AttributeId="urn:FIXME:environment:delegationChain"
      DataType="urn:FIXME:data-type:delegationChain"/>
  </Apply>
</Condition>
```

Administrative policies with further delegation

An administrative policy with further delegation shall contain a condition on the delegation chain where the first constraint step specifies the required attributes of the subject of the administrative right and the second constraint step specifies the required attributes of those that the administrative right may be further delegated. The MaySkipOrRepeat attribute of the second constraint step shall be true, meaning that there is no limit on the number of times the administrative right may be delegated further and also allowing for the creation of an access permission.

The XML fragment below shows an example of a condition which allows anyone in the administrator group to issue either administrative or access policies. The second constraint step in this case is empty, meaning that the administrator may delegate an administrative right to anyone.

```
<Condition FunctionId="urn:FIXME:function:delegationMatch">
  <AttributeValue
    DataType="urn:FIXME:data-type:delegationConstraint">
    <ConstraintStep>
      <SubjectMatch
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string"
          >administrator</AttributeValue>
        <SubjectAttributeDesignator
          AttributeId="group"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
    </ConstraintStep>
    <ConstraintStep MaySkipOrRepeat="true">
    </ConstraintStep>
  </AttributeValue>
  <Apply
    FunctionId="urn:FIXME:function:delegationChain-one-and-only">
    <EnvironmentAttributeDesignator
      AttributeId="urn:FIXME:environment:delegationChain"
      DataType="urn:FIXME:data-type:delegationChain"/>
  </Apply>
</Condition>
```

IV.4.c Transport formats

Policies are to be signed with WS-trust tokens in a way very similar to how WS-Security defines signatures for SOAP messages. Currently the STS does not support asymmetric cryptography, so we cannot yet sign policies for real. For now this section defines an unsigned transport format based on a mock token. The real signature format will be similar to this.

The XML fragments below show examples how an XACML Policy and a PolicySet are signed. For brevity, the actual contents of the policies have been truncated.

```
<AssertWithToken xmlns="http://eu-trustcom.com/PDP/interface">
  <MockToken>
     <Claim>
       <AttributeId>group</AttributeId>
       <AttributeValue>administrator</AttributeValue>
     </Claim>
  </MockToken>
  <Signature/>
  <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy">
    ...
  </Policy>
</AssertWithToken>
```

```
<AssertWithToken xmlns="http://eu-trustcom.com/PDP/interface">
  <MockToken>
```

```
      <Claim>
        <AttributeId>group</AttributeId>
        <AttributeValue>administrator</AttributeValue>
      </Claim>
    </MockToken>
    <Signature/>
    <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy">
      ...
    </Policy>
</AssertWithToken>
```

See the section below on the TrustCoM PDP schema for details of the format.

Obligation to validate issuer

The SICS delegation extensions make use of a special obligation to implement the verification of the right to issue a policy[24]. The policies in the transport format do not contain this obligation to verify the authority of the issuer. Instead the obligation is automatically generated by the PDP based on the token when the transport format is decoded by the PDP. The reason for this is that the obligation is what triggers the policy validation process inside the XACML engine and we cannot trust the issuer to supply it himself.

IV.5 ECA-Policies IC

There are **no** specific standards in this scope on which the policy service is based. The closest standard in scope would be DMTF's Policy Core Information Model (PCIM) and TM-forum's NGOSS which do not explicitly address web-services. Numerous discussions have taken place on the difference between ECA-based models and PCIM like 'if-then' models which would not be appropriate to review here in detail[25].

Amongst the WS-* standards the most relevant would be WS-Policy and to a lesser extent WS-SecurityPolicy. The latter defines configuration directives for the security of a web-service and is applicable only in the sense that ECA actions may push WS-SecurityPolicy assertion documents to web-services in order to configure them. The ECA model adopted in this project is itself agnostic to the specification and does not impose constraints upon the parameters of the actions performed. WS-Policy is a relatively simple and still evolving standard. It aims to define a container format for policy assertions and combinations of policy assertions (e.g., policy alternatives) and defines syntactical elements in order to associate WS-Policy documents with specific web-service scopes. Also note that operators for policy combinations are undergoing important revisions. It would be possible to package the ECA-policies used in this project into <wsp:Policy> tags but little would be gained from this additional packaging as the semantics of the policies would still need to interpreted by a policy service which offers this functionality. Indeed, WS-Policy does not restrict which assertions may be specified and it is up to the web-service to be able to interpret them by offering support for the specific derived standard that defines their content such as WS-SecurityPolicy.

[24] See http://www.sics.se/isl/pbr/xacml/XACML-delegation.doc.

[25] For an example of such discussions see Strassner, J., *Policy based Network Management : solutions for the next generation.* Morgan-Kaufmann Publishers 2003. ISBN: 1558608591

Note however that the policy service defined in this framework can be used in order to reconfigure web-services with specific WS-Policy documents if the services present a management interface that allows them to do so.

Glossary all

Term	Definition
Application Service	*A service that may perform a certain, business oriented task according to a pre-defined workflow.*
Application Service Provider	*Enterprises, companies or individuals that provide Application Services and offer them via searchable registries to customers.*
BP Management System	*The unit responsible for the overall business process that is to be realised by the virtual organisation.*
Business Process	*An abstract workflow that describes the action and tasks a unit has to enact.*
Collaboration Agreement	*→ General VO-Agreement*
Collaboration Definition	*A Collaboration Definition (CD) captures the global view of a business collaboration among roles. It entails roles, high-level activities and interactions.*
Collaboration Definition Template	*A Collaboration Definition Template captures the recurring best practices for a specific well known business collaboration in the format of a CD. It is usually stored in a repository.*
Collaborative Business Process	*A Collaborative Business Process (CBP) entails the set of public and private processes derived from or associated to a CD for each specified role in the CD*
Component	*The smallest functional and/or logical unit. Tightly coupled components interact in order to fulfil a specific task form a → subsystem*
Contract	*A form of convention that designates the behaviour the involved parties commit to. This is principally the legal counterpart to → SLAs.*
Dissolution Phase	*During this → VO lifecycle phase the → Virtual Organisation is dissolved again and all partners are released from their respective bindings (→ contracts, → SLAs etc.)*
Enterprise Network Agreement	*A description of the requirements to be fulfilled in order to become member of an Enterprise Network. This is the basis for the → General VO Agreement*
Evolution Phase	*Sometimes distinguished from the → Operation Phase. During this → VO lifecycle phase, changes to the → Virtual Organisation may occur – this covers in particular the addition & exclusion of individual partners.*
General VO-Agreement	*The "high-level" definition of all the parameters and rules that have to be fulfilled by all participants. This may involve → Contract terms*
Identification	*The → VO lifecycle phase during which potential partners to support*

Phase	*the overall business goal are discovered.*
Formation Phase	*In this → VO lifecycle phase partners are invited to the → Virtual Organisation and they are provided with the necessary information to collaborate. During this phase the VO is actually formed.*
Operation Phase	*The → VO lifecycle phase during which the → Business Process(es) are executed to reach the VO's business goal(s)*
Policy	*Rules defining choices in the behaviour of systems. Within the scope of the TrustCoM project several types of policies are considered.* - *SLA Obligation policies which define the obligations of a party in respect to the provision of a QoS to the other party. These policies trigger notifications when the specified QoS has been violated.* - *Access control policies in the form of authorisation and delegation policies which define who can access services and under which constraints.* *Obligation Policies (in the form of Event-condition-action rules) which define how the VO should adapt to failures, changes in requirements, security events, etc.*
(Authorisation) Policy Decision Point	*Decides which messages are permitted or not depending on the current access control policies.*
(Authorisation) Policy Enforcement Point	*It is the point where the incoming message is intercepted, the tokens provided with the message are verified and an access control decision is requested from the PDP.*
Private Process	*A private process is an executable business process enacted by a BP engine, contributing to the VO's business objective by orchestrating services. A private process is confidential to a VO member domain, the process owner, due to optimisation and associated sensitive information.*
Public Process	*A public process captures the externally visible part of exactly one private process. The public process can be seen as the private process interface with the minimal exposure to let the private process collaborate in a CBP.*
Repository/ Registry	*A database that stores information about (publicly available) services, like e.g. their WSDL, → SLA templates etc.*
Security Token	*Contains authentication relevant information, may also contain access-rights and related data.*
SLA	***S**ervice **L**evel **A**greement: an electronic form of → contract, that is only of limited legal impact. It describes the quality of service that has to be maintained.*
SLA Management System	*Responsible for managing → SLAs – this includes negotiation of the parameters, monitoring & evaluating service performance and*

	enforcing the obligations.
SLA Template	*A document that contains the parameters that can principally be fulfilled by the service that provides the template.*
Subsystem	*A subsystem represents a logical and functional unit of → components that interact in order to fulfil the subsystem's task.*
Supporting Service	*Services that are in themselves not part of the virtual organisation, but that are used by the latter to fulfil certain purposes. Supporting services are mainly → Repositories and Registries so far.*
Trust	*In the sense used here mostly related to "trustworthiness": the expectations put in a service to behave in a particular way. This reflects first of all an evaluation of past performance.*
Trusted Third Party	*Services that participate in a virtual organisation, yet do not directly contribute to the realisation of the overall → Business Process (as opposed to an → Application Service)*
TSC Extension Role	*The TSC Extension Role is part of the BP TSC concept and it configures a TSC task, to perform TSC control functions based on defined subsystem EPRs and parameters.*
TSC Task	*The TSC task is part of the BP TSC concept which provides process control based on the TSC subsystems during process instance runtime.*
Virtual Organisation	*A set of business entities that work together (by message exchange etc.) to reach a common goal – generally represented by an overall → Business Process.*
VO Lifecycle	*The → Virtual Organisation traverses 5 main phases that logically distinguish the actions to be performed. These phases are: → Identification, → Formation, → Operation & → Evolution (sometimes regarded as one phase) and → Dissolution.*
VO Manager	*The central management instance that acts on behalf of the VO-customer. This entity is responsible for "guiding" the VO lifecycle and performing membership-related management tasks.*

Key to diagrams

The following UML diagram types may be used in this document:

- o Dynamic modelling by **Activity diagrams** with swimlanes for different sub-systems, or components of subsystems.
- o **Component** or **Composite Structure diagrams** to represent the structure of components that form a subsystem (respectively subsystems that form the TrustCoM system in the overview case) and their dependencies to each other.

When diagrams are used they should use symbols and notations defined in the standard Rational Unified Process (RUP); the elements for these diagrams are summarised below.

Summary of Activity Diagram Elements

	Diagram Element	**Symbol**	**Represents**
Nodes	Action state	Horizontal capsule	A process
	Decision	Diamond	Next step may be only one of several sucessors
	Swim lane	Parallel vertical lines	A group of related processes
	Synchronization point	Thick bar	All predecessors must terminate before the successor can start
	Object	Object box	An object, component, or subsystem
	Signal receiver	Notched rectangle	The successor cannot be started until the signal is received
	Signal sender	Pointy rectangle	A signal is sent before the successor start
	Initial action state	Filled circle	Predecessor to the first action state
	Final action state	Bull's eye	Final action state
Edges	Control flow	Solid arrow	Pre- and post-decessor relationship
	Message flow	Dashed arrow	Message sent to/from an object
	Signal flow	Dashed arrow	A pair of sender/receiver nodes

Summary of Component Diagram Elements

	Diagram Element	**Symbol**	**Represents**

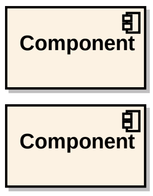
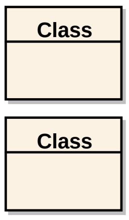
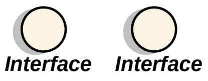
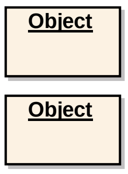
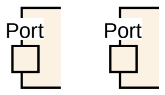
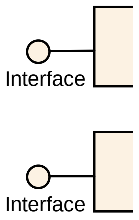
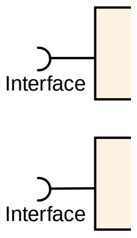
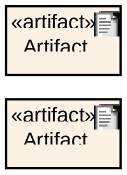
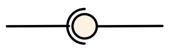
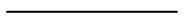
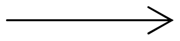
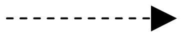
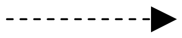
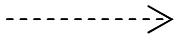
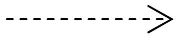
	Diagram Element	Symbol	Represents
Nodes	Component		A modular part of a system, whose behaviour is defined by its interfaces
	Class		Representation of object(s), that reflects their structure and behaviour within the system.
	Interface		A specification of behaviour supported.
	Object		Particular instance of a class.
	Port		A distinct interaction point
	Provided Interface		Interface provided by the component
	Required Interface		Interface required by the component
	Artifact		Physical piece of information used or produced by a system

	Diagram Element	Symbol	Represents
Edges	Assembly		Connection between a provided and a required interface
	Associate		Denotes relationship between two elements.
	Delegate		
	Realise		
	Generalise		
	Dependency		
	Trace		

Summary of Composite Structure Diagram Elements

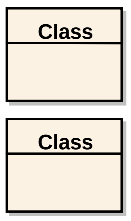
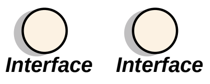
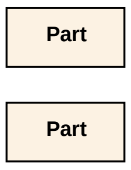
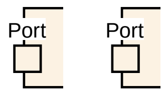
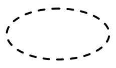
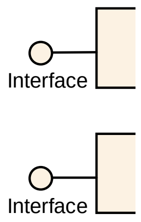
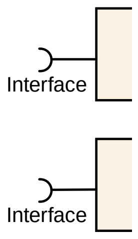
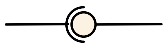
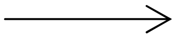
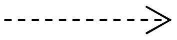
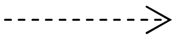
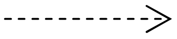
	Diagram Element	Symbol	Represents
Nodes	Class		Representation of object(s), that reflects their structure and behaviour within the system.
	Interface		A specification of behaviour supported.
	Part		Run-time instances of classes or interfaces.
	Port		A distinct interaction point
	Collaboration		

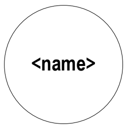
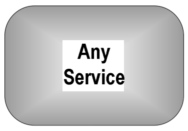
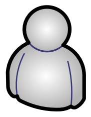
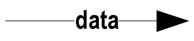
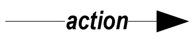
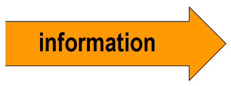
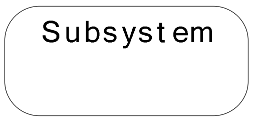
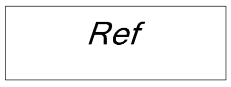
	Diagram Element	Symbol	Represents
	Component provides Interface		Interface provided by the component
	Component uses Interface		Interface required by the component
Edges	Assembly		Connection between a provided and a required interface
	Connector		Communication link.
	Delegate		
	Role Binding		
	Represents		
	Occurence		

This document furthermore makes use of a non-UML based diagram type to depict the so-called “relationship model” introduced with this document. This diagram type reflects the *potential* information transport between components, non-regarding their deployment, respectively actual “usage environment” (see chapter III for a detailed description of the diagram type).

Though such a model could principally be depicted using UML specific representations, we chose the following symbolic representation to avoid confusion:

Summary of Relationship Model Diagram Elements

	Diagram Element	Symbol	Represents

	Diagram Element	Symbol	Represents
Nodes	Component		Components, parts of the TrustCoM framework – generally services and/or libraries.
	Service		Any (web) service that participates in a Virtual Organisation.
	User		Human beings in a VO (customer, service owner / administrator etc.)
Edges	Message		Passing data
	Trigger		Action invocations (triggers)
	Relationship		Data relationship on subsystem level
Other	Boundary		Logical boundary of a subsystem
	References		Reference to other diagrams

References

[1] Preliminary Conceptual Models for the TrustCoM Framework, ID 1.1.2, version 1.0 - http://portal.sema.es/pls/portal30/docs/FOLDER/TRUSTCOM_AREA/WORKFOLDERS/AL1FRAMEWORKDEFINITION/ACTIVITY11CONCEPTUALMODELS/WP1111TRUSTCONTRACTMANAGEMENTMODELS/ID112/ID.1.1.2_V1.0.DOC

[2] TrustCoM Reference Architecture, D09, ID 1.2.4, version 1.0

[3] Baseline Prototype Infrastructure for the CE Scenario, D10, version 1.0

[4] The TrustCoM Concepetual Models, D16, version 1.0

[5] VO Trust, Security & Contract Management Framework, D18, version 1.0

[6] An Intermediate Assessment of the TrustCoM Framework using the CE Scenario, D20

[7] An Intermediate Assessment of the TrustCoM Framework using the AS Scenario, D21

[8] W3C. Web Services Choreography Description Language, 2005. W3C Latest Working Draft from October 8th, 2005, work in progress

[9] Marlon Dumas and Arthur H. M. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In UML '01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, pages 76–90, London, UK, 2001. Springer-Verlag.

[10] Alistair Barros, Marlon Dumas, and Phillipa Oaks. A Critical Overview of theWeb Services Choreography Description Languages (WS-CDL). BPTrends Newsletter, Vol. 3, March 2005

[11] Alistair Barros, Marlon Dumas, and Arthur H.M. ter Hofstede. Service Interaction Patterns: Towards a Reference Framework for Service-Based Business Process Interconnection. http://sky.fit.qut.edu.au/ dumas/ServiceInteractionPatterns.pdf, April 2005.

[12] Roberto Gorrieri, Claudio Guidi, and Roberto Lucchi. Reasoning about interaction patterns in Choreography. In Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM '05), 2005.

[13] WS-Trust specification from February 2005 (http://msdn.microsoft.com/ws/2005/02/ws-trust/).

[14] SAML 1.1 Assertion specification (http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf).

[15] Web Services Security SAML Token Profile 1.1 (http://www.oasis-open.org/committees/download.php/15256/Web%20Services%20Security%20SAML%20Token%20Profile-11.pdf).

[16] W. Saabel, T.M. Verduijn, L. Hagdorn and K. Kumar. A Model for Virtual Organisation: A Structure and Process Perspective. Electronic Journal of Organizational Virtualness, Vol. 4, 2002