Telecommunications
Information
Networking
Architecture
Consortium

TINA-C Deliverable

Issue Status: Publicly Released

Service Architecture

Version: 5.0

Date of Issue: 16 june 1997

Table 1:

Page	Company	Company Contact (Address, Telephone, Fax)

Telecommunications
Information
Networking
Architecture
Consortium

TINA-C Baseline

Issue Status: Final

Service Architecture

Version: 5.0

Date of Issue: 16 June, 1997

Abstract: This document defines the TINA Service Architecture. The architecture consists of a set of concepts, principles, rules and guidelines for constructing, deploying, operating and withdrawing TINA services. It also describes the environment in which such services operate. The service architecture identifies components to build services, describe the way they are combined, and the way they interact.
One of the main separations described is the separation of access and usage, and within usage the separation of service session and communication session.
The access and service sessions are defined to work across multiple domains, thus allowing retailers and third party service providers to offer compound services in a business environment.

Main Authors:	**C. Abarca, P. Farley, J. Forslöw, J. C. García, T. Hamada, P. F. Hansen, S. Hogg, H. Kamata, L. Kristiansen, C. A. Licciardi, H.Mulder, E. Utsunomiya, M. Yates**
Editor:	**Lill Kristiansen**
Stream:	**Service Stream**
Workplan Task:	**Service Architecture**
File Location:	**/u/tinac/97/services/docs/sa/sa5.0/final/ also available through http://www.tinac.com**

Executive summary

The document is of interest to anyone wanting to perform the business role of retailer or third party service providers, as well as to service developers and designers of service creation environments.

This document defines the TINA Service Architecture. The architecture consists of a set of concepts, principles, rules and guidelines for constructing, deploying, operating and withdrawing TINA services. It also describes the environment in which such services operate. The service architecture identifies components to build services, describe the way they are combined, and the way they interact.

The TINA service architecture separates access session from service session. The service session may again use communication sessions. This separation of services and communication allow easy introduction of new services independent of the underlying network. It also supports personal mobility and session mobility.

The access and service sessions are defined to work across multiple domains thus allowing retailers and third party service providers to offer compound services in a business environment. Both a user/ provider paradigm, and a symmetric peer paradigm are described.

The composition framework allows new services to be composed of (simpler) other services. These other services may possibly be offered by another (third party) service provider. Service providers may also federate to provide services in a multidomain environment.

TINA is object orientated. This allows extensive reuse of concepts and definitions. Service components and interfaces are defined so that they are reusable, and extensible, to create new services from predefined components by inheritance.

TINA also bases its work on Open Distributed Processing (ODP), including separate computational and informational viewpoints. TINA assumes services execute in a Distributed Processing Environment (DPE).

The document is part of the TINA baseline series, which comprises the following documents:

- Network Resource Architecture,
- DPE Architecture,
- Service Architecture,
- TINA Computational Modeling,
- TINA Information Modeling,
- Network Component Specification,
- Service Component Specification,
- Documents describing each of the interdomain reference points.

Table of Content

1 Introduction

1.1 Purpose

The TINA service architecture consists of a set of concepts, principles, rules and guidelines for constructing, deploying, and operating TINA services. The behavior of the elements in a TINA environment is modelled by components[1] that operate in a Distributed Processing Environment (DPE) and interact via interfaces, as explained in [5]. The TINA service architecture identifies components to build services, describes the way they are combined and how they should interact. The architecture also examines what components are needed in a support environment to instantiate, manage, and use services.

1.2 Audience

This document should be read by designers and developers of services, as well as providers and operators of services. It is also interesting for designers of service creation environments. Organizations that wish to build and/or offer reusable service components should also read this document.

1.3 Document History

The TINA service architecture appeared in the following snapshots:

- Definition of Service Architecture v1.0 (December 1993) [9];
- Service Architecture v2.0 (March 1995) [10];
- Service Architecture Delta 1995 (April 1996) [11];
- Service Architecture v4.0 (October 1996) [7];
- Service Architecture v4.1 (January 1997) [12].

[12] has status as a technical report, while [7] is still the valid baseline. Some new material was included in the annex part of [12], and hence is also included in this document.

The main changes with respect to [7] are:

- Fine details have been moved to the service components specifications (SCS)[14], leaving the architecture document to supply principles, define terminology and explain high level concepts and achitectural context. The reason for moving details is that SCS may be maintained after 1997. Details moved are, e.g., material like the details of the description of the dynamic behavior. Also, in the information modeling chapter some details have been moved.
- A restructuring of the chapters has been done, e.g., introducing a new chapter explaining the overall principles and terminology.
- Service composition and federation have been integrated in the main body as a new chapter. Composition and federation called for some changes (mainly generalizations) in other parts of the main body as well, as indicated in the open issues in the previous baseline [7].
  - The separation of access and usage has been clearer and stronger;

1. Component, computational object and computational object group:
   For the fine distinctions between them see details in Section 7.1.

  - The service session graph has been extended to handle composition and federation.
- Service management has been included in the main body.
- Material stabilized during the last 6 months' work on the Ret reference point[2] have been included, such as feature sets like e.g., multi-party, stream binding etc. Input from the Vital auxiliary project has been of great value here [30].
- Due to the movement of some material from the annex to the main body, the annex was slightly restructured. Annex 2 (Service Example) which was quite detailed in [7], has been removed and will be placed in a future document [15] entitled 'Developer's Guide to TINA'.

1.4 How to Read this Document

Due to the wide scope of the service architecture, the content of this document is heterogeneous:

The document is heterogenous in *scope* (e.g., from subscription to runtime issues, and from access to service usage) and in *description techniques* (e.g., from principles and supporting concepts, information models etc. to description of service components interacting via interfaces in computational models).

Depending on the reader's background, some readers might find the document easier to read if s/he follows the following advice:

- The beginning of Section 7.2 provides an overview of the service architecture components from the computational viewpoint, which some readers might want to read at an earlier stage (e.g., before Section 4).
- The document contains some cross-references to later parts of the document (e.g., there are cross-references in sections 4 and 6 to section 7). It might be useful to follow these cross-references, and then return to the previous sections later.
- Section 7.4 gives examples in the computational view, which will ease the understanding of the previous sections. The annex provides additional information, including further examples in the computational view. The annex also offers one example of the evolution of the session graph defined in section 6.4, along the steps of a computational example.

Different sections rely (to different extents) on the documents listed in Section 1.5. See 1.5 for further details.

The material in this document also offers another degree of heterogeneity: It ranges from core TINA principles and concepts and mature examples, to suggestions, additional information, and less mature examples which have been included to facilitate readability etc.

In order to master this complexity and to separate different content types clearly, the document is divided in two parts:

- Definition of Service Architecture, (the main body of the document);
- Annex.

Note that the service architecture is one document, and is composed of the two parts above. Throughout the service architecture document, the term "this document" refers to the service architecture document as a whole, while the term "document part" refers to either the main body, or annex separately.

2. The Ret-RP document is not yet (May '97) finalized; a draft can be found in [13].

Chapters in the main body are numbered with plain numbers (1, 2...); chapters in the annex are numbered adding the prefix “A-” (A-1.2.3,...).

The two parts of the document are presented below.

1.4.1 Definition of Service Architecture (the Main Body)

The main body of the document defines the framework we call the TINA service architecture: this means the core concepts to use, the main principles to apply and the rules to follow. This part also presents descriptive material (introductions, mature examples...) that is necessary for clarity and readability.

This part of the document defines a business model (corresponding to the ODP enterprise viewpoint), an information model (corresponding to the ODP information viewpoint), and a model for service components (corresponding to the ODP computational viewpoint). In addition some general architectual principles are described, e.g., the separation of access and usage[3] and principles for service composition and federation.

The ODP viewpoints, described in [40], provide a framework to identify different models; however, the TINA service architecture does not claim compliance with ODP standards.

This part of the document is the reference for compliance to the TINA service architecture framework. In addition TINA will provide more detailed documents for definition of compliance to each of the reference points described in Section 2 [4].

Description of Chapter Content

Chapter 1 (this chapter) provides a general introduction to the whole document, including its relationship with other parts of the TINA architecture.

Chapter 2 presents the business model for the service layer. This is the model of the service architecture from the perspective of business relationships, of interactions between business roles and stakeholders, and of the separation of business administrative domains.

Chapter 3 presents a key concept for the service architecture: the concept of session. It describes the access, service usage and communication separation. These session concepts are presented first informally, then using the semi-formal approach of OMT diagrams [51], which rely on the concept of information modeling.

Chapter 4 explains how the architecture supports composition and federation. It scopes what the architecture provides and states basic principles that form the basis of service level composition and federation. It then discusses a number of generic paradigms which the architecture supports.

Chapter 5 explains the management part of the TINA service architecture. The objectives are to introduce the principles, requirements, and supporting concepts that are the basis of service management in TINA.

Chapter 6 gives an overview of the information models of the service architecture. This includes information models of service, sessions, service session graph etc. using OMT diagram. This corresponds to the high level model of the service architecture in the ODP information viewpoint.

3. This separation and the further separation of usage into service session and communication session, imply that TINA has adopted what ITU calls ‘separation of call control and connection control’. (See NRA [8] page 27.)

4. Currently (May ‘97) Ret-RP, ConS-RP and TCon-RP are being worked on, with Ret-RP as the one of main relevance to the service architecture (see [13]). Work on the reference points RtR-RP and 3Pty-RP will start later this year.

Chapter 7 gives an overview of the components of the service architecture. It also contains some high level examples of the dynamic behavior.This corresponds to a high level model of the service architecture in the ODP computational viewpoint.

Chapter 8 explains how the service architecture can be used to support personal and session mobility. It gives a definition of both types of mobility, together with the requirements they impose on the service architecture and explains the requirements fulfillments.

Chapter 9 identifies a reduced set of important issues that are not solved by the current version of the service architecture, and therefore require further work. Each issue is briefly described in terms of a problem statement.

Chapter 10 lists the bibliographical references. The Definition of Service Architecture part of the document is not based on TINA-C documents other than valid baseline documents, up-to-date at the time of writing, or mature drafts thereof. However, other TINA-C documents (reports or engineering notes) are referenced in the annex in some cases[5].

Chapter 11 lists and expands all acronyms used in the document.

1.4.2 Annex

The content of this part of the document is heterogeneous. More precisely, the annex contains:

- Concepts, principles, rules and guidelines that are not a core part of the architecture. This material constitutes a refinement of the architecture with respect to what is defined in the main body. It is intended as a set of suggestions to designers to build a service architecture implementation;
- Intermediate results on particular topics that are not yet mature enough to be included in the core part of the architecture. These results identify directions for future work, both inside and outside of the consortium, and therefore need to appear in this document. Future extensions to the service architecture are likely to rely on these results (i.e., some material will probably move from the annex to the main body at a later stage).

1.5 Relationship to Other TINA-C Documents

1.5.1 TINA-C baseline documents

The TINA service architecture document relies on the following TINA-C baseline documents:

- **Computational Modeling Concepts (CMC)** [5]: this document defines the modeling concepts and conceptual tools for computational modeling, (i.e. modeling in the ODP computational viewpoint) in TINA. All parts of the service architecture document that define or describe computational models rely on this document. Since the computational viewpoint is the most important in the service architecture, and service components are defined in the computational viewpoint, the understanding and knowledge of the computational modeling concepts is essential to the understanding of the service architecture document.

5. In addition, the main body contains some references to the annex. However, these are references to examples made to facilitate the readability and do not imply that the main body is based on anything but TINA baseline documents.

- **TINA Reference Points** [3]: this document provides a general framework for the TINA reference points, but also the TINA business model; it defines important modeling concepts that are related to the ODP enterprise viewpoint. Familiarity with the document [3], though not absolutely necessary, is of great help in understanding this document. Section 2 relies heavily on the content of the TINA reference points document.
- **Information Modeling Concepts** [4]: this document defines the modeling concepts and conceptual tools for information modeling, (i.e. modeling in the ODP information viewpoint) in TINA. All parts of the service architecture document that define or describe information models rely on this document. A certain acquaintance with the scope of information modelling and with OMT notation is necessary to understand these parts of this document. This concerns chapter 6 in particular, and chapter 3 and 4 to a certain extent.
- **Network Resource Architecture (NRA)**[8]: this document describes (among other things) the relationships between the service architecture and the network resource architecture. It contains useful information on how informational entities in the service architecture map onto informational entities in the NRA, e.g., how stream bindings map onto terminal and network flow connections. The communication session is described in detail in NRA. The document also illustrates the dependencies between the computational entities, e.g., between SSM,CSM, etc. Parts of chapters 6 and 7 refer to this document.

The following TINA-C baseline documents offer helpful support:

- **TINA Glossary of Terms** [1]: this document lists definitions for all TINA terms.
- **Requirements upon TINA-C Architecture** [2]: this document describes the requirements on the TINA architecture as a whole, from which the requirements and objectives of the service architecture, defined in Section 1.7, have been derived.

1.5.2 Future baseline documents planned for 1997

This document will be accompanied by the following baseline documents, planned for 1997, that are based on it:

- **Service Components Specifications (SCS)** [14]: this document provides the more detailed specifications[6] of the service architecture. It will contain detailed information models using the quasi-GDMO and GRM language (as defined in [4]), and detailed specification of the service components using the TINA ODL language (as defined in [5] and [6]).
- **Developer's Guide to TINA** [15]: this document presents a number of usage examples[7]; it is intended to help understand the TINA architecture.

1.6 Basic Definitions

This section defines the meaning of the term "Service Architecture" by defining the term "Architecture" and the term "Service". A service architecture is an architecture for services.

1.6.1 Definition of Architecture

Architecture is the science of designing and constructing systems[8], and consists of **a set of concepts, principles, rules and guidelines to be applied during design and construction**. When the architecture of a system is examined, a style of design and construction can be observed. This style

6. In fact some details previously found in the service architecture have been moved to this document, as explained on page 9.
7. An updated version of the example in Annex 2 in [7] will be part of this document.
8. The term "system" is used here in it widest sense to mean any set of interacting components.

will depict how the concepts, principles, rules and guidelines relate to one another. The relationship depends on the function of the system, the imagination of the designer, and the principles that should be observed. The TINA service architecture (by analogy) defines the concepts and basic principles necessary in constructing TINA services[9]. Also, for certain types of services, additional principles may be defined that are more problem domain specific. Note that to check for conformance requires checking for adherence to the concepts and principles.

1.6.2 Definition of Service

The term “service” is commonly used in multiple senses. A common-sense definition is that a service is a set of goods or valuable functions offered by a service provider to a customer. In the telecommunication industry, a service can be seen as a packaged set of capabilities that is perceived by a human user when interacting with a telecommunications network or a service provider and for which separate billing can be arranged. In [50], a telecommunication service is defined from an ODP enterprise viewpoint as “... a meaningful set of capabilities provided by an existing or intended network to all who utilize it, like customers, end users, network providers and service providers. Each one sees a different perspective of the service.”

This definition is adopted in this document to define services in TINA, with the necessary refinements to align it to TINA terminology:

A **TINA service** is a meaningful set of capabilities provided by an existing or intended system to all business roles that utilize it; each business role sees a different perspective of the service. (See [3] for a definition of business roles.)

Thus, a “service” is viewed as a unit of usage or description by a user or provider; it is manageable and provides, e.g., accounting and performance information as a unit.

TINA services include at least the following types of services:

- **Telecommunication services**
  Services based on the transport of bits of information between terminals attached to a telecommunication network.The telecommunication service is responsible for the establishment of connections and processing of information related to the connections. A telecommunication network is transparent to the information that is carried between network endpoints;
- **Management services**
  Services responsible for the management of TINA resources. It includes fault, configuration, accounting, performance and security functionalities, as well as service lifecycle management, service instance management and user life cycle management;
- **Information services**
  Services that are able to handle information resources such as movie, sound or document. It includes the storage (information content) and the visualization (the application that is able to interpret the resource). The services needed between storage and visualization such as billing, caching and all such services specific to the information resource are also considered as information services.

9. A TINA service exhibits certain properties such as openness, interoperability, etc. While these properties can be achieved using other techniques, the use of the TINA service architecture is expected to guarantee them. Note also that the TINA service architecture does not aim at designing and constructing TINA services, but provides a set of necessary concepts, principles, rules and guidelines for designers and constructors to use.

The definition of TINA services given above is broad; however, the service architecture addresses only services that are within the scope shown in Figure 2-1 (i.e., connectivity ‘services’ are outside the scope of this document, as they are handled by NRA [8]).

1.7 Requirements and Objectives of the Service Architecture

This section defines the requirements and objectives of the TINA service architecture. The requirements are derived from the requirements for the TINA architecture described in [2]. An assessment of the fulfillment of such requirements and objectives is also presented.

1.7.1 Requirements

1. **Support of a wide range of services**. The TINA service architecture has to support telecommunication, management and information services and should be open to allow the introduction of new classes of services. The service architecture addresses the evolution of services, and should be able to support new requirements and business needs.
2. **Rapid service development and provisioning**. The TINA service architecture must support the rapid development and deployment of services in order to respond promptly to market needs, and at the same time, to reduce development costs. Accordingly, a common approach for the design and management of all kinds of services is required in order to maximize the reuse of service components.
3. **Tailored services**. TINA services must be readily customizable in order to satisfy specific requirements of a variety of customers (ranging from large companies to residential users). Service subscribers and end-users should be offered some direct control in managing their services.
4. **Independent evolution of services and network infrastructure**. Services should be defined independently from a specific network technology. Conversely, the exploitation of new technology should be made easier by the flexibility of the service architecture.
5. **Support for a multi-player (or open) environment**. The service architecture should fit in a multi-supplier/provider/operator environment. The coexistence of a number of stakeholders, performing various business roles, must be supported. In addition, the architecture must define a flexible framework with respect to changes imposed by regulatory bodies. Accordingly, the TINA service architecture must define an open environment which enables the introduction and modification of services, the introduction and modification of software and hardware components from different vendors and organizations, and the interoperability among such services and components.
6. **Service manageability**. The TINA service architecture must enable the management of services and the service infrastructure, and must facilitate the integration of control and management aspects of services. It fosters the definition of a common software infrastructure and related models that support service control and management applications in a similar fashion.
7. **Universal service access**. End-users must be able to access services independently from the physical location and the types of terminals being used. In addition, services must be accessible and usable in a standard and uniform way with respect to the user practice.
8. **Integration of non-TINA systems and services.** The architecture should allow for inter-working with existing systems and services, e.g., with Intelligent Network (IN), Telecommunications Management Network (TMN) or World-Wide Web (WWW) based services.

1.7.2 Objectives

The above requirements determine the objectives that the TINA service architecture should accomplish.

1. Definition of a **set of reusable and interoperable service components** to be composed in service definition and construction. They guarantee low time-to-market for services and interoperability of service software.
2. Definition of a **framework for the TINA reference points** related to services. This ensures interoperability between multi-vendor products, as well as among several stakeholders.
3. Definition of principles and mechanisms that allow systems and services to work together in a seamless fashion (**portability of services across domains**), even though these systems and services may be in different administrative domains (**federation among business roles**).
4. Definition of principles and mechanisms that give more effective support in the network to multimedia communications considering **multimedia and multiparty aspects of services.**
5. Definition of **a granular and flexible session model** that can evolve to serve the needs over time of different customer bases. This aspect of a telecommunication architecture may be important in:
   - allowing a large degree of customization of services,
   - enabling quick response of service providers (i.e. retailers and 3rd party service providers) to new customer needs or to new advances in customer equipment[10].
6. Definition of principles and mechanisms in order to guarantee the **smooth service extension.**
7. Definition of **customizable interfaces** and support to different levels of customization, which can be classified into several categories:
   - customization of pre-choices/pre-conditions on access to other stakeholders,
   - customization of the usage of services,
   - customization of configuration of user-system related resources.
8. Definition of partition and layering principles leading to the **separation of services from the network and computing infrastructure** and related resources.
9. Definition of **interfaces providing an abstract view of the network and computing infrastructure** that enable service applications to make use of network and computing resources transparently.
10. Definition of principles and mechanisms that allow **third-party development of services and applications.**
11. Definition of **user/provider/peer paradigms** determining principles and mechanisms for supporting different stakeholders assuming their business roles, i.e. consumer, retailer, broker, third-party service provider, content provider, connectivity provider.
12. Definition of **service management functions**:
    - Identification of service components that are to be managed.
    - Determination of aspects of the components that are to be managed, i.e. functional areas, life-cycle, etc.

10. New advances may also include the network, and hence the connectivity provider (and TCon reference point), but this is outside the scope of the service architecture.

- Definition of mapping from the identified management functionality to a set of management interfaces that all service software should provide for its management.

13. Definition of mechanisms for **customer access to management services.**
14. Definition of mechanisms for **service composition**, both statically (i.e. during design and construction) and dynamically (i.e. during the service utilization).
15. Definition of principles and mechanisms for **global mobility**, such as support of personal and service session mobility. Personal mobility means the ability for a person to access and use services ubiquitously, i.e. independently of both physical location and specific equipment. Service session mobility means the ability to suspend the use of a service and resume it from a different terminal equipment.[11]
16. Definition of principles and mechanisms enabling the **ubiquitous information access**, i.e. access to information independently of the location of both the information and the accessing party.
17. Definition of principles and mechanisms to support **service availability, security, reliability and performance.**

1.7.3 Requirements Fulfillment

In the following, the main requirements of Section 1.7.1 are mapped onto the objectives in Section 1.7.2 and to mechanism and components presented in the other sections of this document.

Table 1-1. Table illustrating how objectives refine the overall requirements

Requirement	Refined by objectives
1 support of a wide range of services	4, 5, 14[a]
2 rapid service development and provisioning.	1,8,14[b],6
3 tailored services	1, 6, 7, 13
4 independent evolution of services and network infrastructure	8, 9
5 support for a multi-player (or open) environment	2, 3, 10, 11, 13,14
6 service manageability	12, 13, 17
7 universal service access	15, 16, 17 [c]
8 integration of non-TINA systems and services	Not addressed by this document[d].

a. Considering that service composition is one way of creating new (advanced) services from simpler ones.

b. Considering that service composition is one rapid way of creating (or packing) services (as service composition is also applicable within one retailer domain).

c. And supported also by the DPE.

d. Solution can be found in the work of the auxiliary project EURESCOM P508 (see [26] and [27]).

11. Terminal mobility, i.e. the ability to access and use services independently of the location of the terminal equipment, and service mobility, i.e. the ability to access and use services independently of the location of the physical resources used to provide them, are also comprised in the global mobility concept, but are outside the scope of the service architecture. These aspects will be covered in future versions of the network resource architecture and the DPE architecture.

Here are some further details on how and where in this document the objectives are fulfilled:

- Objective 1 **set of reusable and interoperable service components.** This is supported by the object oriented methodology underlying the computational view. See Section 7. See also objective 14.
- Objective 2 **framework for the TINA reference points**.
See Section 2 and the reference point document [3].
- Objective 3 **federation among business roles**. See Section 2 and Section 4.
- Objective 4 **multimedia and multiparty aspects of services**.
See e.g., Section 3.5.4, Section 6.2, Section 6.3 and Section 7.2.6.2.
- Objective 5 **granular and flexible session model.**
See Section 3.5.4, Section 6.2, Section 6.3 and Section 7.2.6.2
- Objective 6 **smooth service extension.** This is supported by the object oriented methodology underlying the informational and computational modeling. See Section 6 and Section 7.
- Objective 7 **customizable interfaces**. See the definition of User Profile (UPrf) in Section 6.2.2.3, Domain Usage Service Session[12] (D_USS) in Section 6.3.2.3, and the computational objects supporting these concepts as described in Table 7-3 and Table 7-4.
- Objective 8 **separation of services from the network and computing infrastructure** See Section 3, and in addition, the separation of the TINA architecture into TINA DPE architecture, TINA network resource architecture (NRA) and TINA service architecture.
- Objective 9 **interfaces providing an abstract view of the network and computing infrastructure** See Section 2 (ConS interdomain reference point), and the way TINA handles streams, as described in Section 6.4.4 and Section 7.2.6 and Section 7.2.7, and as illustrated in Figure 7-2. See also Section 3.3.
- Objective 10 **third-party development of services and applications.** See Section 2 (3Pty and ConS interdomain reference points), and in addition, the reference point document [3], which also mentions *intra*domain reference points.
- Objective 11 **user/provider/peer paradigm**.
See Section 3, Section 6.2, Section 6.3 and e.g., Section 7.2.
- Objective 12 **service management functions**. See Section 5.
- Objective 13 **customer access to management services.** See Section 3 and Section 5.
- Objective 14 **service composition**.
See mainly Section 4 but also Section 6 and Section 7.
- Objective 15 **global mobility**. A separate chapter (Section 8) is devoted to this, going into more detail on how specific requirements for personal and session mobility is supported by TINA service architecture.
- Objective 16 **ubiquitous information access**.
See also objective 15. Supported by the DPE (location transparency). A naming framework which separates names from addresses will also support this[13].

12. Including the subclass UD_USS supported computationally by the USM.
13. See Section 9 Issues Requiring Further Work.

- Objective 17 **service availability, security, reliability and performance.**
  These topics are not specifically handled in this document, but hooks are present in the service components to plug in the appropriate security mechanisms (e.g., as specified in OMG/CORBA). The separation of computational and engineering viewpoints, also allows for concepts like replication etc. to enhance performance.

2 Business Model

This chapter describes how the TINA business model and reference points [3] are applied to the TINA service architecture.

Figure 2-1 shows the scope of the service architecture with respect to the initial set of business roles and business relationships as defined in [3].

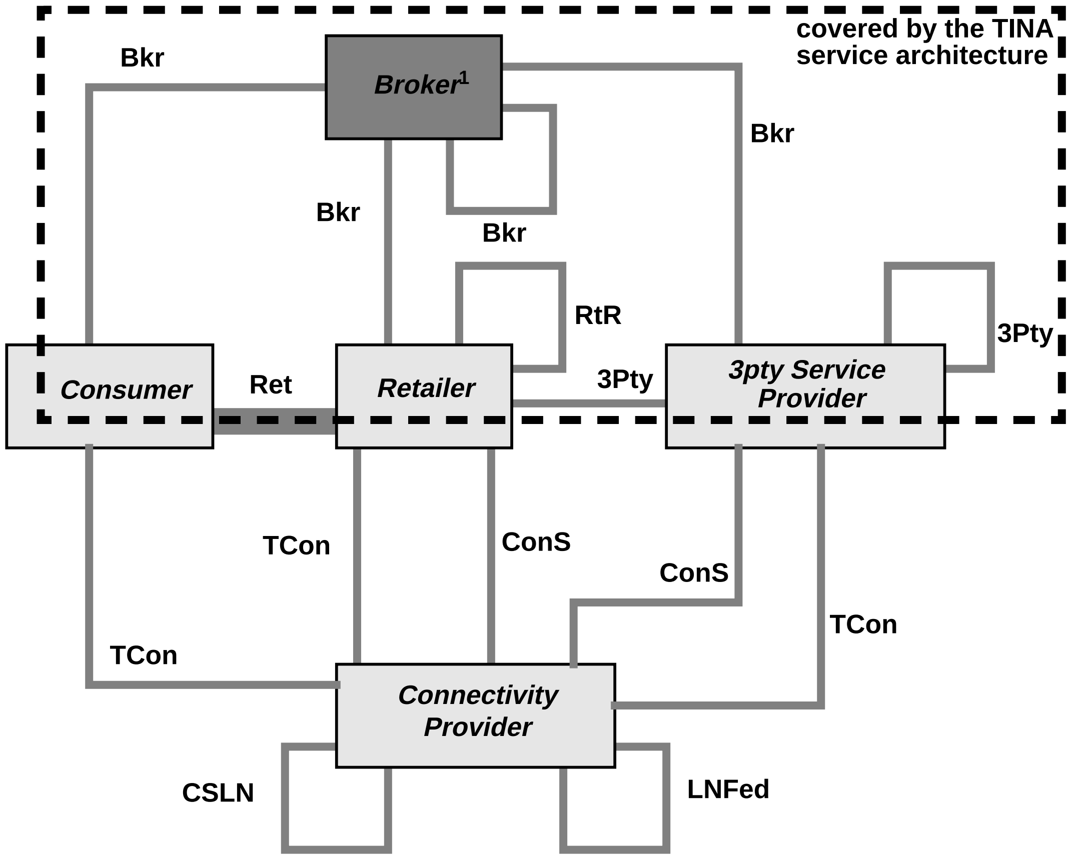


Figure 2-1. Scope of the service architecture with respect to the business model

The **Consumer** business role uses services provided in a TINA system. Consumers use services provided by the stakeholders in the broker and retailer business roles.

The **Broker** business role provides stakeholders with information that enables them to find other stakeholders (business administrative domains) and services in the TINA system.

The **Retailer** business role serves stakeholders in the consumer business role, by providing them with access to services. A retailer may use other providers to support the provision of services to consumers.

The **Third party service provide**r aims to support retailers or other third party providers with services. This can be regarded as “wholesale” of services, as the third party service provider does not have a contractual relationship with stakeholders in the consumer business role.

The connectivity provider manages a network. This network can constitute a transport network to support stream bindings[1] in TINA (to support user connections) or can constitute (a part of) the kernel transport network, to support computational binding in TINA, by supporting the DPE node interconnections.

The business roles indicated in Figure 2-1 are further detailed in the session model given in Figure 3-8 and realized via the computational model described in Section 7. This model provides a high level overview of the computational components present in the business roles (Section 7.2) and the descriptions of the interfaces provided between these components. The information held within the business roles and exchanged between them is described in Section 6.

The consumer, retailer, and third party service provider business roles are mostly within the scope of the service architecture. The business relationships of these business roles with the connectivity provider (ConS and TCon) are within the scope of the network resource architecture [8].In addition, the interactions related to the communication session in the Ret business relationship are also within the scope of the network resource architecture and are not treated in the service architecture. Note that the broker business role has not been studied in detail and is left out of this version of the service architecture.[2]

To explore the maximum functionality of the reference points in the service architecture, each business role is considered to be performed by a separate business administrative domain. The business relationships will thus be mapped one-to-one to inter-domain reference points.

The mapping is made as follows:

Table 2-1. Mapping of business relationships in the service architecture

business relationship from [3]	reference point name	current specification of the reference point in document:
Broker to Consumer, Retailer, Third Party Service Provider, Broker (Brk)	Brk-RP	<not available yet>
Consumer to Retailer (Ret)	Ret-RP	The Ret Reference Point (draft v0.3)[13]
Third Party Service Provider to Retailer, Third Party Service Provider (3Pty)	3Pty-RP	<not available yet>
Retailer to Retailer (RtR)	RtR-RP	<not available yet>

Each of these reference points is specified in a separate document only specifying that reference point. The service architecture provides the descriptions (through the information and component models) of how these reference points work together to provide TINA services. The service architecture also provides the mechanisms (through the session, management, mobility, federation and composition concepts) used to implement the requirements posed on the reference points by the business relationships.

1. The TINA Network Resource Architecture document includes the communication session that handles stream binding. It is not included in the connectivity provider business role.
2. The broker business role is not addressed in this version of the service architecture. A high level description of the required funtionality of the broker can be found in [3].

All of these reference points can be split into two parts: an access part and a usage part. The access part deals solely with the administrative interactions between the business administrative domains, while a usage part, dealing with the functionality to provide the actual interactions for the service and management for that service. The access part will be, more or less, the same for all types of business administrative domains. The asymetric access is treated only once in the reference point documents (in the Ret-RP specification [13]). The symmetric access will also be treated once, (e.g., in the RtR-RP specifications).

2.1 The Broker Reference Point (Brk-RP)

The broker reference point provides access and management of the information controlled by the broker business role by any other TINA business role. The broker can provide different kinds of information to different business roles for different purposes. For example, the consumer can interact with the broker to get references to available retailers. Likewise, the retailer can interact with the broker to get references to consumers for invitations or to third-party service providers for provisioning. Other brokers can also interact with the broker to complement their own information using the same interactions as all other TINA business roles.

The Brk-RP is not explored yet in the TINA service architecture and its impact and specification remains for further study.

2.2 The Retailer Reference Point (Ret-RP)

The retailer reference point is used between stakeholders in the consumer business role and stakeholders in the retailer business role. It is used to support the consumer’s needs for access to the retailer’s services. The Ret-RP [13] offers functionality for the following high level requirements:

- access part:
  - initiation of dialogue between the business administrative domains,
  - identification of the business administrative domains to each other (N.B., either domain can remain anonymous dependent on the interaction requested),
  - establishment of a secure association between the business administrative domains,
  - set up of the default context for the control and management of usage functionality,
  - discovery of service[3] offerings,
  - initiation of usage between the business administrative domains according to the agreed conditions,
  - control and management of sessions (e.g., stop, suspend, resume, join, notify changes, negotiate transfer of control rights, etc.).
- usage part:
  - control and management of sessions (e.g., announce, stop, suspend, invite, notify changes, negotiate transfer of control rights),
  - control and management of stream flow binding,
  - business administrative domain management (e.g., subscriber management, service management)

3. These services can be primary (e.g. Video on Demand (VoD)), ancillary to the primary (e.g. configuration management for VoD) or administrative (e.g., subscriber management for VoD). See Section 3.3.3 for definitions.

The following principles described in the service architecture are used when implementing the Ret-RP reference point requirements:

- Session concept (Section 3.5) and session graph (Section 6.4), providing the definition of the session model and the information structure shared between the parties involved in the service.
- Personal (Section 8.2) and session mobility (Section 8.3), provides the description of how to transfer and manage personal environments between user access points inside a session.
- Management (Section 5), providing the mechanisms to manage both administrative information (e.g. subscribers) and FCAPS (e.g., fault management for a service).

2.3 The Third Party Service Provider reference point (3Pty-RP)

The third party service provider reference point allows a retailer business role to interact with a third-party service provider business role to provide a range of third party services to its consumers, without actually possessing the services. It also allows for interactions between two third party service providers.

The 3Pty-RP fulfills the following high level requirements, in addition to those posed for the Ret-RP:

- control and management of service content,
- control and management of services in wholesale[4],
- management of service offerings in wholesale in the retailer domain or other third party service provider domain (e.g. version control).

The following additional principles described in the service architecture provide meaning for the 3Pty-RP reference point specification.

- Composition (Section 4.2.3), provides the description of how to combine services from other retailers or third party service providers into services offered on the Ret-RP.

2.4 The Retailer to Retailer reference point (RtR-RP)

The retailer to retailer reference point allows retailer business roles to interact to provide services to each other's consumers, essentially allowing the consumers of one retailer to be involved in service sessions with consumers of the other retailer.

The RtR-RP will re-use the functionality from the 3Pty-RP and the Ret-RP, and in addition, focus on service federation.

4. Ret provides the operations for retail provision and management of services (differences being e.g. that wholesale services are provided with all options open, whereas retail services may already be tailored to a consumers needs).

3 Architecture Overview

This chapter aims to explain the high level concepts in the service architecture, with details to be found in later chapters. This explanation should assist service implementers in understanding conceptually how a service should be built to “plug” into the TINA service management environment. It should assist service environment implementers to understand what control and management “dials and buttons” the environment can expect the service to offer, and what facilities the service needs to operate.

Not all of the concepts have well-defined computational specifications yet. Specifications that are available are given in Section 7 (high level) and in [13] and [14].

3.1 Commercial Environment

The overall objective of the service architecture is to support the most general case of business administrative domains, interacting with one another over a DPE, in order to offer business objects or applications for commercial gain. The following is an example of this commercial environment. In Figure 3-1, a business administrative domain ‘A’ (the 1st party) wishes to make use of a business application offered by a different business administrative domain ‘B’ (the 2nd party). The business application may be provided entirely by B, or B may subcontract it to be (partially or wholly) provided by one or more other business administrative domains ‘C’ and ‘D’ (3rd parties), each making a different and unique contribution in satisfying the A’s request. Domains ‘C’ and ‘D’ may or may not interact directly with ‘A’. These interactions reflect business relationships or contracts, where the nature and direction of commercial gain to each party is unique to the specific occasion and type of business application.

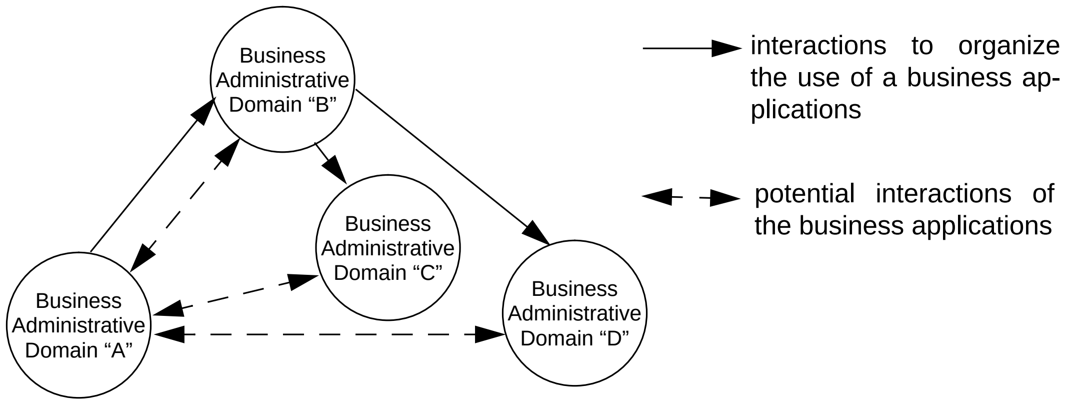

Business Administrative Domain “B”
Business Administrative Domain “C”
Business Administrative Domain “A”
Business Administrative Domain “D”
interactions to organize the use of a business applications
potential interactions of the business applications

Figure 3-1. Example: general model of service architecture interactions between business administrative domains

The model in Figure 3-1 does not assume that business administrative domain ‘A’ always has the same type of interactions with the other domains. It can also act as provider towards other domains at the same time under other contracts. Thus, the relationships are not static, unless a very limited business function is imposed on the activity of a domain. For the specification of limited reference points (as in [3]) such restrictions are useful. The general conceptual ideas in the service architecture are not restricted to this extent. Figure 3-1 only indicates that in explaining concepts such as session, and session roles which are introduced later in this chapter, it is useful to start with a request for a business application and see how service architecture terms and concepts apply.

It should also be noted that a ‘chain’ of business administrative domains can be used to provide services. For example, B used C and D to provide part of a business application. In turn, C or D could use another domain to provide part, or all of the service they are providing to B.

3.2 Service Environment

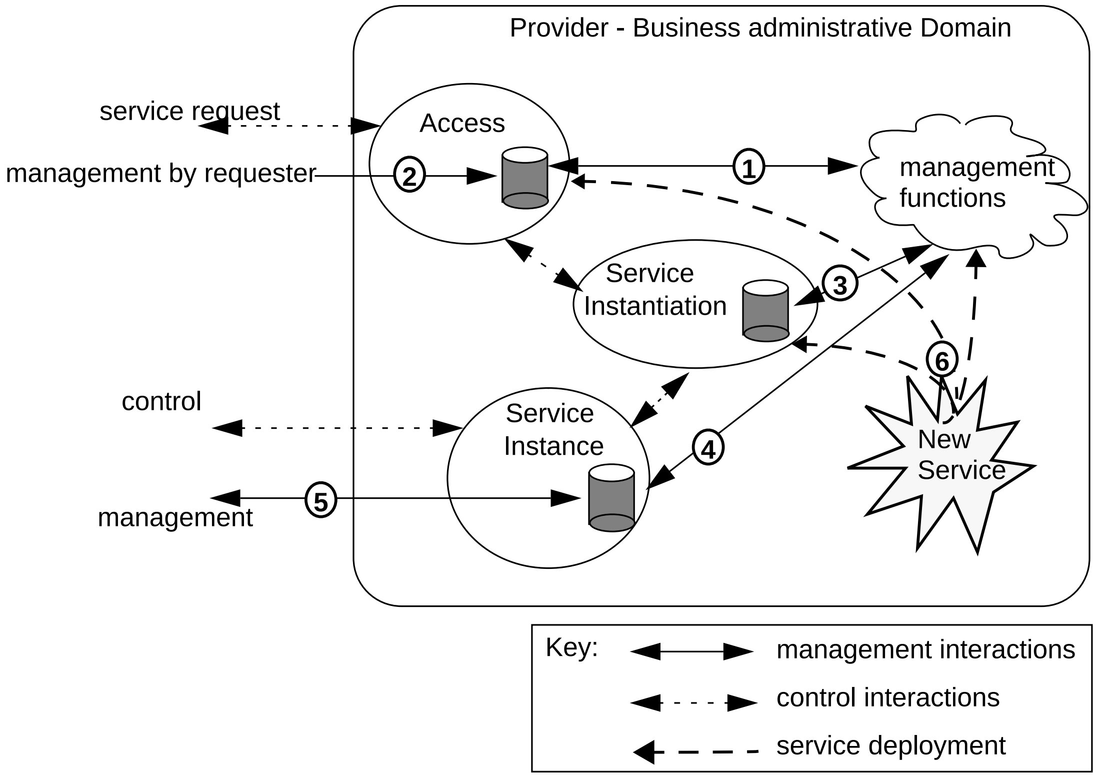


Figure 3-2. Perspective on management in the Service Architecture

Figure 3-2 shows the scope of the service architecture, where a business administrative domain provides access to services (business applications) to requesting domains. The diagram also shows how management considerations can affect the way a requesting domain accesses and uses services. It is important to note that the provider domain must take the following management considerations into account (points below correspond to the numbered arrows in Figure 3-2.):

1. Access - by provider management
   The service requester will be subjected to management policies that limit access to services, or constrain permissions within them. The formulation and application of these policies is internal to a provider business administrative domain.

2. Access - by requesting domain management
   Requesting domains are represented by principals. A principal is an identified entity (e.g.,person) that has the authority to act on behalf of the requesting domain. Typically, subscriber principals are responsible for nominating and administering a group of end-user principals. Therefore, while both end-users and subscribers act on behalf of the requesting domain, the end-user has an even more restricted set of capabilities, which are subject to both provider domain policy and subscriber policies. Access user types are shown in Figure 3-7.

3. Service instantiation - by provider management
   This covers the management policies that dictate where and how a service should be

instantiated when requested. Implementation of instantiation as well as initialization management is internal and specific to a provider domain and is abstracted to a high computational level in the service architecture. Only computational interactions to instantiate a service from a request are considered in the service architecture.

4. Service instance management - by provider management
This covers the interactions necessary to manage running services (e.g. control of its life-cycle and monitoring service activity). Service specific controls are outside the scope of the service architecture. Management output from the service, such as accounting information, is considered within the service architecture. Accounting and billing systems in the service environment are only considered with respect to their interactions with running services. Typically, FCAPS functions are considered in service architecture under the description of management contexts (see Section 5.3.4).

5. Service instance management - by participant
The participant that is interacting with the service may require management facilities that allow him or her to change management contexts, such as accounting practice, security, and performance (for example to constrain other parties involved). Again, these are typically FCAPS functions.

6. New services
An essential service environment issue is that of installing new services. Services have a life-cycle as described in Section 5. To be offered to requesters, a new service must be deployed which takes into account all the above management perspectives. If a service is in compliance with the TINA service architecture, this will be straightforward. The service architecture does not deal with the process of deploying and provisioning of a service. Instead it gives the information models and computational environment to which services must comply in order to be able to be deployed and provisioned rapidly and at low cost.

In addition to the management interactions, Figure 3-2 shows dashed arrows which are control related interactions over the DPE (e.g. requestService). The behavior of such control interactions is subject to the management policies discussed above and manipulated by management interfaces. However, the execution of control and management invocations can be closely related.

3.3 Access, Service and Communication Separations

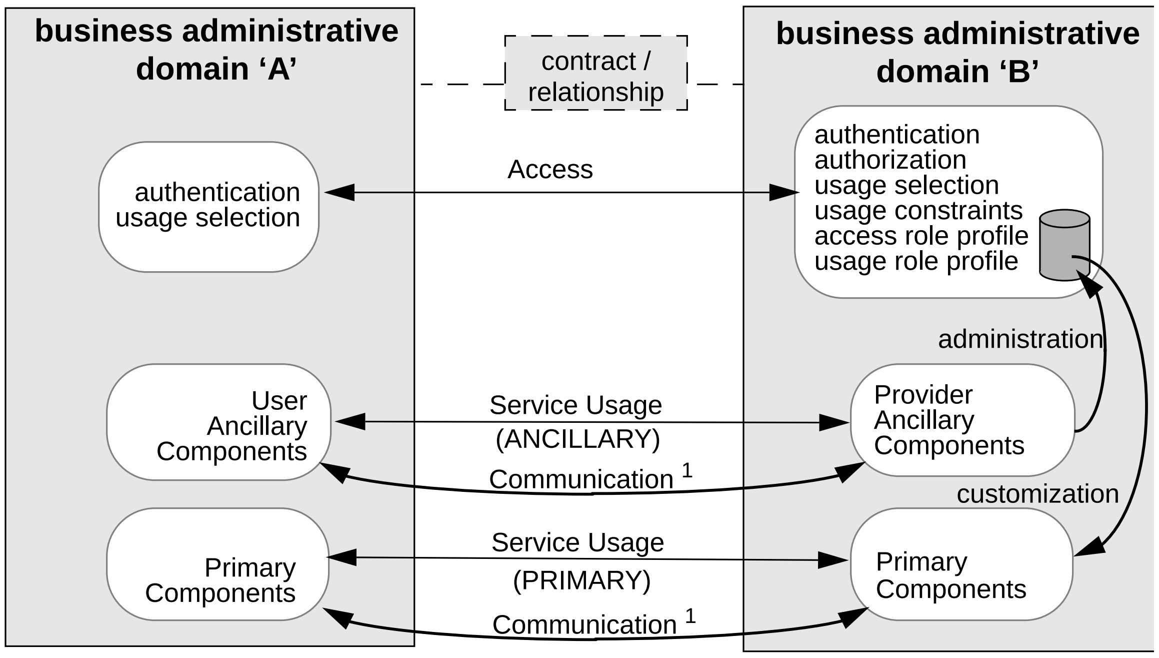
The above discussion and Figure 3-2 introduces a fundamental concept: the separation of access and usage. The interactions required to discover and request services are captured under access. On the other hand, those that control service behavior or deliver stream content are captured under usage. Figure 3-3 illustrates fundamental architecture divisions (which map on to the sessions shown later):

- Access
- Usage
  - Service
  - Communications

Figure 3-3 shows an example where business administrative domain ‘A’ requests a service (usage) relationship with components in domain ‘B’. The example is limited so the discussion can concentrate on what B could offer A to support the request. In addition to these fundamental architectural divisions it is also helpful later in the architecture to consider two different inter-domain purposes of usage interaction (interactions that include DPE control interactions and potential communication stream connections). These inter-domain usage types are

- Ancillary Usage

- Primary Usage.



1. The use of communication sessions are optional, as some services may only need operational interfaces.

Figure 3-3. DIagram showing access, ancillary and primary usage.

3.3.1 Access

The access part covers the interactions required for two domains to establish the usage parts. To assist the long-term relationship, B maintains information about A's principals (who make requests), such as authorization and user designated preferences. The access functionality is kept to a minimum to allow easy implementation at low end of the TINA equipment market. All other functionality is handled as usage (including the re-negotiation of the domain contract, subscription etc.) and can be introduced when needed.

Aims of the access part are to:

- initiate dialogue between the business administrative domains,
- identify the business administrative domains to each other (N.B., either domain can remain anonymous, depending on the interaction requested),
- establish a secure association between the business administrative domains,
- set up the default context for the control and management of usage functionality,
- allow discovery of service[1] offerings,

1. These services can be primary (e.g. Video on Demand (VoD), ancillary to the primary (e.g. configuration management for VoD) or administrative (e.g. subscriber management for VoD).

- initiate usage of services between the business administrative domains according to the agreed conditions.

3.3.2 Usage

3.3.2.1 Service

Interactions amongst components are required to control the behavior of the service. These are diverse and include:

- control and manipulation of the overall service,
- specific control of service behavior or exchange or service content,
- exchange and manipulation of management information.

3.3.2.2 Communications

Interactions are required to establish and maintain stream connections among components implementing the service. This includes arranging a certain QoS, setup and modification of multiple connections which may represent multi-point and multi-media stream binding. In general the impact of consumption of network resources for carrying streams is reflected at the service level and tied into the cost of the service content and functionality.

3.3.3 Types of Inter-domain Usage Interaction - Defined by Purpose.

From the perspective of the requester, the procedures for obtaining interfaces for both ancillary and primary usage are indistinguishable. They are separated here to be explicit about the way TINA anticipates ancillary usage services to be supported architecturally and to avoid their confusion with access functionality.

3.3.3.1 Ancillary Service Usage

These interactions are designed to support or assist the requester relationship (contract) with the provider. Ancillary usage comprises interactions with components that are deployed by the provider that enable the requester to personalize or customize the presentation or employment of provider components to the user; limits are agreed to by the parties. Because these components administer information in the providers domain they are sensitive and therefore distinctly part of the management responsibility of the provider. Since the ancillary usage part does not fulfill the primary contractual purpose of the relationship, it has no independent value, but rather adds value to the primary usage part. The ancillary usage part may modify data and policies that are used for decision making in the access part. Typical examples are subscription services, mobility arrangements and payment/billing services. Aims of the ancillary usage part are to:

- control the life-cycle and attributes of ancillary usage services,
- allow the role user to customize the presentation or employment of the provider domain,
- arrange and potentially execute the deployment of software to the user domain.

Ancillary services are separated architecturally because they are an obvious source of competitive differentiation among providers. Treatment as services allows these to be deployed in a way consistent with the general service environment and be subject to the same scope of accounting and management requirements as primary services.

3.3.3.2 Primary Service Usage

This covers the usage that aims to meet the main objective of the contract between two domains, for example, a multimedia conference, the provision of an information resource component, or a management service. Aims of the primary usage part are:

- control of the life-cycle and attributes of primary usage services,
- interaction with and exchange of service content.

3.3.4 Service Usage Across Multiple Domains

Because of the complicated multi-party relationships involved in TINA, the access and primary usage parts of the interaction can occur between different business administrative domains. Generally, however, the access and ancillary usage parts do not occur between additional parties because the access attributes and ancillary usage functions that modify them are closely bound. There are many examples of these combinations in Section 4 which discusses session composition federation across domains.

3.4 The Session Role Concept

Besides the separation of business administrative domain interactions into access and usage, a separation of the interactions can be made according to the role performed by the domain in the session (see the OMT definition of Figure 3-4).

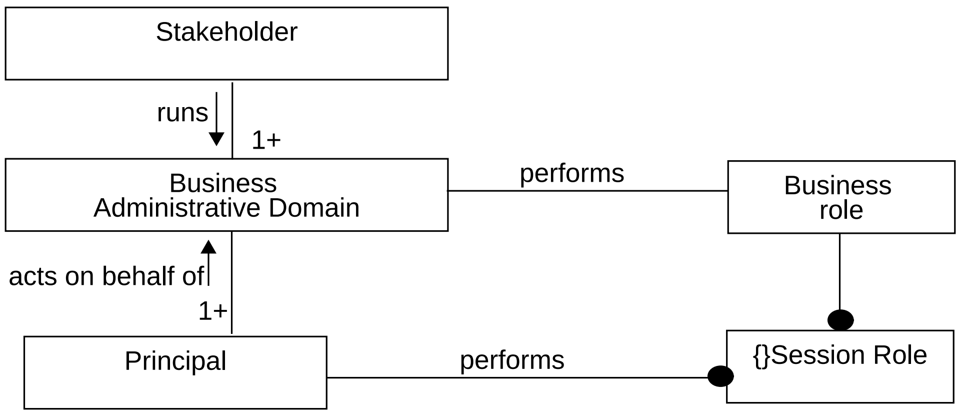


Figure 3-4. Relationship between business role and session role

Domains are able to take different session roles during the access and usage phases of an interaction. Figure 3-5 shows the separation of access role and usage role, with subclasses for each of the user, provider or peer roles. Only the ‘leaf’ roles (at the bottom of the figure) can be taken by a business administration domain. The other roles are abstract roles, and cannot be taken by a domain.

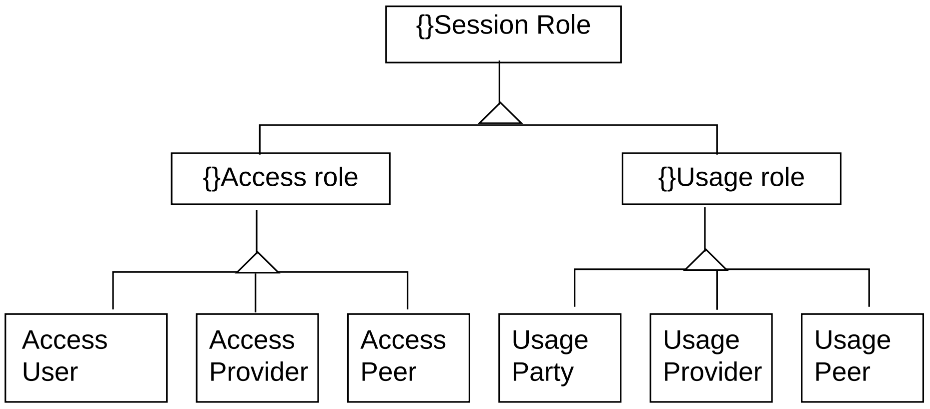


Figure 3-5. Session roles

These generic role types provide structure to the interactions between business administrative domains and their interfaces. At present three types of session roles are identified:

- **user/party[2] role**: supporting the interaction capabilities to request and use services (e.g. subscription where the party requests the subscription, supplies information and absorbs a “token of subscription”),
- **provider role**: supporting the interaction capabilities to supply services (basically the server methods on an object, e.g., subscription, where the provider checks information and supplies a “token of subscription”),
- **peer role**: supporting interaction capabilities that alternate between party and provider (e.g. negotiation of a context where peers bid and counter-bid).

The session roles are dynamically performed by business administrative domains so a domain can change roles as frequently as is required by the interaction. Roles are always complementary and come in pairs, as indicated in Figure 3-6.

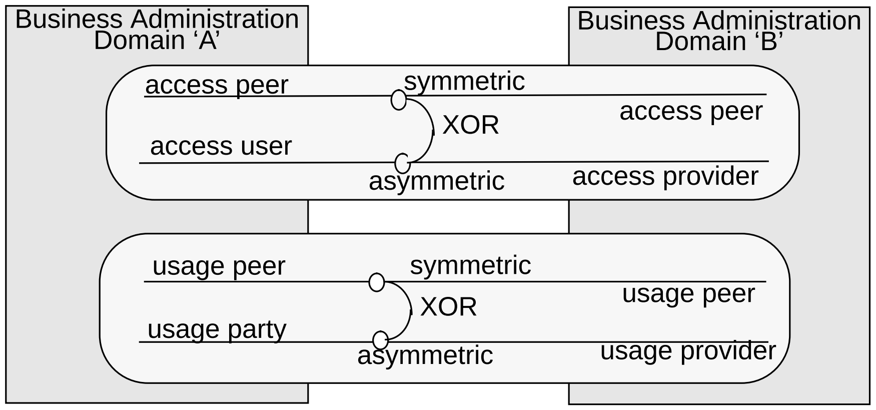


Figure 3-6. Roles between domains in the usage part and access part

2. The term ‘user’ is used for access, while the term ‘party’ is used for usage, due to the fact that ‘user usage’ made sentences too complicated, since these sentences tended to contain the verb ‘use’ as well.

3.4.1 Session Roles in Access

Figure 3-5 describes the proposed access roles that reflect the roles of the principals[3] in the domain.

- **Access user** requests and uses ancillary and primary services and is specialized into three classes shown in Figure 3-7:
  - **end-user** requests services and uses service content. Usage charges may be non (free usage), assigned to subscription contract (part of the overall business role contract) or paid on-line by the end-user. Only an end-user can be invited to participate in a service.
  - **subscriber** administers subscription contracts on behalf of the business domain. Administration includes: contract negotiation, arranging end-users, and setting subscriber constraints on the service to limit end-user capabilities. Subscribers may be responsible for payment of charges incurred by end-users; payment may be on-line or by other means. By this definition a subscriber cannot administer anonymous users. A physical entity, such as a person, may be both the end-user and the subscriber. Alternatively, a business administrative domain may have many subscribers administering a contract to benefit many end-users.
  - **anonymous user** is unknown or unrecognized by the access provider role. An anonymous access user may initiate and utilize services that are free or where payment guarantees will be supplied on-line. By this definition an anonymous user cannot assign usage to a subscriber. By invoking ancillary services (see next section) such as subscription, the anonymous user may become a subscriber and/or end-user. This will require permanent records in the provider domain and a capability to identify the user. This identity would allow the user to act as end-user or subscriber. Anonymous users cannot be invited to service sessions because they do not have resolvable names.

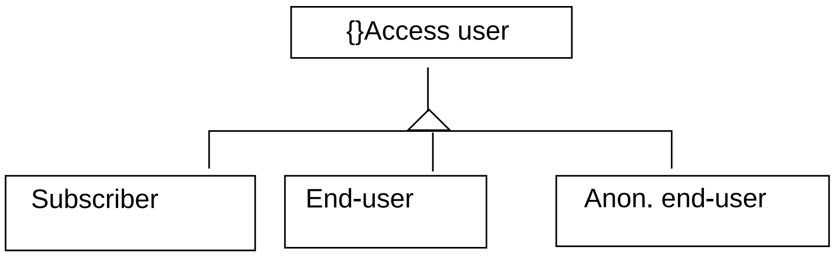


Figure 3-7. The specializaton of the consumer business role

- **Access provider** role is the complement of access user, and offers services and supplies service content. The provider role retains knowledge about users associated with requesting domains, such as subscriptions, user profiles and constraints. No specializations are specified currently.
- **Access peers** may request and offer services to the other domain. The interfaces (but not policies behind them) are identical across the domain boundary. In fact, a subscriber in each domain might administer their users' profiles in the other's domain. The relationship is always symmetric across the domain boundary.

3. Principals are the entities representing a domain that hold an identity that can be authenticated. The identity is used to assign usage to a subscription, for example.

3.4.2 Session Roles in Usage

In the usage part, complementary pairs of roles are peer-peer and party-provider. The roles are independent from the access part to some extent. Access peer roles may instantiate service (usage) components that fulfil the roles of peers or party-provider roles. Equally, user-provider access roles can instantiate usage peer roles.

However, there are dependencies between access and usage roles that apply to some classes of service. For example, a service component in domain ‘A’ that supports the usage peer role may offer a suspend method to a usage peer component in domain ‘B’. However, so that B can resume the session at a later time, ‘A’ will also be required to act in a provider or peer access role because the resume method is only specified in those roles. Hence, in order to offer certain usage roles, a domain may be required to support particular access roles.

The following is an example of usage roles where a domestic consumer requests a service from a retailer. The consumer acts as user in the access part and usually acts as a usage party in services in which the service is supported by retailer domain components. Another example is where two providers regularly interact to supply different usage components at the request of the other, to compose a wide range of services. They may interact in peer or user-provider access roles, depending on the services.

The degree of independence of access and usage roles has an impact on the service component specifications [14]. Currently, specifications are limited to solutions for the access user role and usage user role always together in a single domain, and likewise for the access provider and usage provider together in a single domain. This case defines the Ret reference point[13] together with internal specifications in the consumer and retailer domains.

Despite the above caveats, the computational interfaces themselves are probably still reusable, but they are collected together differently to make components that fulfil the other combinations of roles.

Table 3-1. Examples of how business roles may take session roles[a]

	Access User	Access Provider	Access Peer	Usage Party	Usage Provider	Usage Peer
Consumer over Ret RP	X			X		
Retailer over Ret RP		X			X	
3rd Party Provider over 3Pty RP[b]	X[c]	X[d]	X	X[c]	X[d]	X
Retailer over 3Pty RP[b]	X[d]	X[c]	X	X[d]	X[c]	X
Retailer over RtR RP[b]	X	X	X	X	X	X

a. For further details see chapter 4.

b. Domains may take different roles depending on the action they wish to perform and the role taken by the other domain.

c. This combination is typically used when registering a new service with the retailer.

d. This combination is typically used when providing a service session to the retailer, for use by a consumer.

3.5 Session Concepts

In addition to the access and usage roles that domains take, the interactions themselves comprise a contiguous, related set in time. The history of events creates a particular state configuration of the objects that are conducting the access and usage interactions. The configuration state, and interaction and event history have significance in the service architecture and are embodied in the TINA session concepts. Sessions have various purposes relating to, for example, accounting, fault recovery, component separations, and the common purpose reflected in consistent shared knowledge among session components.

Definition of Session: The temporary relationship among a group of objects that are assigned to collectively fulfill a task for a period of time. A session has a state that may change during its lifetime.The session represents an abstract, simplified view of the management and usage of the objects and their shared information.

Objects in a session are subject to common policies (laid down in the management context) that govern the session, although individual objects may be subject to derived aspects of those policies. Such policies cover issues such as accounting, security, and QoS (see Section 5.3.4.).

Sessions can span multiple business administrative domains. However, since policies are only valid within a business administrative domain, it is useful to define a portion of the session that covers a single domain. The domain session, which holds the policy of that domain, as well as the informational and computational objects operating under that policy, is thus defined.

The objects participating in a session are various and dependent on the nature of the session. Objects in TINA sessions correspond to computational objects and information objects that characterize the state, policies and behavior of the session. Some sessions map on to computational resources in a single business administrative domain, while other sessions cover computational objects in two or more business administrative domains.

TINA sessions are concepts around which the operations and attributes of TINA computational objects have been defined. There is no general one-to-one mapping between TINA sessions and computational objects. Session concepts represent a simplifying view of a collection of resources that come under common management and usage policies. The aim is to hide the complexity of the resources from the management and usage views.

3.5.1 TINA Defined Sessions

The scope of the TINA sessions is simply portrayed in Figure 3-8. This shows how sessions relate in an example of a consumer interacting with another participant within a service session supplied by a retailer. If the other party were a consumer, it could be a video conference; alternatively, if the other participant were a third party content provider, it could be a VoD service. Detailed information specifications of sessions and their relationships are covered in Section 6.

3.5.2 Access Related Sessions

3.5.2.1 Access Session (AS)

An access session is established when two Domain Access Sessions (D_AS) are bound together in a secure relationship (i.e., in a domain session binding). The early stage of the access session is the agreement of terms between domains to continue interaction and authentication of the principals represented in the D_AS. Security protection may be delegated to subsequent service sessions.

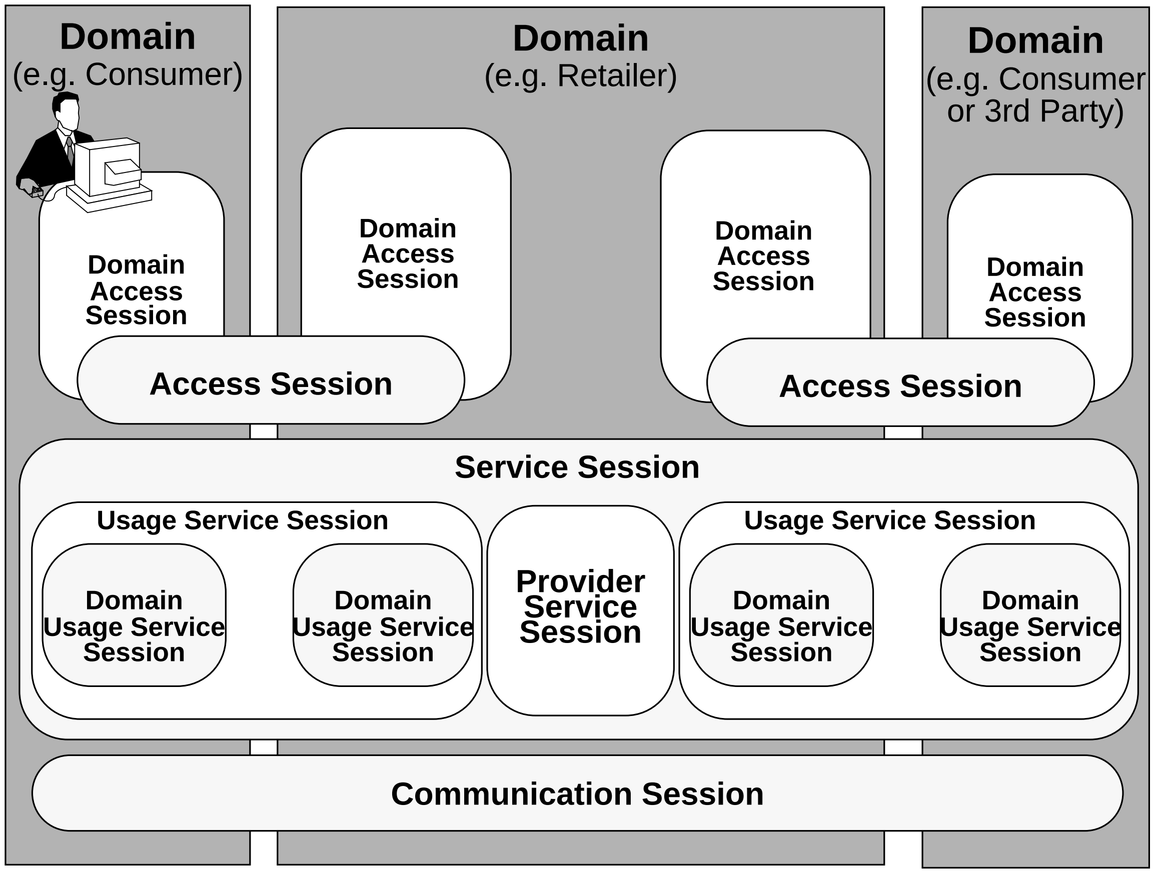


Figure 3-8. TINA sessions superimposed on an example of a business administrative domain model for consumer, retailer and third-party or consumer business roles

Domains that offer services will enforce certain policies in the access session which are derived from the commercial and technical aspects of the contract. Policies are held in the user profile information object, D_AS. Thus, the access session represents:

- **A doorway** to the system of TINA services offered by one domain to another. A requesting domain may request services independently of the service location;
- **Customized access** to TINA services, the interaction between domains is customized by previously arranged principal/role specific profiles, taking into account the end-user's preferences or terminal capabilities. Note that determining the profile is not part of access session functionality, but done in a specific service session;
- **Mobility**, ubiquitous access to the system of TINA services, irrespective of the terminal being used and the point of attachment to the network;
- **Secure** access, the means to create a secure binding between the two domains.

From an access session many service sessions may be invoked, which are the responsibility of the access session until they are terminated, assigned to another access session, or assigned to a D_AS. The access session can be terminated by either domain. The domain access session is an abstract concept, which is further specialized into user--, provider--, and peer--domain access sessions (UD_AS, PD_AS, and PeerD_AS). These specializations support the different access roles discussed in Section 3.4.1.

A user domain access session represents the capabilities and configuration that an end user or some other member of a domain employs to contact a provider. Once an access session has been established, it supports invitations to that member and sends requests to the provider domain. A provider domain access session supports the provider role. It maintains permanent information about a user domain, including identification, and the associate user's capabilities in this domain. A peer domain access session supports the peer role. Thus, It supports a combination of user and provider capabilities, plus capabilities to allow either domain to initiate an access session and to maintain the consistency of the peer domain access session for requests from both domains. Further information models are in Section 6.

3.5.3 Service Related Sessions

These objects and their relationships are shown in Figure 3-8.

The *Service Session* (SS)represents information and functionality related to capabilities to execute, control and manage services. Such services include primary services (e.g. a multi-media conference) and ancillary services (e.g. on-line subscription). The capabilities include service specific control (not TINA defined), generic session controls, and management capabilities. A service session is an instance of a service type and includes information necessary to negotiate QoS, security context, use of service and communication resources, and to control relationships among participating members of the service session. The service session comprises a provider service session, and usage service session(s).

The *Provider Service Session* (PSS) represents the core service logic and control for the one or more domains participating in the service in one role or another. The *Usage Service Session* (USS) represents the participation of other domains with the PSS, and therefore reflects an across domain relationship between two *Domain Usage Service Sessions* (D_USS). The D_USS are specialized according to the role of the participating domains within the USS specializations are: peer (PeerD_USS), composer (CompD_USS), user (UD_USS) and provider (PD_USS).[4]

The UD_USS represents a simple participant in a service. From this user's point of view the (personalized) service logic is supplied by the complementary domain role in the PD_USS. The PeerD_USS complements itself, and is a combination of user and provider roles. Therefore, it is associated with a PSS in each domain. The CompD_USS is complemented by the PD_USS and is applied in service session chaining for composition scenarios where one PSS requires services from a subsidiary SS. More details on composition are in Section 4.

If a service session is the responsibility of an access session, the service session can remain active while that access session is active. When that access session ends, the service session must be assigned to another access session (possibly a different participant) or a provider domain access session (same or different user). In either case, when the access session is ended, the related usage service session must be suspended or ended.

3.5.4 Communication Session

The *Communication Session* (CS) represents a general, service view of stream connections and a network technology-independent view of the communication resources required to establish end to end connections. A communication session can handle multiple connections which may be multi-point and multi-media.

4. Though it is stated that PSS represents the core service logic, the service logic may be distributed among several service provider domains. This is achieved through PeerD_USS and CompD_USS, as explained in Section 4.

A communication session can arrange QoS, set-up, modify, and pull-down multiple connections. It can also interact with multiple connectivity sessions (that exist in the connectivity provider domain) as explained in NRA [8]. The communication handles connectivity provider interactions for a service session.

The adoption of the "session" concept for controlling communication capability has the advantage of allowing services to instantiate dynamically, hold, resume and maintain a suitable configuration of communication resources that satisfies their needs. A more complete description of the communication session and its capabilities is part of the TINA NRA [8].

A communication session is controlled by one service session from the PSS, PD_USS or Peer_USS. Only one service session may be associated with a communication session at any one time.

3.5.5 Lifetime dependencies between sessions

The following general principles apply to the lifetime dependencies between sessions:

- A service session cannot exist without the access session of the party holding the ownership of the service session (N.B., ownership can be transferred between session parties).
- An access session can encompass many service sessions.
- A service session can encompass many communication sessions.
- A communication session cannot exist without a service session.
- A communication session can only be controlled by a single service session to avoid control conflicts.

This can be illustrated with reference to Figure 3-9 which shows the activities of two users, Micky and Minnie. Micky utilizes three service sessions at various time periods during his access session. During service session 3 (SS3), Minnie is invited to join and share his enjoyment. Although Micky started the shared session, he decides to leave and hands over the control rights of the session to Minnie. He ends his access session, terminating his involvement in the session.

Because the service sessions require stream connections at various times according to the services' specialized needs, the service sessions request communication sessions that are supplied by provider Y. The communication sessions are requested and ended according to the service needs as the service session progresses. For example, Micky needs to download software to browse the services provided by X and does so by the communication session 1 in Y. He then starts viewing the show in a separate service session (SS3) and decides to suspend his viewing and to invite Minnie. Service session three will keep existing while the communication session (CS2) supporting the viewing is released. Minnie needs to download special software as well, which is done through communication session 3. In the meantime, Micky has resumed viewing (N.B., a new communication session (CS4) is created, which is now a 3 party session) and once Minnie is ready, she joins the communication session CS4 with Mickey. He then transfers ownership of service session 3 to Minnie and leaves the session. After a while, Minnie terminates the session.

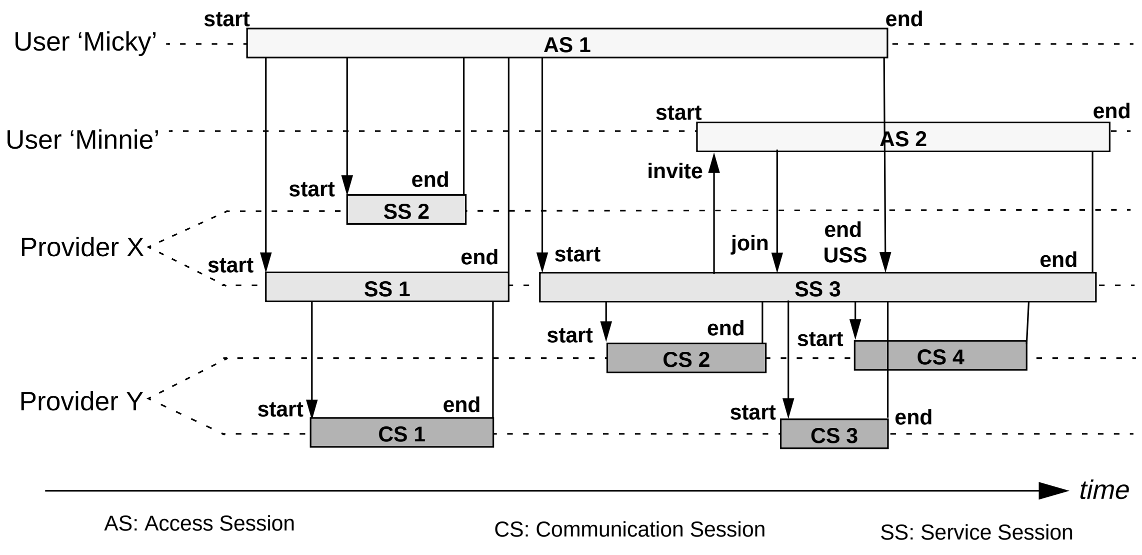


Figure 3-9. Lifetime dependencies among sessions

3.6 Mapping Session Roles and Sessions to Interface Levels

This section explains how the abstract concepts of role and session are mapped on to the more detailed information and computational viewpoints. In general, to ensure the ubiquitous ability of domains to interact and for components to be reusable within domains, there must be common interface specifications. These Interfaces are aggregated into computational components that implement the higher level concepts, such as a role within a session. The interfaces manipulate or examine information objects that are related to each other by information models. This aggregation of interfaces into a component ensures the semantic understanding that operations at one interface may affect the behavior at other interfaces because they may be linked by a common, underlying information model held by the component. The information models obviously reflect the parameters and semantics of operations found on the interfaces.

Thus, although a UD_USS computational component could not support a CompD_USS, some of its interface types are reusable. This is true because some of the information models can be reused and, hence, also the interface(s) that manipulate or examine these information models.

Detailed information models can be found in Section 6 and computational component solutions for some of the abstract session and role concepts can be found in Section 7. The following sections aim to explain some basic terminology and lower level concepts used in these sections as well as in Section 4, "Composition and Federation".

3.6.1 Information Models

The architecture has so far considered information models for the access session and the service session. In addition Section 6 describes the service session information model, called the service session graph. The service session graph relates members (whether they are resources or parties) together with various relationships, such as control, ownership or stream bindings.

In addition to these information models, there are management contexts which relate to typical FCAPS functional areas. Management contexts describe the agreements between components supporting access or service sessions (whether those agreements are by default or negotiated). The management contexts considered in the architecture are:

- Accounting (charging arrangements, policy, events etc.),
- Security (quality of protection, policy, events etc.),
- Performance (QoS, degradation alarms, etc.).

In many cases the entirety of the information models exceeds the particular requirements of specific services. For example, compare a simple, single user retrieval service with a multi-media, multiparty conference. To support this range of functionality in an extendable fashion, the concept of *feature sets* is introduced to structure related groups of interfaces.

3.6.2 Interface Structuring

The service architecture structures interfaces according to the separation principles described previously. Interfaces support a particular session role, and so there are interfaces associated with the access user role, usage provider role, etc. (usually multiple interfaces support each role). Interfaces associated with each role are defined by the reference point specifications. So here is described the structuring of the interfaces only.

Access related interfaces are associated with the access role taken by the domain. Interfaces are defined for a domain taking the access user, and access provider roles. (These are then specialized for the definition of the Ret RP, and can be reused for other reference points). Access related interfaces are, in general, mandatory, so if a domain wishes to take a particular role it must support all of the interfaces defined for that role.

Usage related interfaces are similarly associated with the usage role taken by the domain. However, many of the interfaces are optional. This allows session components to tailor the interfaces they support to the requirements of the services, e.g., components of simple single user services are not burdened with unnecessary interfaces required for multiparty services, and components capabilities can be “expanded” according to the needs of a service.The service architecture uses two concepts to represent this optionality: session models and feature sets.

Session models define how service session components in each domain can interact in a generic manner. A session model defines an information model for the session, and relates how operations on interfaces affect the information model. Sessions may support one or more session models to describe the behaviour of their operations.

TINA defines a single session model, the TINA Session Model. This model allows service session components to make requests about ending and suspending the session; the parties involved; set of stream bindings between parties, for example. The TINA Session Model uses the session graph as the information model to describe the behaviour of operations on its interfaces.

The TINA Session Model groups interfaces into feature sets. A feature set is a group of interfaces that exposes restricted parts of the information model for manipulation or examination. If a feature set is supported by a domain, the domain supports all of the interfaces in the feature set, which are associated with their session role. In general all domains in the session must support the same feature sets.

Additionally, the basic features in both access and service session interfaces allow business administrative domains to exchange and agree the use of additional non-standardized interfaces. This is essential to obtain service specific interfaces and value added interfaces above or instead of

TINA standard session control. To illustrate this structuring by example, consider the Ret reference point specification [13]. Three of the several feature sets are given below. Details of all the feature sets currently defined are given in Table 7-2 page 112.

- BasicFS (simple fundamental session control, e.g. end and suspend)
- MultipartyFS (control for multiparty services: invite a party, suspend a party, get information about parties, etc.)
- ParticipantSBFS (enables participants to set up stream bindings)

In addition feature sets for manipulating management contexts will also be defined. Management context interfaces are used to manipulate and examine the management context information models. They will include facilites for accounting and security. More details on management contexts can be found in Section 5.3.4

3.7 Summary of Concepts

Figure 3-9 illustrates the mapping of concepts discussed in this chapter. To sum up, these include: the separation of interactions into access and usage (and usage into service and communications), the associations of these interactions to specific sessions within each domain and across domains, the complementary roles the domains may play in those sessions, and the implementation of the sessions by computational components.

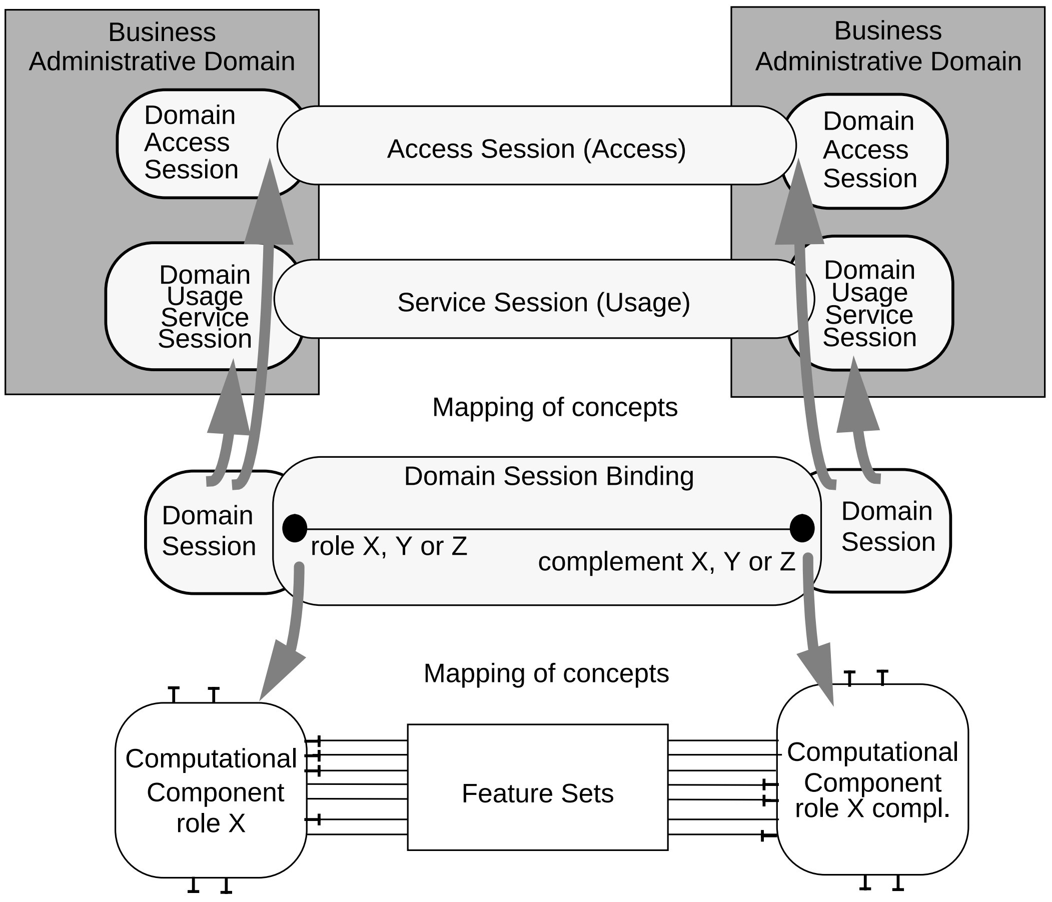


Figure 3-10. Mapping of the concepts sessions & domain sessions onto roles, onto computational objects and the feature sets and management contexts for roles

4 Composition and Federation

This chapter scopes the service composition and federation support of the TINA service architecture. It introduces the principles, requirements, and supporting concepts that form the basis of composition and federation. Finally, it outlines paradigms that can be used to compose or federate services.

4.1 Definition and Scope

4.1.1 Composition

Service Composition concerns the creation of a new service or service instance by composing services or service components. Many types of composition are possible. This section scopes the composition problem in order to address service architecture issues. Figure 4-1 illustrates service composition showing a meeting room service example[1]. This service is providing all the facilities required for remote meetings. It is based on a video conference service and is composed of a number of facilities (component services[2]).

Two end-users[3], Bruce and Sheila, are attending the "Electronic Conference for Environment

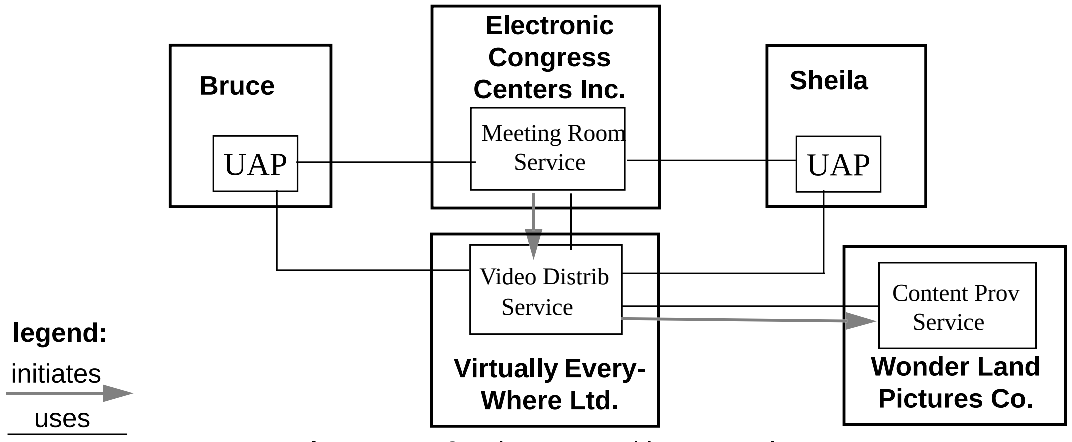


Figure 4-1. Service composition example

Care in Central Africa". Both have joined a meeting session. During the session Sheila suggests showing a film on "Wild Life in Central Africa" to illustrate her speech. On her request, the meeting room service session locates dynamically a service provider offering a video distribution service. During the access session between providers (Electronic Congress Centers and Virtually Everywhere), the terms of the relationship between the services are defined; the meeting session will act as a controller party in the video distribution session, and this one will inherit all the meeting session parties, acting as destinations of the video stream. Then, the video presentation session is started. The video distribution session searches for a content provision service with that movie (using its own search mechanisms or some additional service). Once located (Wonder Land Pictures), it is invited to join the video distribution session as the source. An access session is created to establish the terms of cooperation between services. Then, the content provision session joins the video distribution session as a party, acting as the video stream source.

1. The diagram does not follow any standard notation, but is a simple high-level representation.
2. Although only video presentation is shown in the example, the set of component services could be wider, including document distribution, slides presentation, shared whiteboard, etc.
3. Of course, some other users are supposed to participate in the session, but they are not shown for simplicity's sake.

This example or slightly modified/expanded versions will be used throughout the remainder of this chapter[4].

In this document, only service level composition is considered. This includes the composition of service sessions, service resources, or service components. Impact of composition on the communication session requires further study, but a communication session may be considered a specialized service, offering a standard set of interfaces to its users (the service sessions). Service specific issues are not addressed. The TINA service composition framework is designed to be general yet extendible: that is, adaptable to the specific circumstances of particular services. It is based on a set of generic paradigms that can be specialized or extended for specific services.

Composition focuses on the provider viewpoint, as consumer composition is largely application specific and so not within TINA's domain. The visibility of services, service components, and resources to the consumer determine their ability to compose services. Visibility is dependent on DPE and retailer location services. It is assumed that users can locate and establish service sessions, run multiple service sessions concurrently, and nominate management and usage requirements. Only a subset of the composition paradigms are applicable to consumer composition.

Finally, composition focuses on run-time support. This includes support for both static (predefined) and dynamic[5] composition between service level components[6]. To do this, it builds on the service architecture, rather than defining generic components or specific mechanisms for supporting dynamic composition such as interpretive languages.

We assume that services and resources are already deployed. Deployment is considered in Section 5.4.3.1 and Section A-3.3 of the annex. Once a service, resource, or component type is exported, it is assumed to be deployed and ready to use.

We assume that the DPE provides support for composition, including interworking (or federation) of DPEs and the availability of standard DPE services, e.g. trading services. Other run-time support is provided by the service architecture, including access and communication support. Other support services can be considered, such as location, directory and type browsing services. For static composition, such services may be part of the service creation environment. For dynamic composition and run-time support, however, these services should be available at run-time between domains.

4.1.2 Federation

In this section, we consider two distinct aspects of federation at the service level.

- **Domain federation** establishes an environment for two or more service providers[7] to offer a variety of services across their domains in a transparent manner. This is achieved by setting up an access session and negotiating a *federation contract* (Section 4.3.2.1).
- **Service federation** is mainly concerned with the establishment of service sessions across two or more retailer domains. A service federation is governed by a domain federation; that is, it should follow the terms agreed upon in the federation contract.

Domain federation is supported by the access part of the RtR reference point, whereas service federation is supported by the usage part[8].

4. The above paragraph style will be used throughout this chapter when the example reoccurs.
5. The service sessions or components included are determined during the service session life-time.
6. These components are assumed available. Service creation is not covered.
7. Service provider denotes either a retailer or a third party service provider.
8. This reference point is not yet defined.

Some of the principles and requirements given in Section 4.3 apply to both domain and service federation; in these situations *federation* will be used without qualification.

Consider the same example as for composition, but now consumer Sheila is subscribed to the meeting room service through the retailer 'Global A/V Services Ltd.'. This scenario is shown in figure 4-2. Assuming that Bruce has started the session he invites Sheila (whom he knows

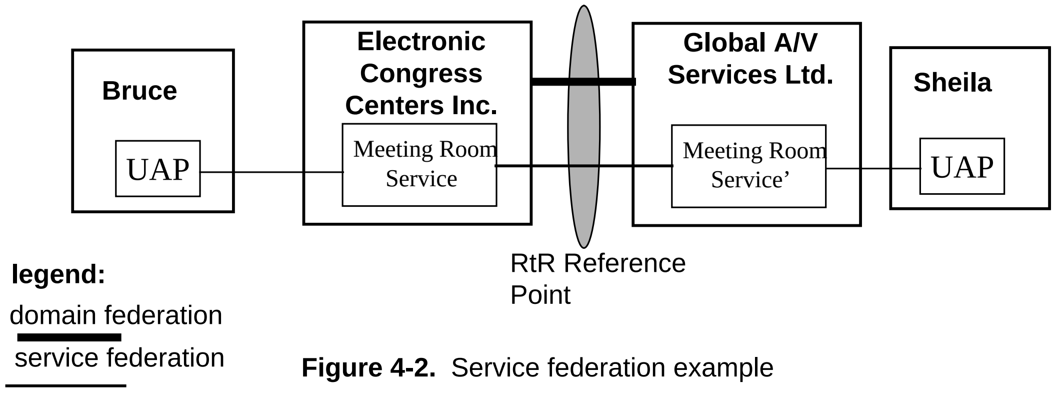


Figure 4-2. Service federation example

is subscribed to Global A/V Services) to join the session. To fulfill that request, Electronic Congress Centers contacts Global A/V Services to set up an access session in order to negotiate and establish a domain federation with the goal of providing a federated meeting room service. Subsequently, Sheila is invited and accepts the invitation. A service federation is then established to provide the actual meeting room service session.

4.1.2.1 Domain federation

Domain federation is required whenever more than one retailer is needed for the provisioning of a service. An example is a situation in which a consumer invites another consumer from a different retailer domain. In this case, we must provide a framework for enabling the involved retailers to provide the requested service in a transparent manner to the consumers, i.e., without the consumers noticing, that a federation is actually taking place. In some cases, provider capabilities may limit transparency.

The federation concepts described in ODP are used here to define the meaning of domain federation in TINA:

Federation is a peer-to-peer relationship between two or more partners[9] involved in order to achieve a common goal. It is characterized by a community that includes all partners involved, a contract defining the terms of the relationship, and a policy to rule the relationship's life-cycle[10].

Domain federation may also support other goals, such as passing invitations and personal mobility[11].

In general, domain federation includes service, DPE and network domains. However, only service-level domains are considered in this document.

Only domain federation initiated by the service provider is foreseen at the service level. Even though the need for the federation arises from a user request, the involved providers are not expected to grant users the necessary control to negotiate and setup the federation, as federation represents a

9. When the term "partner" is used in this chapter it denotes a service provider participating in a domain federation.

10. This is the general ODP definition; in this section we will focus on the common goal of providing a service or service functionality across several retailer domains.

11. Mobility support has not been elaborated yet.

(contractual) relationship between service providers. Furthermore, this federation approach is generic and applies to both on-line and off-line federation contracts, with the main difference between them being the duration; typically it is finite in on-line contracts and unlimited for the off-line ones.

4.1.2.2 Service federation

A federated service is offered, which joins two or more peer services or service components across several or more retailer domains, between parties who have subscribed to different retailer domains. The federating services are of the same type or compatible types. A service federation is governed by a domain federation.

4.1.3 Relation of Composition and Federation

Service composition and service federation are closely related and share a number of basic principles. The most important difference is that service federation focuses on how users associated with different retailer domains share services. As a result, it has more specific goals and requirements than composition[12]. The service federation may be considered a specialization of composition, that builds on the service composition framework for its own, more specific, purposes. In contrast, service composition aims to be as flexible and open as possible, to allow different types of combinations of services and components to cope with the different (and evolving) business relationships amongst service providers. Service composition often relies on some type of service level domain federation.

4.2 Basis for composition

4.2.1 Principles

TINA service composition is based on the following four principles. These principles support service level composition: that is, the composition between service sessions, or service components and resources that provide services. Composition may take place over multiple domains. This composition framework supports multiple composition paradigms in an extendible manner.

1. **Identification and Location between Domains**
   Users and compatible services, components and resources can be located between domains;
2. **Separation of Access and Usage**
   Services may act in different roles during access and usage stages of a composition;
3. **Consistent Management of Compound Services over Multiple Domains**
   A compound service maintains a consistent management environment;
4. **Session Relation and Interaction Support**
   Support is provided for the many different ways sessions can relate and interact.

We will now discuss the requirements arising from these principles. Some are supported by the TINA service architecture, while others, such as identification, need to be supported by additional infrastructure. Architectural support is noted where relevant.

Some of these principles are illustrated in the example in Section 4.1.1:

> *(1) The meeting session locates the video distribution service, based on the service type, and the latter locates the content provision service, based on a specific resource type. (2) The meeting service acts as a user, requester, in the access session with the video service, and as a party, controller, in the usage session. On the other hand, in the composition between the video distribution and the content provision services, the former acts as initiator, user, in the access ses-*

12. The requirements and goals are restricted to the ones of peer type relationships between federated services.

sion, but as provider in the usage stage. (4) In the relationship between meeting session and video session , the former requests the start of and controls the latter (control type composition, see Section 4.2.3.3). The video distribution service is acting as a provider to the content provision service, although the former, in the role of access user, has invited the latter (usage provider type composition, see Section 4.2.3.2).

4.2.1.1 Identification and Location between Domains

Services, service sessions, resources, service components, and end-users need to be identifiable between multiple domains. Thus, type descriptions that allow a service to find compatible components are required. Location mechanisms need to support instance and type naming, and type descriptions.

4.2.1.1.1. Naming

To make composition possible, agreed naming schemes, for end-users, services, resources and other service components are necessary for both types and instances. A naming framework is outlined in Section A-2 of the annex. From the point of view of composition, naming schemes need to provide universal type naming (i.e. common types between domains), support naming of instances within domains, allow names to be used across multiple domains, and be extendible.

Instance and type naming issues also relate to various components of the management and composition framework. In particular, types and instances of contexts[13] and types of roles and feature sets (and hence composition paradigms) require identification. Contexts, which describe a relation between two domains, need only be identified within the relation of the two domains. Some contexts may be general to a service type and may be named in relation to the service. However, role and feature set types need unique identification across multiple domains.

4.2.1.1.2. Type descriptions

Service type descriptions are used to describe the attributes or features of a service which may be used as criteria for selecting a particular implementation or instance of a service or resource. For service composition, it is useful to be able to describe services in terms of roles and feature sets that the service supports. These attributes allow a service to determine whether another will support a particular composition paradigm and/or to export what types of composition it can support.

4.2.1.1.3. Location services

Location services are required for composition. The wide spread use of a few types of location service will simplify implementation, but composition does not require a particular location service. Rather, it needs location services that support the following requirements:

- identify and locate service, component, and resource types and instances, and users,
- support type descriptions that include relationship (role and feature set) information,
- locate a standard access point[14] related to the service(resource) type or instance,
- identify end-users and locate a related access session (either of the end-user or some associated retailer domain),
- direct access to service sessions or components is optional; it must be possible for a service to determine when it has been given direct access.

13. See Section 5.3.4 for the definition of management context.

14. The terminology access point is used to indicate access either within a single domain or between domains. When access is between domains, the access point is typically an access session.

4.2.1.2 Separation of Access and Usage

The clear separation between access and usage roles provided by the service architecture allows very flexible support of service composition. In particular, it is possible to separate a service session's access and usage roles. For instance, an initiator (access user) can act as a usage party or provider.

Also, it allows us to standardize access, which shields services from the complexities of instantiating other sessions or components and supports access between multiple administrative domains. This aids reuse and makes it easier for service sessions to interact with many different components and resources. We assume the standard access provided by TINA access sessions (see Section 6.2.2 and Section 7.2.5), which support:

1. common access mechanisms (i.e. invitation, start, join),
2. requests for particular types and instances of services, components or resources,
3. associate management requirements (context) with service session requests,
4. associate usage relations (roles and feature sets) with service session requests.

The access mechanisms available depend on the access roles supported in a particular domain. The reference point between domains determines the access role. In fact, access sessions may be asymmetric or symmetric. However, symmetric access allows more flexible composition arrangements.

4.2.1.3 Consistent Management of Compound Services over Multiple Domains

It must be possible to define a management environment for a service session and maintain a consistent environment across different domains. The management environment needs to be able to support requirements from a number of members of a session. These requirements need not be the same, but they must be consistent.

Management contexts (see Section 5.3.4) are used to describe management requirements between entities. Contexts agreed between two domains must be consistent with their contractual relations and management policies. A context can be selected or modified at access. Service transactions (see Section 5.4.2) associate management contexts with a service session, and allow a consistent management environment to propagate over multiple domains to support a service session.

From the composition point of view, context semantics should support multiple sessions involved in an overall service session. Various service sessions may play different roles in the overall service and so have different requirements. For example, a service session may want to nominate one of its managers (another service session) to play a management role in another session. This would allow a compound service with many components to be managed as a single service session.

4.2.1.4 Session Relation and Interaction Support

Services need to be able to relate and interact with each other in many different ways. These relations may be described in the context of composition paradigms which may constrain the interactions and behaviour between various parts of a compound service. These paradigms are based on sets of roles. Support for multiple composition paradigms and the many possible relations between services and components is required. Some generic composition paradigms are described in this section. These may be extended or specialized for particular services, see Section 4.2.2.3.

4.2.2 Supporting concepts

The previous requirements are supported by a few key concepts, including roles, feature sets, and management context. Some of these concepts have been previously introduced in Section 3.4 and Section 3.6 and are further elaborated in Section 5, Section 6 and Section 7. Here we will briefly consider aspects of some of these concepts particular to service composition.

4.2.2.1 Roles

Section 3.3 defined the concept of role, which described relations between sessions, and introduced a set of access and usage roles which are generic. These generic roles can be used directly or specialized to describe each particular composition paradigm. Some specialized roles, applicable to usual composition relationships (see Section 4.2.3), could be considered:

- *initiator:* Aservice session role related to the initiation of a service session or composing relation. The complement of this role is supported by an access point. An initiator may select or negotiate the management context and nominate entities as members of the session, choosing the role and feature sets. It need not participate in the session itself.
- *nominated*: The service session that is the target of an initiation (i.e., access) request.
- *specialized (privileged) usage party*: Some services may support a number of different party type roles, e.g. a game may have *participants* and *spectators*. These are derived from the usage party role and may be associated with specific feature sets or privileges. They may be used instead of the generic usage party role in composition paradigms.
- *specialized usage provider*: Complement of the specialized usage party.
- *controller*: A specialization of the usage party role, which allows one session to control others and coordinate composed sessions. They have privileges to add, modify and delete any members of the controlled sessions. However, they may be restricted in other areas. For instance, a controller may request stream bindings but not participate in one.
- *controlled*: The complement of the controller role.
- *manager*: A specialization of the usage party role for management. A session in the manager role has access to management related information and may take management actions on the managed session. It is associated with management related feature sets.
- *managed*: The complement of the manager role.

This list can be extended, i.e. roles can be further specialized and combined, as new commonly applicable roles and paradigms are identified.

4.2.2.2 Feature Sets

Feature sets have been introduced in Section 3.6. Support of composition may lead to the definition of new feature sets. Here are some feature sets that might be considered:

- *Multiparty control* feature sets: allow a controller to join members to another session and manipulate them. Controllers must support this feature set.
- *Controller split and merge* feature sets: allow a controlling session to establish control of another, and divest or terminate control of another session.
- *Peer merge* feature sets: allow sessions to share information with each other and establish composing relation between members.
- *Peer split and integration* feature sets: allow the sharing of information and transfer of resources and components between sessions.

- *Management* feature sets: feature sets associated with FCAPS functionality or negotiation of management context.

4.2.2.3 Composition paradigms

Composition may be described in terms of composition paradigms. A paradigm is described by a particular combination of relations (described by roles) between components. By different combinations of the various access and usage roles, we can achieve different paradigms. Some generic paradigms are outlined in Table 4-1 and detailed in Section 4.2.3. By introducing new roles and combinations, new paradigms can be supported. General paradigms are not constrained to a particular set of functionality. To be implemented, however, a paradigm needs to be associated with particular functionality, which is described by feature sets. By aggregating feature sets that support generic (and special) roles, these composition paradigms can be used for a wide variety of generic and service specific capabilities.

Table 4-1. Summary of Generic Composition Paradigms

Paradigm Type	Service's Access Role	Usage Role	Description
Usage party	Initiator (access user) Nominated	Usage Party Usage Provider	Allows one service session to use another like a consumer
Usage provider	Initiator (access user) Nominated	Usage Provider Usage Party	Allows a session to request another to join, acting on the session like a party
Controller	Initiator (access user) Nominated	Controller Controlled	Allows one session to coordinate a compound session
Manager	Nominated Initiator/nominated	Manager Managed	Allows one service session to manage another
Peer	Initiator/nominated	Usage Peer	Supports federation and peer type composition relations

The next section will introduce various general composition paradigms. These cover the most usual service composition cases. Nevertheless, it is necessary to be able to specialize paradigms to implement composition behaviour for a particular service and to allow the extension of paradigms to support specialized or new composition types. The paradigms can be extended by adding new roles (generally usage roles) and combining different usage and access roles. These roles may be specializations or combinations of existing roles or entirely independent of them. They may support generic or service specific behaviour.

In general, to use a role, it needs to be associated with one or more feature sets. Basic feature sets[15] support user and provider roles. By introducing new feature sets, and defining the interfaces associated with particular roles, it is possible to extend existing paradigms to new services.

To allow the use of specialized or new roles, a category of roles known as "special" is allowed. An example of a "special" role could be synchronization. Currently, synchronizing roles are undefined as they seem too service specific, as they relate the delivery of different services to some party or resource. General relations that can be used as the basis for new roles include the following:

- Client-Server: The client makes the requests and the server responds.

15. Basic feature sets (in plural) must not be confused with BasicFS (in Table 7-2).

- Peer-Peer (Symmetric): Each party has the same ability to act on the other.
- Asymmetric: Both parties can initiate actions, but these may not be the same both ways.

4.2.3 Generic Composition Paradigms Description

This section will introduce a variety of generic composition paradigms and describe related access and usage roles. These paradigms will include simple user provider relations, peer type relations and access based relations. Paradigms can be extended by combining roles with particular feature sets or introducing new roles. This section will also discuss how new paradigms can be introduced.

4.2.3.1 Usage Party Type Composition Paradigms

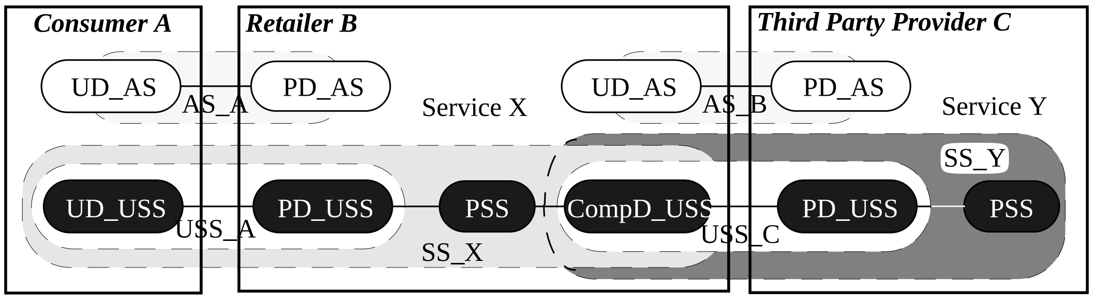


Figure 4-3. Usage party type composition paradigm

In the usage party type composition paradigm, one service uses another in the same way a consumer usually does. In Figure 4-3, the initiating service session, SS X, uses start or join requests on the access point to initiate a service session SS Y, in which it assumes the usage party role. The scenario in Section 7.4.7 also illustrates this paradigm. Generally, the initiator (SS X) adopts the usage party role, but it could nominate another entity, e.g. an end-user or third session, to this role. The usage party and usage provider roles are associated with the same feature sets.

The usage party service sees the subsidiary service as a resource and has access to related service information, including its resources and other parties (if any). In the session graph model, the usage party service can use composing session relationships, see Section 6.4.6, to associate resources with the subsidiary service session. A composer usage session component, see Section 6.3.2, supports interactions between the two services. The usage provider service sees the service in the usage party role as a normal party: i.e. the usage party service becomes a party of the other service.

This paradigm may be applicable to cases like the relationship of a service with a directory service (acting as a broker), to locate other services or resources in the information network, or with any information retrieval service. A hotel chain accommodation browsing service, using a number of individual hotel accommodation browsing services, may be an example of the latter.

4.2.3.2 Usage Provider Type Composition Paradigms

In usage provider type composition paradigms, the initiating service (or its nominee) acts in a usage provider role. This paradigm is related to the previous one, except that the initiator (access user) acts in the usage provider role, while the access provider acts in the usage party role. Figure 4-4 shows an example of a provider type composition. In this case, SS X acts as the initiator and usage provider,

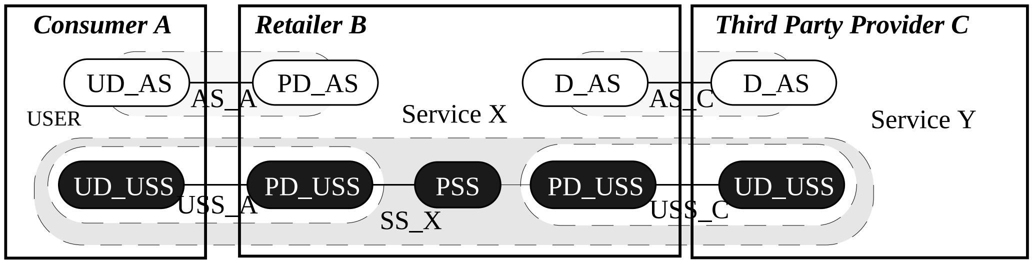


Figure 4-4. Usage provider type composition.

while provider C's service acts as the usage party. Though provider C's service is shown as a normal party, it could be a service session in its own right. The scenario in Section A-4.1 of the annex also illustrates this paradigm.

The initiator could use either invitations or requests to initiate the composition. If invitations are used, the access session needs to be able to initiate service sessions in response to an invitation. Alternatively, the initiating service needs to be able to nominate the requested service in the usage party role.

As before, the usage provider service sees the usage party service as a party, while the usage party service sees the usage provider service as a resource and has access to service related information.

This paradigm may be used as a content provision model, in which service Y (Figure 4-4) represents a content provision service, as shown in the Section 4.1.1 example. For this case, generic roles can be used or specialized roles, such as the content provider role, can be defined.

4.2.3.3 Control Type Composition Paradigms

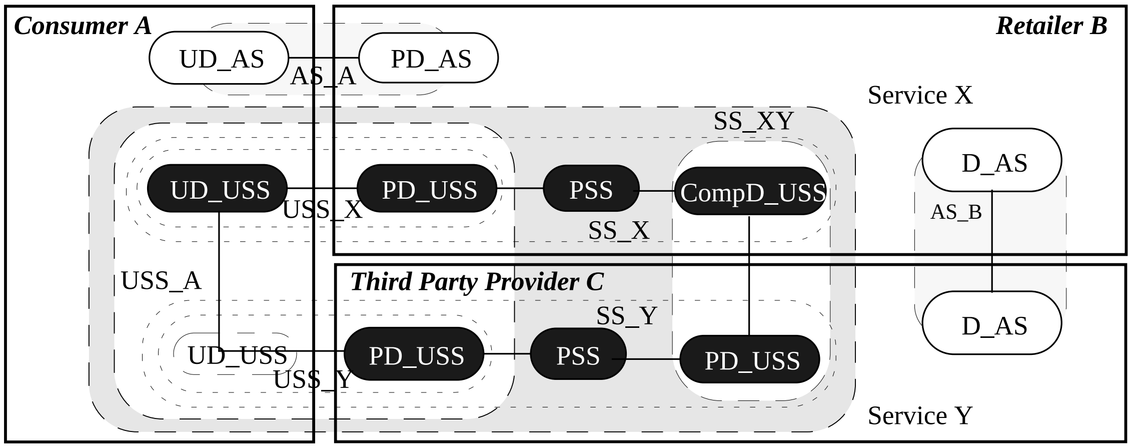


Figure 4-5. Control type composition paradigm.

Control type compositions are closely related to usage party compositions. In a control type paradigm, the initiator (or its nominee) forms a control association with a subsidiary service. Unlike the usage party paradigm, the subsidiary (or controlled) service also forms usage relations with other entities, e.g. acting as a usage provider. This gives us a *parallel* style composition shown in Figure 4-5.

SS X acts as the controller and SS Y acts as the controlled (or subsidiary) service. Both service sessions establish an association with the consumer, though not necessarily of the same type with the same feature sets. The controlling session may add or remove parties to a controlled session or delete it. The compound session could be viewed as a single combined service session, as shown in Figure 4-5, or as separate service sessions and usage service sessions that interact to provide a single service, as shown by the dotted lines. This may affect session object relations in Section 6.3.2.

Controller roles are specialized types of usage party roles. They may be represented as parties in the subsidiary session, while controlled sessions may be represented as resources. In the session graph model, the controlling session may associate its own members with the subsidiary session using composing session relations. For example, a controlling session may add one of its parties to a subsidiary session and indicate this by a share session relation (see Section 6.4.6) associating the party and the controlled session. The controlled session need not support any complementary relation. Actions on a member in one session can affect the related member in the other session.

The control type paradigm can be used to support merging or splitting services into components or sub-services. One session can join another in the control role to achieve dynamic merging via access roles. To split a service, the controller may transfer some control aspects to another session (either an access session or another controlling service session) and end its relation with the controlled session. It may also need to modify the context of the subsidiary session. Special feature sets may be required to support information sharing between component services and to allow them to split.

The relation between the meeting room service and the video distribution service in the example in Section 4.1.1 can be modelled through this composition paradigm.

4.2.3.4 Management Type Composition Paradigms

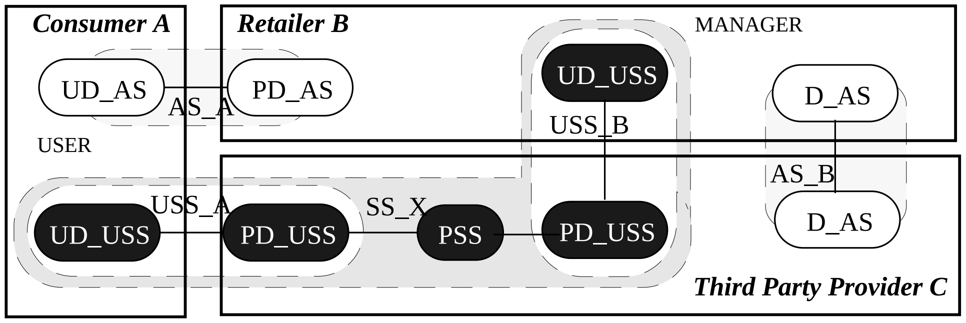


Figure 4-6. Management type paradigms.

Manager type compositions are similar to usage party compositions. Unlike those compositions, the service in the manager role does not initiate a composition. Instead, managers are usually nominated to the role. Services in manager roles usually represent particular management systems, so composition with a particular service instance is needed. Management relations can be classified as follows:

- “monitor” relations where one session collects events from another (either directly or via a notification service) but may not intervene;
- “active” relations where one session can intervene in another to carry out management functions. These management functions could relate to FCAPS or context configuration. Active and monitor relations can be combined by an active manager.

In Figure 4-6, consumer A initiates SS X via AS A with the retailer, which acts on AS B with the service provider. SS X then invites a management system in Retailer B to become a manager of the service. Active manager roles are similar to usage party roles. The managed service would see the manager service as a simple party, as represented in Figure 4-6. The manager may in fact be a service session itself. The manager service would have access to management related information in SS X.

By introducing management roles, composition can be used to support session management. It is possible to use the same manager to manage multiple sessions involved in a compound session. This allows more consistent and uniform management. It is also possible to combine control and management relations. This would enable a controller to be aware of the status of its components and any corrective actions it may need to take. A session may support multiple managers.

4.2.3.5 Usage Peer Type Composition Paradigms

A number of usage peer related composition paradigms are possible. These paradigms are closely related to federation but do not necessarily support all federation requirements, such as distribution of control. We have identified paradigms related to merging, integrating, splitting sessions, where these relate to peer services sharing or transferring members between each other. Merging and splitting components can be handled by control type paradigms, see Section 4.2.3.3. For all paradigms, specific feature sets may be required. The merging of stream bindings requires further study.

Merging refers to two or more sessions forming a cooperative relationship to support different or shared users. *Integration* refers to two or more sessions cooperating to form a single session (i.e. all but one session are dissolved). *Splitting* refers to creating a new session from an existing session. Generally, these paradigms support dynamic changes to existing sessions. In contrast, federation usually requires the instantiation of sessions to support a federation of partners. These paradigms could also be used to “statically” distribute service logic, in which case they may require instantiation.

Direct merge composition paradigms

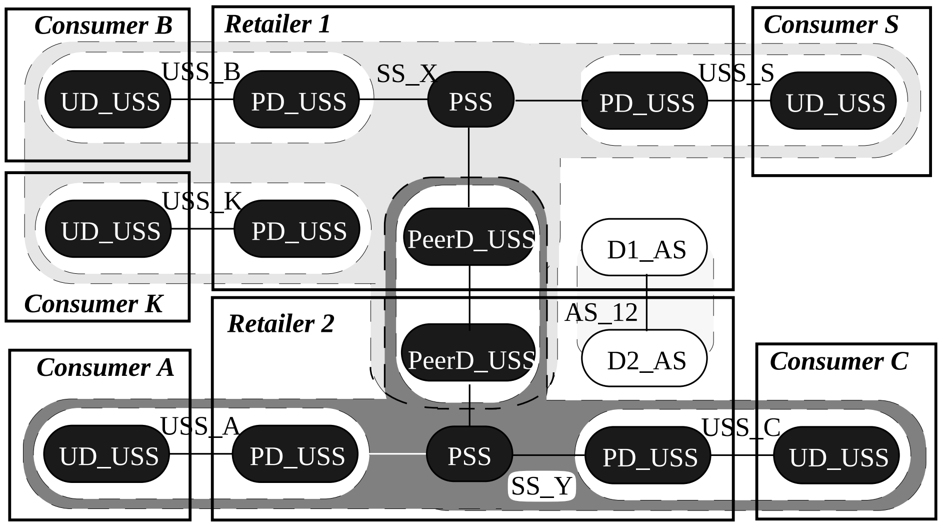


Figure 4-7. Direct Merging Services Type Paradigm.

Direct merging is shown by Figure 4-7. Direct merging involves two services directly forming a relationship with each other. That is, one service session, X, invites another, Y, to merge (i.e. join it). Two types of relationships can be established: tightly coupled or loosely coupled.

Tightly coupled relations. In this type of relation, each service session sees the other as another service session in a peer relation. Sessions need to transfer information to inform each other of their members, though each session may choose the information shared. Each session sees members of the other session, and may treat them as its own members, except that all interactions must be directed via the peer usage session with the other service session. This is supported by composing session relations in the session graph model, see Section 6.4.6.

Loosely coupled relations. In this type of relation, each session sees the other as a member, though a special peer type of member. Each session has a limited view of the other, and members of the other session are not directly visible. This paradigm may be useful for distributing service logic for services supporting very large numbers of members.

Multiple session or domain paradigms

This section considers the complexities when more than two sessions wish to merge. While it would be possible to support such merges by each session establishing a direct merge relation with every other session, other paradigms are possible and may be preferable (e.g. for efficiency reasons).

Chaining: sessions merge along a chain. Each session forms a direct merge relation and interacts with its neighbours on the chain. A session controls how its neighbours view the merged session. Normally, unless at the end of the chain, each session has two neighbours. However, it would be possible for a chain to divide, resulting in a session with more than two neighbours.

Star: the initiator creates a special merging service session which is used to handle relations with the other sessions. All the merged sessions interact via this special merging session. The merging service controls the view each merged service has of the others (e.g. tight or loose coupled view).

Coordinated: one session (the initiator) acts as the contact point coordinating the merger. Its role would be similar to the merging service session described above, except that it would also support members of its own. The other sessions would interact with this session to relate to the others.

Integration type composition paradigms

In this paradigm, a service session forms a relation with one or more other sessions, with a view to creating a single integrated session and dissolving the other sessions. This type of relation is only possible for closely related services, as the integrating services need to support compatible feature sets and session models. Sessions need to negotiate which session survives and exchange information, resources, and possibly components.

Splitting type composition paradigms

Splitting may not appear as a “peer” type relation, but it actually is closely related to integration type paradigms. The initial session creates a new session of the same type and transfers information, resources, and possibly components to it. The sessions may then terminate their relation.

4.2.3.6 Access related composition paradigms

Access type paradigms are used to support access behavior. In TINA, the access session is used to handle incoming invitation or access requests. The access session can be configured with a great deal of the intelligence previously confined to IN related services. While this behaviour could be contained within the access session, composition could be used to extend access functionality.

Invitation support paradigms

Invitation behaviour may be quite complex. The access session may need to interact with services, access sessions and other entities to support its actions. Auxiliary services are required to configure invitation actions. This could be considered as part of access session configuration management. Actions could include the following:

- Pass on the invitation to another access session: to do this, the initial access session may need to use a location or broker service to discover the access session. It then acts in the role of inviter to that access session.

- Invoke a session: the access session can initiate the service session to which the responder is invited. This session is usually initiated in the responder's domain.
- Store invitations: the access session may use a service to store invitations for recall.
- Initiate a session to handle the invitation: the access session may use an ancilary service to process the invitation. Such a service may be specialized to support a particular service (or family of services) or provide negotiation abilities (for instance, the checking of the acceptability of a context).

Access request (start/join) support paradigms

An access session may need to search for a service provider, access session, or service factory related to an access request. To support a request, the access session (or responding domain access session) may use location services. Once an access session or service factory is located, the access session may rephrase the request (if necessary) and pass it on. If the service is provided by another domain, then the initial access session acts as an initiator to the other access session.

4.2.4 Indirect relation type paradigms

When an access session is established, it can be used to control the actions of associated service sessions. This relation can be exploited to give one session indirect access to another. For example, a scheduling service uses an access session to resume or initiate a session. Or, if an access session can interrupt service sessions, the access session could be used to interrupt other services to deliver invitations or other information (such as billing information, disconnection warnings and so on). This may be useful if the connections to a terminal are limited. This behaviour could be controlled by the access session invitation configuration. It depends on the control relations between sessions.

4.3 Basis for federation

4.3.1 Principles

This section presents the principles for federation. They are specific to either domain or service federation or common to both types of federation.

4.3.1.1 Common principles

- **Mutual agreement**: Partners in a domain or service federation coexist and interact on the basis of a mutual agreement upon interaction principles and sharing of resources. This mutual agreement will materialize in a contract.
- **Decentralization**: There is no superior administering object in the establishment of the domain federation nor the subsequent service federation(s). The administration is realized in a distributed way, in the sense that every domain involved is responsible for its own administration. This is opposed to a joint federation where an administering entity would be chosen and responsible for the administration of all domains involved.
- **Autonomy of Partners**: Partners have complete control over the resources in their own domain. A partner shares resources with or performs actions for other partners according to its own decision. It also views and uses resources or results provided by other partners according to its own judgment. An example is that subscription data is not shared. The danger in letting your competitors know such important information about your customers is obvious. For this reason, access sessions between consumers and retailers take place in the home domain (the domain which the user is subscribed to), even if the user is visiting a remote (and federated) domain[16].

4.3.1.2 Domain federation principles

- **Federation autonomy**: Each partner joins or leaves a domain federation at its own decision. No partner can be forced into a federation, nor be forced to remain in a federation. However, the contract establishes the obligations and conditions for entering and leaving the federation through federation policies.
- **Domain federation transparency**: Since federation domains may differ in their entire structure, information belonging to one domain may not be compatible for its use in other domains. Some mechanisms to provide a common basis of understanding (shared knowledge) have to be defined and agreed on during the negotiation of the federation contract.
- **Security**: A domain federation has to cope with the mutual suspicion among the authorities/organizations which own or govern the federated domains. Since federation is a relationship to overcome system boundaries, it has to be ensured that it does not grant more than what is wanted, i.e., access is not given inadvertently to unauthorized objects. For these reasons, mechanisms to secure each domain in a federation are necessary.

4.3.1.3 Service federation principles

- **Service federation transparency**: Once the service federation has been established, the fact that different structures are connected should be hidden from the application viewpoint. This transparency might be affected by the QoS aspects of the federation.
- **Composition principles**. Service federation also needs to support the principles associated with composition. In particular, it must be possible to identify and locate services and end-users associated with various business domains. Identification and location are necessary for both service federation and most domain federations. Section 4.2.1.1 gives general location and identification requirements. Section 7.4.8 gives an example of service federation and end-user location between domains.

These principles are supported by the following concepts and the information and computational models. These models introduce new types of Domain Usage Session Managers to support federation. These could include functionality to face the problem of combining heterogeneous federated systems, acting as adapters that hide technical or organizational differences between domains. The suitability of the federation framework for scalability and wide area distribution is to be considered further, although no major problems have been found in a first analysis.

4.3.2 Supporting Concepts

The following concepts provide support for federation principles at both domain and service levels.

4.3.2.1 Domain Federation Contracts and Support

Domain federations are specified by federation contracts, and may be supported by on line services for negotiating and managing contracts. Note that a domain may be involved in multiple federations. Federation contracts are usually negotiated prior to establishing federated sessions.

4.3.2.1.1. Federation contract and policy

A federation contract describes the goals and terms of the domain federation and the obligations it places on the partners. The federation contract also specifies policies and shared knowledge between domains. Federation policies define how a domain federation is created, how new partners can

16. Note that this is the current view. Results from auxiliary projects might affect this.

join, how partners can leave, how the terms of the contract can be modified, and how the federation is terminated. Federation contracts may be used as the basis of the management context, see Section 5.3.4, negotiated between federated sessions. A federation contract may include the following:

- The list of all retailers participating in the federation with associated information (name, contact, etc.),
- A unique federation identifier,
- The starting date and possibly a duration (or periodicity of the agreement),
- The common goal to achieve: this could be to support invitations, personal mobility, or allow the federation of a particular service or services,
- Modification policy: unanimity or majority (percentage required); the entity responsible for the voting procedure; a participant or independent entity providing a voting service.
- Termination policy: length of required notice or other conditions for either leaving the federation or dissolving the federation,
- Partner obligations, e.g., indications of behavior which invalidates the contract,
- Shared data: information to be shared between partners, e.g. common shared knowledge (Section 4.3.2.3), associated consumers, available services, and shared resources,
- FCAPS management contexts that apply to the federation relationship. Quality of service attributes and charging agreements are, for instance, part of this information.

4.3.2.1.2. Domain Federation Negotiation and Management Services

While the federation contract describes a federation or proposed federation, domain federation negotiation and management services are used to negotiate the setup and modification of such arrangements. This section outlines the capabilities such a service should offer.

Federation establishment. A federation relationship may be established as soon as two or more entities desire to provide a service between their domains. The partners create a contract to describe the goal and terms of the federation. All partners must agree on the contract before the federation is established. If they do not, either the contract is modified, or the dissenting partner(s) does not join.

New partner joins. New partners may join an established federation, with the consent of existing partners. The federation contract may indicate what kind of consent is necessary: e.g. unanimous agreement, majority agreement, etc. The potential members need to be made aware of the terms of federation contract before joining.

Partner leaves. Partners retain the autonomy to leave a federation at any time. The federation contract may impose constraints, such as length of notice.

Modifying the terms of the federation. The terms of a federation may be modified. The partners in the federation must agree before the modifications are adopted. As before, the federation contract may indicate the consent necessary.

Terminating a federation. A federation is terminated when the last partners involved leave or agree to end the federation, when the federation expires (according to the contract), or when the goal is achieved and the federation is no longer needed.

4.3.2.2 Service Federation Support

In general, service federation support is derived from service composition support: it may use concepts such as roles and contexts to define the federation relation. However, a federation is restricted by the principles and requirements of federation. In service federation, these restrictions are expressed in part by the contexts exchanged between domains on a request to join or establish a federation. These contexts are derived from the federation contract agreed on between the domains. They are also supported by the shared knowledge exchanged between the sessions, see Section 4.3.2.3.

4.3.2.2.1. Quality of service of federation

When using the service, a user should not notice whether the service is federated or not. There is, however, the problem that two different retailers involved in a federation might be offering services with different values of QoS. It is very likely that the QoS of the federated service will be limited to the lowest of the involved services QoSs, this yields a federated service with a QoS different from the non-federated one.

4.3.2.3 Common Shared Knowledge

The Common Shared Knowledge (CSK) forms a common basis of understanding among all partners of a federation. This allows any partner to establish several peer-to-peer relationships with several partners while using the same common basis of understanding. It is required for federations at both domain and service levels.

4.3.2.3.1. Domain Common Shared Knowledge

Domain CSK is the level of visibility , in relation to its domain and its business, that a partner wants to provide to other partners of the federation relationship. The domain CSK takes into account security problems and the mutual suspicion that arises between different partners involved.. For these reasons, security aspects represent an important part of this CSK. The CSK is a global view common to all partners.

For each federation relationship there is one CSK. It may be argued that several CSKs can be defined between pairs of federation partners. However, this is understood as having different federation relationships. An example of a CSK is the international numbering plan, based on Recommendation E.164, where all administrations agree to use certain prefixes for each different country.

4.3.2.3.2. Service Common Shared Knowledge

Service CSK is the combined information exported from each service session in the federation to its partners. Each partner sees its local information and the information imported from its partners; hence each partner sees the CSK. The CSK may contain generic and service specific information. CSK may include parties, peers, and resources of the participating session. Exported session members remain under the control of the exporting session, but other sessions (or sessions' members) may interact with them. It is also possible that there may be resources “common” to the federation , i.e., all participating sessions have ownership rights.

In principle, all sessions see the same CSK. However, the paradigm used to support the federation and policies between particular sessions may allow, or result in, slight differences. These may be minor differences , reflecting perceived differences in relations among the sessions, or they could reflect different policies in sharing information. (e.g., In a chain type paradigm, Section 4.2.3.5, a session determines what CSK it will share with its neighbors.) Federation contexts (a specialization of management contexts, see Section 5.3.4) should guard against arbitrary differences.

Partners in a federation need to agree on the type and format of the CSK. The TINA service model, the service session graph, supports the CSK concepts, see Section 6.4.6. Since CSK need not be a session graph, the CSK must be agreed on during the access phase.

4.3.3 Federation Paradigms

We have previously outlined a set of high level principles for the capabilities that must be supported in the federation framework. Below we try to couple these principles more tightly to SA definitions to illustrate domain and service federation. In particular, we will consider the access and service federation paradigms, and how they relate to access and service session concepts.

4.3.3.1 Federation Access Paradigms

To establish a domain federation, the participating domains need to establish access sessions between themselves. These access sessions are associated with federation contracts[17] which set the terms of federation for the domains (access) and particular services. The *Terms of Management* (see Section 5.3.4.1) for the federation express these contracts for particular services and sessions.

To support domain federation, an access session must support both symmetric behavior and be associated with one or more federation contract. Symmetric access sessions support the same interactions in both directions across the domain boundaries. The *peer domain access session*, described in Section 3.5.2, supports such a behaviour. Such a session can be termed a *federation access session*. It should provide access to federation contract management services.

A domain federation may support many types of services between domains. These may include invitation handling, federated location services, personal mobility, general service composition, and service federation. To do this, they may need to support the behaviour outlined in Section 4.2.3.6.

Though one of the partners initiates the access session, this does not affect subsequent behaviour. Behaviour may be constrained by the terms of the federation contract, but not by the access session initiation role. Either partner may initiate the access session. Once the access session is established, either partner may establish a federated service session.

17. An initial access session is required for negotiating the federation contract.

4.3.3.2 Service Federation Paradigms

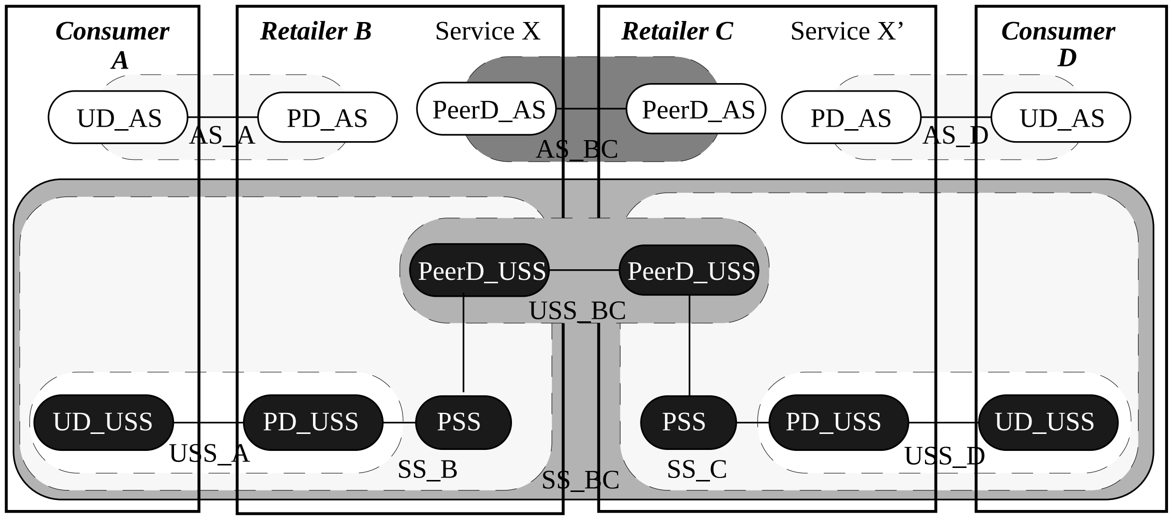


Figure 4-8. Sessions involved in a federation of service sessions.

Service federation paradigms rely on the federation access paradigms outlined above. They are also closely related to peer type compositions, see Section 4.2.3.5. Federated service sessions support symmetric, peer-to-peer behaviour like federated access sessions. Figure 4-8 shows a service federation example. Consumer A participates in SS B, and invites consumer D to join. Consumer D is associated with another retailer domain , and to join, SS C must be started and a federation relation between SS B and SS C established by a *federation service session*. Section 7.4.8 details this scenario from the computational viewpoint.

Once the *federation service session* is established, each of the sessions is able to act on the other's session graph, or on a shared part of it. Each session may define this part (the exported or visible one) following its own policies or the agreed context. This is further explained in Section 4.3.3.3.

As before, session behaviour is only constrained by the terms of the agreed federation context, but never by the role played in the establishment of the federated service session (requester or responder). Both partners play a peer role in the session and are supported by a *peer domain usage service session* in each peer domain. Each PeerD_USS component supports and requires the same interfaces. The specific interfaces depend on the feature sets selected.

In general, each partner in the federation must support a usage peer role. This implies that the same set of capabilities is offered on both sides on the relationship. Federation components, supporting usage peer roles, may be composed by integrating interfaces corresponding to usage party and usage core provider roles.

4.3.3.3 Representation of Service CSK

When several sessions participate in a service session, each one usually needs to keep the control of its own session members while providing a view (maybe partial) of them to the other sessions. This set of shared views is an important part of the Service CSK. The Service Session Graph (SSG)[18] provides the required support for the representation of this part of the service CSK. Other information could be included in the Service CSK, e.g. management information specified by management con-

18. Refer to Section 6.4 for details on the session graph and how it supports CSK. A prior reading may help the understanding of this section.

texts. The use of SSG is not prescribe; other representations could be negotiated during the access phase and defined in the federation contracts. This section will discuss service CSK in relation to the SSG.

A SSG represents a session's own and imported[19] members, and session relationships in a session. It also indicates which members are exported (made known to other partners).

Each session participating in the federation represents the other sessions in the federation as peer session members. The members imported from that session are associated with the peer through a composing session relationship. This relationship indicates that the manipulation of the member and the notification of changes on it are the responsibility of the session represented by the peer. Similarly, members exported to other sessions are indicated by another type of composing session relationship, between the exported session member and the peer representing the importing session.

Each service session works on a SSG composed of own members and imported members. It is possible that no retailer has a view of all the members participating in the session, as every session can hide some members. So, each session in the federation may have a different SSG. Regardless, all the views are kept consistent and their intersection is considered the service CSK. The SSG defines a set of constraints and dependency rules to ensure consistency (see Section 6.4.6.1).

19. An imported session member is a member that is owned/controlled by another session. For a more formal definition of imported/exported session members refer to Section 6.4.6

5 Service Management

This chapter explains the management part of the TINA service architecture. The objectives are to introduce the principles, requirements, and supporting concepts, but not detailed definitions of information models and computational models[1]. This chapter deals mostly with management issues within the service architecture. Pointers to other TINA baselines exist, e. g., the network resource architecture, and will help the reader's understanding. Since one of the major focuses of the current version of the service architecture is federation and composition, this chapter will provide a few examples to illustrate its support for management issues.

5.1 Scope of Service Management in TINA

Service management is a very broad discipline, and is also an important part of TINA service architecture. In its broadest definition, service management may include most of the service supporting functionalities of TINA service architecture, such as subscription, user account management, etc. Because of its breadth and versatility, it is often difficult to make a clear distinction between service management and service, as is evident in terms such as "billing service", "on-line subscription service" etc. Another potential cause of confusion lies in the dynamic nature of TINA service itself, since some service management functionality (notably accounting) often needs to be built-in to the service itself to support flexibility, when it is instantiated.

5.1.1 Four Axes

There are four conceptual axes associated with service management as listed below. These axes encompass the management issues that need to be considered by TINA.

1. Partitioning axis: TINA is partitioned into three layers: service, resources and DPE. The management architecture is likewise partitioned, concentrating on service and resource management. It also supports the concept of domains and management of and between the domains.
2. Functional axis is represented most notably by FCAPS functions. To support the FCAPS integrity of a service session, constructs such as management context and service transaction are provided.
3. Computational axis represents computational support for management needs. These computational supports are mostly offered by DPE, but we will consider some TINA refinements to event management and grouping concepts.
4. Life cycle axis represents the life cycle issues, including service life cycle (SLC) management and user life cycle management, i.e. the life cycle management of consumers.

Of the above four axes, service life cycle management and user life cycle management are mostly independent from the rest; they are often seen as separate subjects. In this document, however they are treated as parts of broader service management issues. Axes 1 and 2 are mutually related, since the user's service requirements trigger FCAPS activities in the service, which mostly occur in the utilization phase of the service's life cycle. On the other hand, service life cycle management may utilize configuration (C) of FCAPS to deploy or withdraw some part of its service.

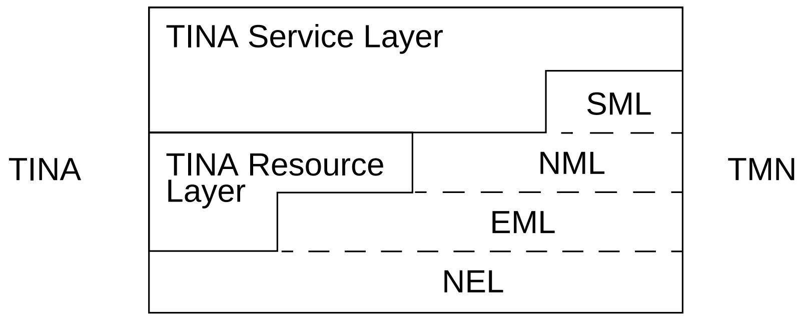
1. The current version of information and computational models are found in section A-3 in the annex of this service architecture.

5.1.2 Functional Scope

Functional scope considers the capabilities that service management should encompass. TINA encompasses the FCAPS management areas, which is the well-known acronym of OSI functional classification. Although FCAPS requirements in TINA are still research subjects at large, they can be summarized as follows;

- **Fault Management**: Fault management deals with fault alarms in general, which includes alarm correlation and alarm distribution. It also includes identification of faults and the types of faults. It is particularly important for a TINA system to be reliable, resilient, and fault-tolerant, since it consists of many semi-autonomous distributed elements. This distributed nature makes it very difficult to control and manage reliability of the entire system. For the resource level fault management, please see NRA [8].
- **Configuration Management**: Configuration management deals with configuration of resources such as network resources, computational resources, software resources, etc., so that the configured resources become available for TINA services. Since the types of resources a TINA system deals with are very broad, TINA configuration management, in fact, is a collection of separate, but mutually-related configuration management schemes for different resource types. Configuration of TINA services is treated in the service life cycle management, Section 5.4.3. Configuration is related to fault recovery, where systems need to be reconfigured using the results of fault diagnostics.
- **Accounting Management**: Accounting management deals with issues such as billing, accountable objects, and delivery of accounting events. It is important in TINA that flexible billing options, such as on-line billing and third-party billing, are available and universally applicable to any TINA service. TINA accounting needs to be distributed as well, for ex ample, to enable it to deliver consistent accounting results of multi-party sessions across multi-provider domains.
- **Performance Management**: Performance management deals with monitoring, control and administration of performance in a TINA service in a broad sense.
- **Security Management**: Security management deals with security issues at the service level, and subsequently with those at the resource level. TINA security management has to address issues such as authentication and authorization in the multi-provider domain environment. To establish an end-to-end trusted service session in the multi-provider domain, the chain of trust between domains needs to be maintained and managed properly.

5.1.3 Layering



SML: Service Management Layer
NML: Network Management Layer
EML: Network Element Management Layer
NEL: Network Element Layer

Figure 5-1. TINA Service Management Principle

Layering corresponds to the traditional distinction between resource management and service management. Resources are network capabilities and equipment. Resource management relates to the management of network resources. We define service management as all the management activities within the service layer. Thus, service management is an inherent part of the service layer. This section will concentrate on the service layer management issues.

In comparison with TMN, which so far focuses mainly on the issues at NML and below, TINA takes a service oriented approach, starting with service and business related concerns with the aim of achieving an information and communication service architecture over a DPE. This provides a natural basis for considering an integrated approach to services and management. In comparison, TMN takes a resource-oriented, bottom up approach, where service and business requirements are considered last. TMN service management corresponds to a fixed set of functionality on top of the network management layer. Figure 5-1 compares the TINA and TMN [32] management models.

5.1.4 Computational Aspects

TINA has adopted the ODP viewpoints, including the information and computational viewpoints. Traditionally, management has focused on the information viewpoint, which considers the semantics of a system. But management has also included information processing activities, which in ODP belongs to the computational viewpoint. The computation viewpoint considers the decomposition of a system into a set of interacting objects that are candidates for distribution. As TINA is based on a DPE architecture, the computation aspect is very important. It provides the basis for the structure of managed and managing systems. The DPE infrastructure provides basic services upon which a more complex management framework can be based.

As managers, managed objects, or agents representing managed objects, computational objects will naturally tend to use DPE services rather than more traditional management protocols, such as SNMP and CMIP. However, interworking between DPE based systems and existing management systems is important. X-Open[49] and NMF [48] are working on GDMO to IDL translations. These form the basis of interworking between DPE based and OSI systems and for the reuse of current management semantics.

Typically a DPE offers a wide range of support services to aid system management. These include various life cycle management services that support object creation and deletion; security services, which could include encryption, authentication, and authorization; and various basic facilities such as event management and logging.

In general, TINA has similar requirements to the OMG when it comes to considering DPE services and generally has adopted OMG services as the basis of its management services[2]. In some cases, TINA can build on basic DPE services for particular management service requirements. For instance, to locate managed objects in a distributed system, a locator service is required. A locator service could be based on the general CORBA trading service.

5.2 Requirements

Although the management axes present unique requirements of its own, there are some principles common to all TINA service management axes:

- **Support for Multiple Business Domains**: TINA service management must interoperate across multiple business domains. Moreover, it has to offer a consistent and guaranteed service management quality (typically QoS[3]) across the multiple domains.
- **Flexibility**: TINA is essentially an open market. It is expected that the market and price structure change dynamically, reflecting today's competitive market. Thus, service management must be flexible to satisfy those needs. TINA service components can be customized following FCAPS service management needs, upon their creation at a service factory.
- **Controllability**: although TINA is essentially an open distributed system, the provider needs to exercise control over service instances in its domain. The operator often needs to monitor or control the status of service instances, to maintain or to manage their quality.
- **Federation and Composition:** It is necessary to support the management of composed sessions and federated domains. In a sense, this is also a consequence of multiple business domains which TINA supports. For example, given QoS requirements for an overall service, its associated management responsibility must be shared by the federated parties, and the QoS must be supported by the composed service.

5.3 Information Support

The TINA management architecture uses management context and service transactions to support FCAPS management. In this section, we introduce supporting information objects, in particular:

- **Management Context** (MgmtCtxt) is a specification of management requirements between domains in relation to some functional management area.
- **Terms of Management** (ToM) is an aggregation of binding MgmtCtxt instances, when the service session is instanced.

2. Primarily, it meant event service and notification service. It is expected, however, that more COS (common object services) useful for telecom management services continue to appear via the initiatives from OMG TelSIG.

3. There is no definition for QoS so far.

5.3.1 Domains

TINA does not envision a single service provider responsible for providing all TINA services. Instead, it assumes that there will be a number of stakeholders acting in various business roles. Stakeholders are real world commercial entities that provide services or communication resources. Business roles arise from analysis of a business model (see Section 2 and [3]). The NMF has performed similar analysis [47].

To deal with this multiple stakeholder environment, TINA needs to recognize domains. These may be administrative domains that are controlled by various stakeholders. Within an administrative domain, various management domains may exist to help organize the necessary management activities. On the other hand, by its nature (aggregation of objects), a management domain could span more than one administrative domain, although this is not a usual case.

5.3.2 Management Domains

A management domain can be modelled by an information object associated with a certain management functionality, such as accounting, security, or DPE management. Domains are similar to object groups in that they both represent a set of objects.

The followings are characteristics of the management domain:

- The domain boundaries are usually based on natural affinities between objects, such as network topology, business stakeholder, or geographical area.
- A domain may be decomposed into a number of subdomains, allowing a hierarchical composition.
- Domains are by no means disjoint. Not only may an object belong to two or more different management domains, but these domains may also relate to each other in different ways.
- The behavior of the managed and manager objects in a domain is governed by the management policy that is applied in that domain.

The generic management domain can be specialized with a set of management functions (policies) to become a specific management domain, e.g., an accounting management domain. Each management domain may have its own information model. The information model will usually consist of policy, resource, and manager objects. For example, an accounting management domain would consist of an accounting policy, a metering manager responsible for collection and recording of accounting information, and various accounting resources, see Figure 5-2 This information model maps to a computational model, with corresponding computational objects.

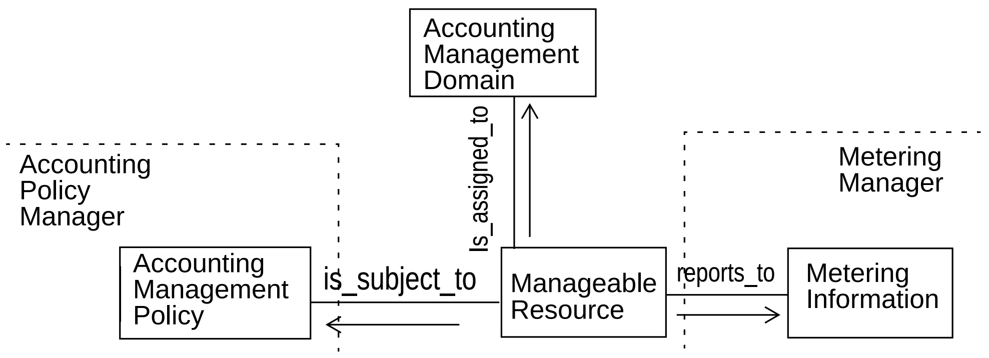


Figure 5-2. Accounting management domain example

5.3.3 Management Policy

A management policy is a set of rules governing a particular management function in the domain that is associated with the policy. For example, an accounting management policy may prohibit accounting events from being transported across the domain boundary, for some security or enterprise considerations. This rule effectively forbids on-line billing.

A management policy is described by an information object, and implemented by manager and managed objects, as shown by the accounting domain example in Figure 5-2. A management policy can address many issues, including:

- **Managers**: Management policy determines the number and types of managers deployed within the domain and which resources they manage. Managers are responsible for ensuring management activities are in accordance with management policy.
- **Event Management Policy**: This dictates acceptable event management options, such as translation, forwarding and duplication. Policy may limit options, even though they are physically supported, for security or performance reasons. Also, event management options may limit federation or inter-operability between domains.
- **Security Features**: Since management actions and events may govern resource access and billing, they are natural targets for consideration. As TINA service and resource layers are built on open DPEs, security features are mandatory. For instance, non-repudiation security options may be required for accounting events.
- **Policy Applicability Rules**: The policy is a set of rules. Rule applicability may vary depending on how the management domain is operated. Each rule is associated with one of three different applicability values: *mandatory (hard)*, *optional (soft)*, or *negotiable*. These applicability values are also used to resolve conflicts when management domains overlap.

Management policy governs a management domain. By contrast, a service transaction and its associated management context set management requirements for the delivery of an instance of a service. The management context only exists for the duration of the service transaction. Objects in a management context are also part of a management domain and are in the scope of its management policy. For a service transaction to proceed, the management context and the management policy must be compatible.

5.3.4 Management Context

A **management context**[4] represents a set of functionality requirements for a session (or type of session) with respect to a specific management functional area. It defines the rules that governs that particular management functional area during a session. The management functional area a management context is considering may be one of the classical FCAPS areas or associated with DPE or lifecycle management issues.

A management context is usually agreed on by stakeholders in different administrative domains participating in that session. One of the stakeholders may play the usage provider role, and the others may play the usage party role,. But the management context concept is also applicable to other types of relationships between administrative domains, such as composition and federation, in which several providers cooperate in the session acting in a usage peer role[5].

4. There needs to be more elaborate analysis of management contexts and management policies based on realistic examples, but so far it was observed (in the accounting management) that the level of interaction between the two can be made relatively small, though it depends how contexts and policies are designed.

5. See Section 3.4 for more details on role definition, namely usage party, usage provider and usage peer roles.

The details of the management context depend on the individual FCAPS management objective. For example, the accounting MgmtCtxt can consist of the following information:

- **Event Management Configuration**, which specifies delivery timing and the event channel to be used for accounting event delivery.
- **Tariff Structure**, is essentially a function[6]. Provider calculates charge/charging-rate from accounting event sequence.
- **Billing Configuration**, which contains types of billing options, such as on-line or shared billing.
- **Recovery Configuration**, which specifies recovery actions to be taken, when the service transaction is aborted for some reason, such as network failure.

Each individual FCAPS MgmtCtxt is envisioned as a sub-class of the top-level, MgmtCtxt class. Although the top-level MgmtCtxt class is expected to be the super-class of all the variants of FCAPS MgmtCtxts, it still awaits further studies in each management area to specify more details of the MgmtCtxt information model itself, since it has been observed that the information contents of the MgmtCtxt (e. g. accounting MgmtCtxt) tend to be quite specific to the management area it handles. In other words, what is most likely to be common properties across all types of management contexts are not specific to any management function. Those commonalities are its relationship to its associated service session (binding), and the context management aspects in view of the context configuration/management service.

Although most of the information in a management context is defined explicitly, part of it can be stated implicitly as a part of the user profile or as a part of the subscription contract, which may initialize certain management conditions in the associated service session.

Other contexts are used in TINA service architecture, like usage context, subscription context, etc. The term context is considered a generalization of all these types of contexts.

5.3.4.1 Terms of Management (ToM)

A ToM is an aggregation of all the management context instances that are bound to a session. It is the part of the agreement, related to service management, reached by two stakeholders for the session execution. It defines the behaviour of all the management functions that will govern the session. This agreement may include other service aspects not directly relevant for service management. The details of ToM are yet to be determined, but ToM is meant to comprise all the management agreements, expressed by means of management contexts, which are bound to the given service session[7].

5.4 Management Concepts

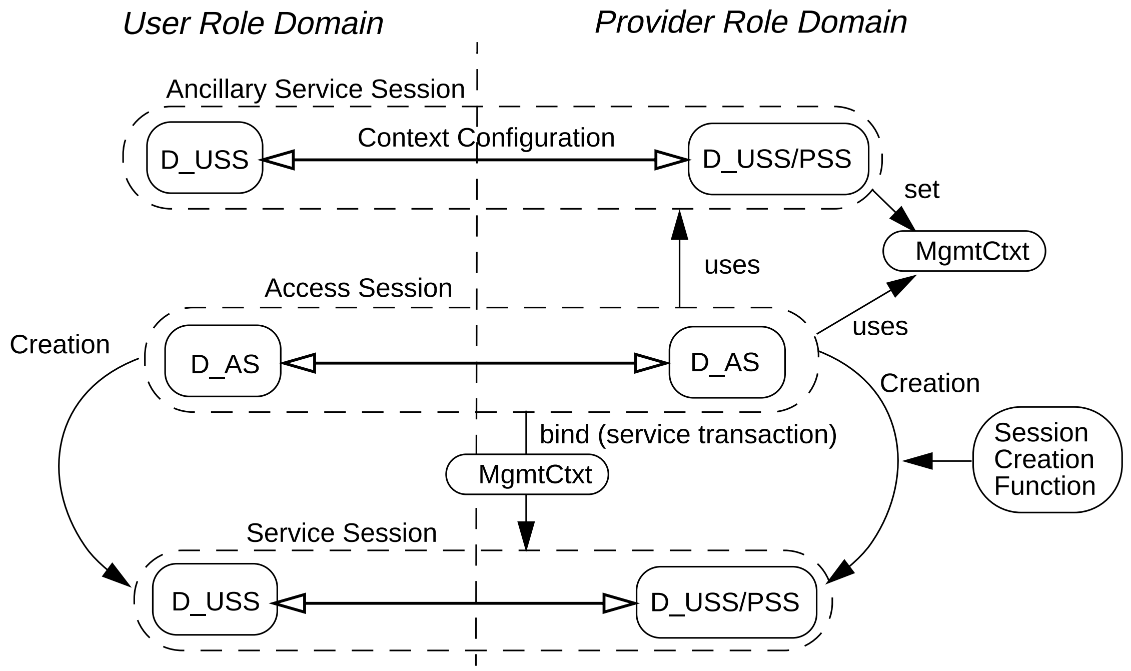
These concepts discuss how the management context and terms of management are applied in the service architecture to support service management. We will consider:

- **Management Context Negotiation** allows domains (or entities within a domain) to set management context for access and service sessions.

6. Tariff structure itself is not essential for the accounting management context. Optionally, the context may specify the tariff structure. It may consist of some pointer to tables or to functions, that provides functionality for the billing/charging calculation.
7. ToM is just a conceptual aggregation, whose only objective is easing the reference to the set of management agreements bound to a service session. Each of its components, MgmtCtxts, are supposed to be interpreted and handled separately, since they represent specific management areas. No specific aggregation rules are applied.

- **Service Transaction** is a construct to guarantee the integrity of the service session with respect to its FCAPS management. There is some similarity to the database transaction concept, though the purpose of service transactions is to manage the integrity of the service session, not the integrity of data.
- **Life Cycle Support** is one axis of management from the viewpoint of the life cycle of TINA service components and objects. Service life cycle and user life cycle are considered.

5.4.1 Management Context Negotiation



D_USS: Domain Usage Service Session
PSS: Provider Service Session
D_AS: Domain Access Session
D_USS/PSS: D_USS and PSS

Figure 5-3. Role of Management Context in Service Architecture

Management context negotiation allows two domains to set the management context for a particular session or type of session. Context negotiation is supported by ancillary services. A context may be prenegotiated and selected for use later, or negotiated at access. Also, it may be possible to renegotiate the context during a session (but this is not mandatory).

When a context is being negotiated, it must be consistent with the management policy of the domains and the terms of the contract between the stakeholders. If the management context is being negotiated in relation to an existing multiparty or composed service session, then it must also be consistent with the management context for existing session members.

Figure 5-3 illustrates the role of management context in the TINA service architecture. Whenever a service session is joined or started, a management context must be bound to it. This context may be defined in several ways. The usage service session in the user domain may request a preset management context, which the user has negotiated prior to the service session instantiation,. Alternatively, the usage service session in the provider domain may use a default management context, which the service provider may provide for that service as a default, in case there is no negotiation or preset one. Another option is that the user domain requests to negotiate or configure a management context during the service instantiation phase. The resulting context is then bound to a service session, which corresponds to the service transaction explained in the following section.[8]

From the computational viewpoint, management contexts usually consist of a set of attributes and often a set of methods or interface references. Management context attributes have the following structural elements:

- a name, that identifies the type of attribute,
- a type, that indicates its data type, and
- a value (or set of values), of that data type.

The details of the attribute-value pairs, however, are expected to be dependent on the FCAPS management function. In particular, the details of Fault, Configuration, and Performance management contexts are still largely an open area for further study at this point.

5.4.2 Service Transaction

A service-transaction is a construct which associates a number of selected management contexts with a service session and guarantees that the associated service session satisfies management requirements which are prescribed by the management contexts. The service transaction can distribute the management contexts between domains, so that a consistent level of service management throughout multiple administrative domains is achieved. This concept is applicable to multiple party services and federation and composition, where a TINA service may consist of multiple service sessions and multiple providers, but is required to support certain QoS and management requirements as a whole.

Figure 5-4 illustrates, through an example, the relationships between service session and service transaction.

The service session has two participants, represented by the two respective usage service sessions (each represented by a UD_USS in user domain and a PD_USS in provider domain). Each service transaction is associated with a management context, MgmtCtxt 1 and MgmtCtxt 2, respectively. When service transaction 1 executes, MgmtCtxt 1 is interpreted and translated into resource level operations. The provider service session (PSS) activates the communication session (CS), which activates metering activity for the communication service session. The usage service session in provider domain, corresponding to the service transaction, passes its notification interface to the metering manager, and the metering manager reports notifications to the usage service session as required. Requirements for service transaction 2 are met similarly.

The service transaction represents a view of the service session local to the concerned user-provider relationship. The ToM, in particular, represents the agreement between two stakeholders participating in the service session. The service transaction is a virtual framework, which guarantees the completeness of the management requirements expressed in the ToM.

Execution of a service-transaction consists of three phases. Here is an example, using an accounting management context:

8. More details regarding context negotiation and set-up are provided in section A-3 in annex of this document.

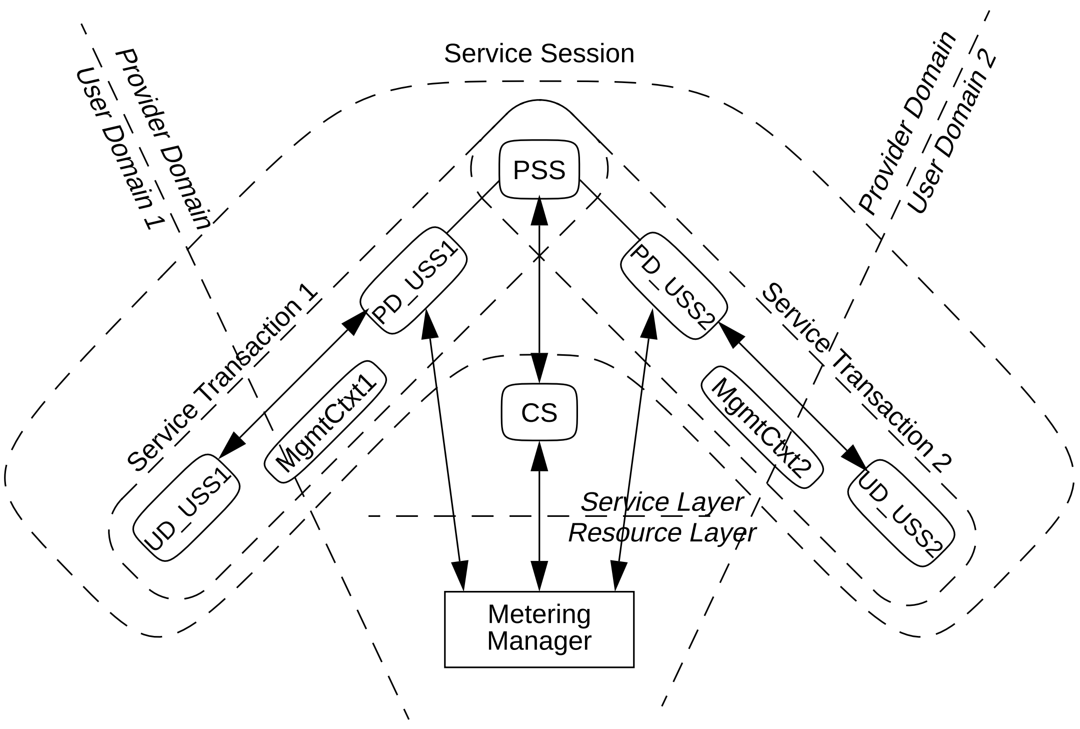


Figure 5-4. Relationships between Service Session and Service Transaction

- **Set-up**: The respective management contexts are interpreted by the provider service session and the associated usage service session and necessary resources, such as the accounting-log record and event-channels, are reserved or assigned.
- **Execution**: The usage service session in the user domain starts running. Following the accounting/billing specified in the accounting management context, accounting events may be logged or reported. If the service guarantees QoS, notifications from the performance management may be reported during this execution phase. If the QoS does not meet the guaranteed quality, it may lead to an early termination of the service transaction.
- **Wrap-up**: The usage service session stops running. Reports from performance management or fault management can be summarized, and are then checked to see if the reported QoS has satisfied the guaranteed level. If it has, the service transaction concludes successfully. If it hasn't, some recovery actions, such as billing compensation, may be considered, and actions may be taken by the usage service session and the provider service session.

5.4.3 Life Cycle Support

This section deals with the life cycle aspect of the management architecture. Life cycle is a complex issue, since all TINA service elements (sessions, services, service components, computational objects, etc.) may have different, as yet mutually related, life cycles. For example, a particular service component may be used for different services, and a new version of service may require one version of the component, whereas its old version still needs an older version of components. TINA architecture needs to provide support to make these life cycle management issues manageable. The life cycle issues bear proximity to the configuration management, at least in the deployment and withdrawal phases. And, in fact, they should be related as an integrated part of the life cycle management system. Unfortunately, the configuration management part in the service architecture is still mostly a research subject at this point. In the following part of this section, we explain the three basic concepts of the life cycle management:

- **Service Life Cycle** handles life cycles of services types, service components participating in the services, and service instances.
- **User Life Cycle** is about the life cycle management of users. The subscription management is a part of it.
- **List of Life Cycle Types** is the list of life cycle types, which TINA service management is supporting.

5.4.3.1 Service Life Cycle

Figure 5-5. shows the overview of the service life cycle management.

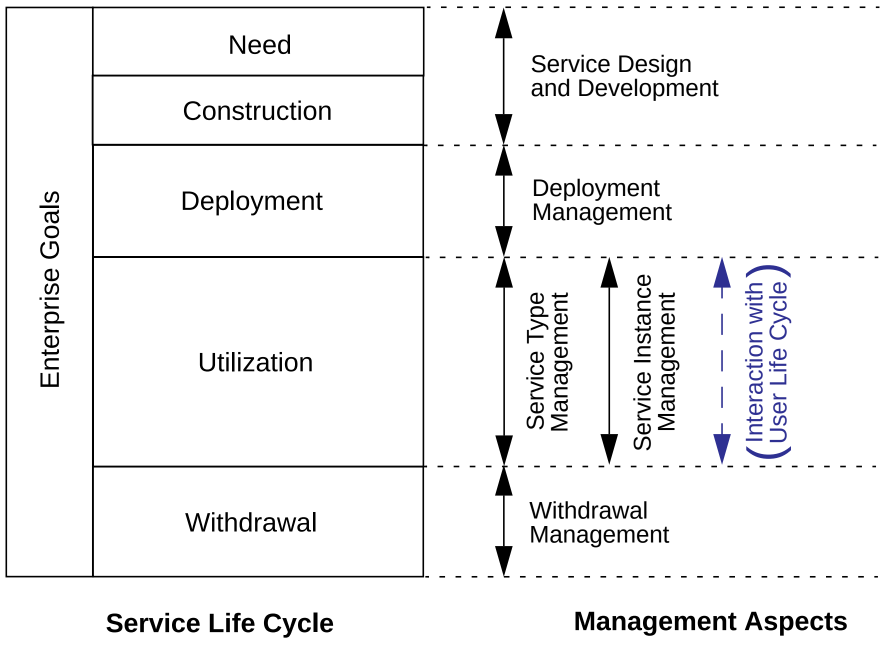


Figure 5-5. Service Life Cycle Management Overview

In the following, we proceed to explain each stage of the life cycle.

1. **Need Stage and Construction Stage:** These two stages include all the off-line activities required in analyzing requirements, designing and developing the software and any special hardware associated with a service. These stage can be further broken down into sub-stages that resemble those of a traditional software development life-cycle. Issues to be considered in these stages are:
   - static service composition (Designing of service),
   - unit of service component,
   - unit of deployment,
   - dynamic service composition,
   - relation to the existing service, including version management, etc.

2. **Deployment Stage:** This stage is responsible for introducing a pre-defined service component, including its management capability, into the TINA environment. This activity encompasses the planning and installation and testing of its constituent parts, and concludes with the activation of the service component. Thus, service interactions with the other services neither impair the service being deployed nor the services already deployed.
3. **Utilization Stage:** All management aspects for service except deployment and withdrawal, such as FCAPS, context negotiation, and service transaction are applied in this stage. The utilization stage of a service is broken down into service provider activities, customer organization activities, and end-user activities, while the other stages are closed in customer organization activities. Therefore, interaction to the user is considered, and these activities may occur in parallel. Taking into account the roles, the provider can control and operate the service (Provider Control & Operation) through this utilization stage.
4. **Withdrawal Stage:** This stage is responsible for removing a service component, including its management capability, from an environment without negatively impacting other live and dormant services. This activity encompasses the planning, de-activation, de-installation and/or de-commissioning of its constituent parts, and the testing for adverse consequences. At the conclusion of the activity, the service component is no longer available to the service provider.

5.4.3.2 User Life Cycle

Figure 5-6. shows the overview of the user life cycle management.

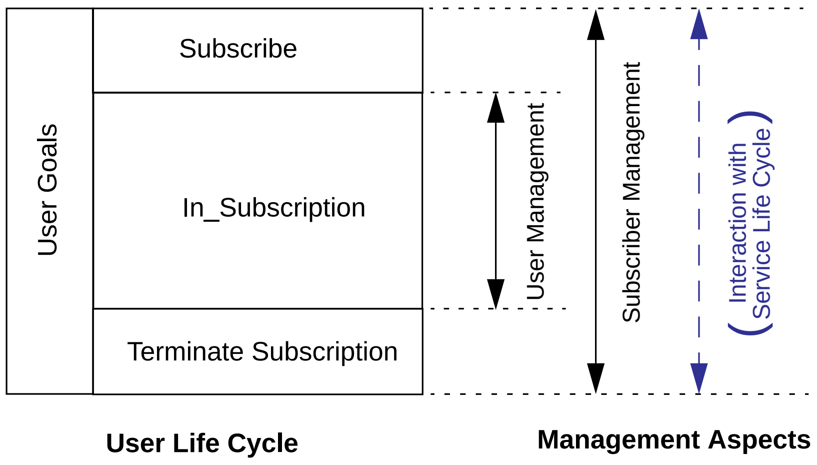


Figure 5-6. User Life Cycle Management Overview

1. **Subscribe Stage:** In the subscribe stage, the user and the provider make a contractual agreement, upon which the SP commits to provide the service to the subscriber in the terms specified in the subscription contract.
2. **In_Subscription Stage:** As described in the utilization stage in the service life cycle, all the management as the subscriber and end user points of view is considered and these activities may occur in parallel. Taking into account the roles, subscriber and end user can control and operate the service during this usage stage.
3. **Terminate Subscription Stage**: In the terminate subscription stage, the user ends its status as a subscriber in the subscribed service, and the subscription is terminated.

5.4.4 Management Aspects of Life Cycles

We introduce the management aspects of life cycles which are considered in the TINA service architecture. We also expect that the coverage of scope derived from life cycle aspects may broadened in the future.

The following list relates the entities that participate in TINA life cycle management:

- Users. There are three types of users; these roles correspond to the type of services accessible to the role (see Section 3.4). For example, a subscriber and an end-user may have the same user identity, whereas he/she is called a subscriber when he/she is using a subscription service.
  - **End-User**
  - **Anonymous-User**
  - **Subscriber**
- Service Provider (including Retailer/Broker).
- Service Designer / Developer.
- Deployer / Withdrawer.
- Service Manager: is specialized as follows.
  - **Service Type Manager**: is responsible for the service type life cycle.
  - **Service Instance Manager**: is responsible for the service instance life cycle (namely, service session).
  - **Subscription Manager**: is responsible for the subscription life cycle.
  - **User Manager**: is responsible for the user life cycle and their access (access session).

Table 5-1 shows management aspects and summarizes managing and managed entities within the life cycle as well as roles associated with them. The column Scope describes what should be considered and be provided as our goal in those management aspects. For example, the column for the Service Design and Development row explains that these management aspects consider requirements on informational aspects, and not on a computational aspects.

Table 5-1. Management Aspects in Life Cycles

Management Aspects	Managing Entity	Managed Entity	Scope
Service Design and Development	Service Designer/ Developer	Service (software and resources)	Information requirements for Designer and Developer
Deployment Management	Deployer/Withdrawer in Service Provider	Service (software and resources)	Information and computational aspects for Provider
Service Type Management	Service Type Manager	Service Type and Service	Information and computational aspects for Provider
	Other Provider (Service Provider)	Service Type and Service	Management Service with Information and computational aspects (Federation)
Service Instance Management:	Service Instance Manager	Service Session Instance	Information and computational aspects for Provider
	User	Service Session Instance	Management Service with Information and computational aspects (Customer Network Management: CNM)
	Other Provider	Service Session Instance	Management Service with Information and computational aspects (Federation)
Withdrawal Management	Deployer/Withdrawer	Service (software and resources)	Information and computational aspects for Provider
Subscriber Management	Subscription Manager	Subscription	Information and computational aspects for Provider
	Subscriber (User)	Subscription	Management Service with Information and computational aspects (CNM)
User Management	User Manager	User	Information and computational aspects for Provider

6 Information Modeling Overview

The notation used for the information model in this chapter is the OMT object model notation[51].The abstract classes (for which there is no object instance) are shown in ‘curly bracket’ style[1]: they have no other purpose but to simplify the object oriented modeling.

The relationship to the computational model is presented in Section 7.3.

6.1 Service Information Model

6.1.1 Sessions, Services and Domains

This section presents the information model as related to session related services[2]. As a session related service may extend over multiple domains, which are treated as business administrative domains, it is useful to model the service as an aggregation of one or more domain services, where each domain service represents a part of a service confined to a single domain. A session is an instance of a service and is defined in Section 3.5. Just as a session may represent an instance of a service, a domain session represents an instance of a domain service. Domain sessions may interact to establish services extending over multiple domains.

Figure 6-1 shows the relationship between domain, service and session objects. A session related service is an aggregation of other session related services and domain services. Each domain service is instantiated by one domain session, which is established in and managed by its associated domain. A domain session may be bound to another domain session via a domain session binding, which represents the dynamic information used to link the two sessions.

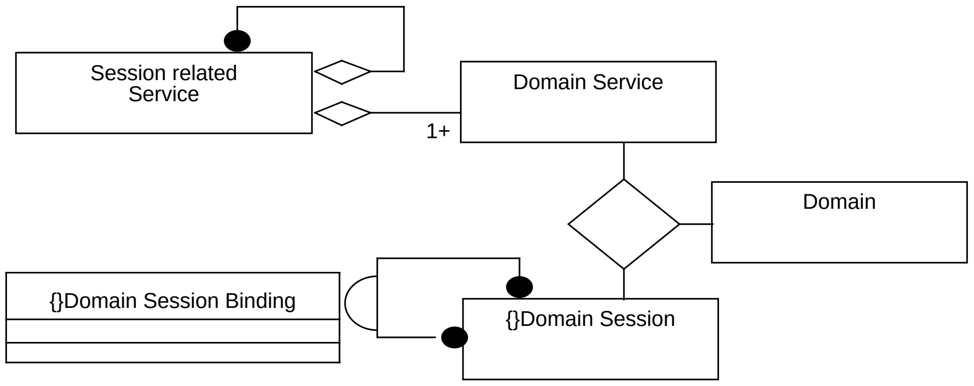


Figure 6-1. Session Information Model

1. I.e., ‘{}’ is inserted in front of the class name in the diagrams.
2. A TINA service is classified into two types of services: one is ‘session related services’, and the other is ‘non session related services’. ‘Non session related services are out of the scope of the service architecture. The session related services are defined as follows:
   - Session related service: represents an on-line telecommunication service which is offered to users **with** telecommunication equipment and resources. For example, all telecommunication services, from POTS to multimedia services and/or internet related services, are categorized as session related service.

6.1.2 Classification of Sessions

This section considers the classification of sessions. A number of classification schemes are possible, but they are all based on the TINA session object. This generic object defines the common properties of a session. All derived objects, regardless of their classification, inherit these common attributes and operations, such as: session identifier, session type, state, terminate, suspend, and resume.

A common TINA classification is based on the service (functional) separations of access, service and communication. Sessions have been identified to support each of these separations. Though specialized to support the particular requirements of a functional area, each session retains the common properties of a TINA session.

It is also helpful to use the domain session and domain session binding introduced in Section 6.1.1 to help classify sessions. Again, the TINA session object is the root of the hierarchy[3], and the domain session and domain session binding inherit its properties. This type of classification can be combined with the previous scheme to classify all TINA related session objects for each separation. The following sections will use this combined scheme and the following aspects to classify objects:

- User aspects: They represent the entities, semantics, constraints and rules governing the availability and usage of service capabilities with respect to a specific user, and the resources and mechanisms needed to actually support the service capabilities according to a specific user standpoint;
- Provider aspects: They represent the entities, semantics, constraints and rules governing the provision and usage of service capabilities to users, as well as the resources and mechanisms needed to actually support those capabilities;
- Peer aspects: They represent both of the user and provider aspects.

6.2 Access related Information Model

6.2.1 Classification of Access Sessions

Access sessions can be classified in terms of a specialization hierarchy as shown in Figure 6-2. The classification is along the separation into user, provider and peer aspects (as described in Section 6.1.2).

Relating this to Figure 6-1, the *Domain Access Session* corresponds (by subclassing) to the domain session, and the *Access Session* corresponds (by subclassing) to the domain session binding.

3. The TINA session object is an abstract object as defined in [51], which means this object is not instantiable.

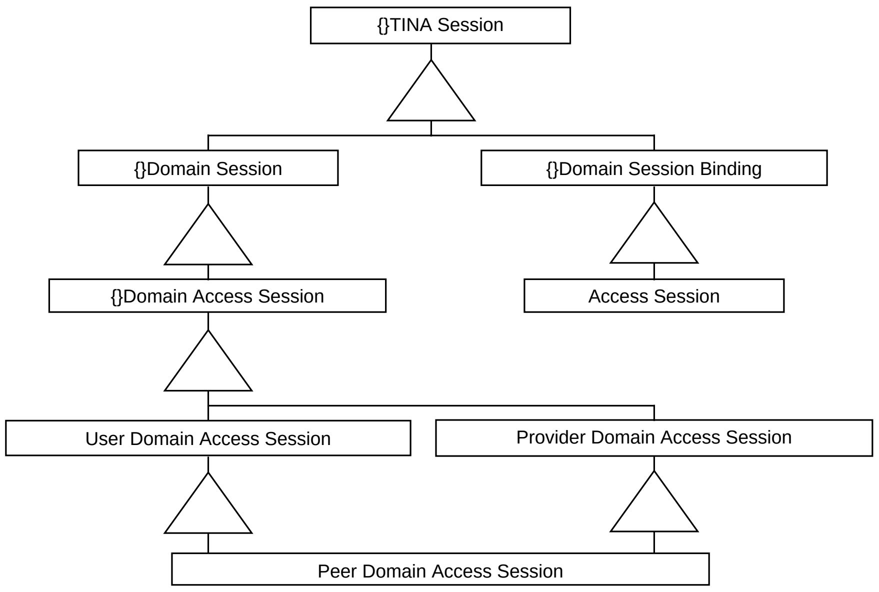


Figure 6-2. Classification of the Access Session

6.2.2 Access Session

The access related information objects and their relationships are shown in Figure 6-3.

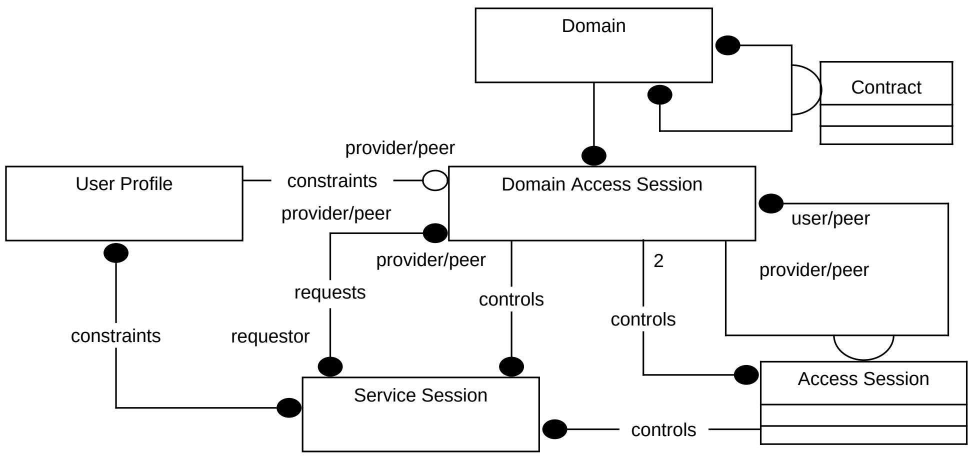


Figure 6-3. Relationships between Access Related Information Objects

Each domain access session is associated with a particular access role. Three roles have been identified: user, provider, and peer. The user role accepts invitations and makes requests on the associated domain access session. The provider role accepts requests and sends invitations to the associated domain access session. The peer role combines these user and provider roles.

The access session is categorized by the roles supported by each D_AS. These roles depend on the relation between domains. The following access session types may be required:

- **Asymmetric type access session**: One domain acts in the user role and the other domain acts in the provider role;
- **Symmetric type access session**: Supporting symmetry in the access session, both domains support both roles.

The information objects and their relationships for the asymmetric type access session are shown in Figure 6-3. Figure 6-3 presents a more detailed version of Figure 6-3 for the asymmetric type access session. An asymmetric session is supported by the User Domain Access Session and the Provider Domain Access Session. A symmetric session would be supported between Peer Domain Access Session components, which combine features of the User Domain Access Session and the Provider Domain Access Session.

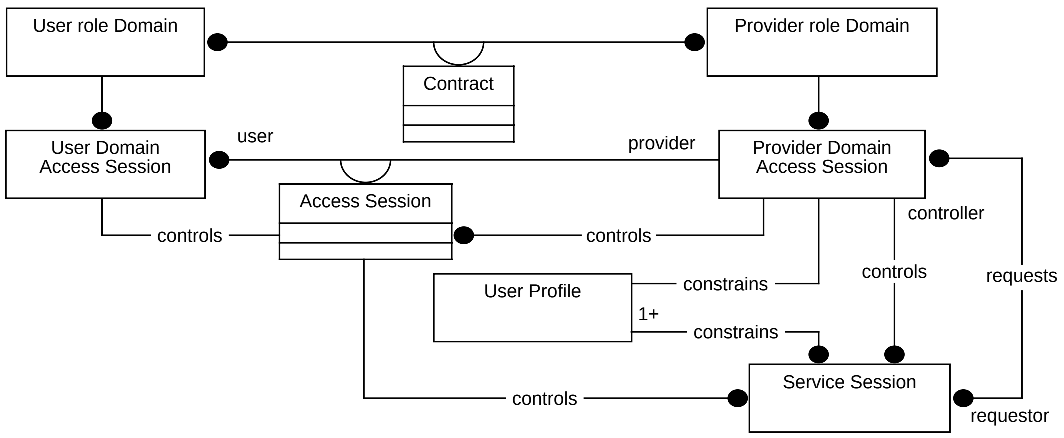


Figure 6-4. Information objects and their relationship for asymmetric type access sessions.

6.2.2.1 Domain Access Session (D_AS)

This is an abstract object which represents the generic information required to establish and support access between two domains. Each domain access session is associated with a particular domain. However, a domain may have many domain access sessions. A domain access session is usually associated with one (or possibly more) contractual relations with another domain. There may be multiple domain access sessions within the domain for each contractual relationship.

The D_AS is specialized into user domain access session and provider domain access session type informational objects. These two objects may be further specialized into peer domain access session information objects. They are all described in the following sections. See also Section 6.2.1.

6.2.2.1.1. User Domain Access Session (UD_AS)

This object is managed by the user and represents the collection of capabilities and configuration that the user employs to contact a provider. The UD_AS holds user defined policies that determine the terms of the interaction with a provider, such as security policy and accounting; they form the basis of the negotiation with a provider in the establishment of a mutually acceptable access session. The UD_AS may comprise one to many terminals or DPE nodes. Whatever the UD_AS configuration, both the user and provider will have a perspective on how trustworthy the UD_AS is. This trust level (e.g. confidentiality, password protection, cipher implementations) will be reflected in the management restrictions imposed on the access session (e.g. refusal of high value services). For example, a UD_AS supported on a tamper resistant terminal supplied by the provider is more trustworthy than a small multi-user PC LAN.

6.2.2.1.2. Provider Domain Access Session (PD_AS)

This object is managed by the provider and can be considered to be created when the user becomes a recognized, identifiable entity with specific capabilities and data within the provider domain. The session terminates when the user ceases to have any user relationship with the provider (so the provider stops holding permanent information about the user in the User Profile, see Section 6.2.2.3). This object knows persistent information about the user. Part of this information specifies policies that determine the terms of interaction with the user, such as security policy and accounting. These policies form the basis of the negotiation with the user in the establishment of a mutually acceptable access session. Both the user and retailer will have a perspective on how trustworthy the PD_AS is. This trust level (e.g. confidentiality, password protection, cipher implementations) will be reflected in the management restrictions imposed on the access session.

6.2.2.1.3. Peer Domain Access Session (PeerD_AS)

This object represents a domain access session that can support a symmetric relation between domains. As such, it must include the information and support the capabilities associated with both the user and provider roles. The information and required capabilities are explained by the UD_AS and PD_AS definitions, respectively. The PeerD_AS may be considered to be derived from both the UD_AS and PD_AS. Besides, it includes additional information required to support a symmetric access session.

6.2.2.2 Access Session (AS)

This object is managed by the user or the provider via the D_AS, and represents the collection of dynamic information for a binding between D_ASs, such as security policy, accounting and session description for this binding. This object terminates when the user or the provider ends the relationship (dynamic binding between D_ASs) with the provider or the user.

6.2.2.3 User Profile

The User Profile contains all information that is used directly by the D_AS for authorization decisions, constraints and customization of the D_ASs, Access Sessions (within Access Session Bindings) and Service Sessions. The user profile aggregates four other objects that describe information specific to a user:

- **Usage Context**: Specifies a configuration of network access and terminal equipment which defines the user's domain. A usage context details the configuration of the user domain in order to gain access and/or use services according to a contract. It may also refer to security and accounting contexts. For each access session there will be a specific usage context for the user, and it will constrain the invocation of service sessions within that access session.
- **Service Profile**: Specifies for the user whether or not a service can be invoked, and its characteristics when invoked. It contains information that specifies retailer imposed constraints, subscriber imposed constraints and preferences set by the user. This information may be altered by the subscriber. Service profiles may also refer to security and accounting contexts. The service profile may be constrained by one or more usage contexts.

- **Session Description**: Specifies an existing service session which the user is either active in, or entitled to resume, join or schedule. A session description is created when a user creates a service session (either joining an existing service session or after receiving an invitation), suspends a service session or his/her participation in it, or schedules a service session. A session description may be constrained by the usage contexts and service profile. A user profile may have many session descriptions. An example of session description information is:
  Service type **|** session ID **|** date-time suspended **|** valid until date-time **|** participant list
- **Registration**: Specifies actions to be taken when an invitation or scheduled session is received. The service type selects a handling procedure, such as refuse, deliver to a specific location registered by the user, start a service session, etc. Information in a registration may be used to resolve a user's location by agreement between the user and the provider.

The relationship between these information objects is shown in Figure 6-3.

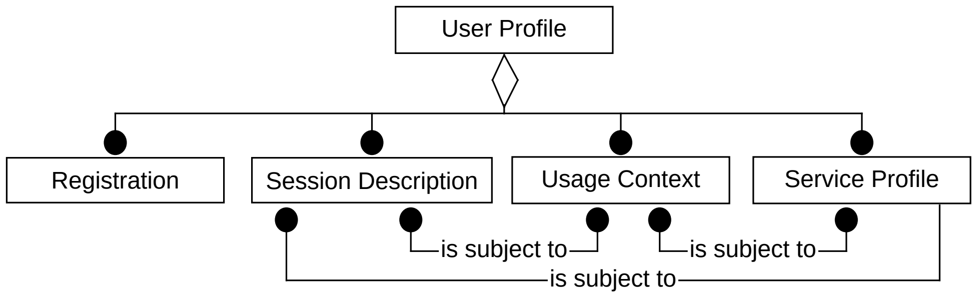


Figure 6-5. Relationships between user profile related information objects.

6.3 Service Session Information Model

6.3.1 Classification of Service Sessions

Figure 6-6 gives the classification hierarchy of service sessions along the separation into user, provider and peer aspects (as described in Section 6.1.2). The classifications are described in detail in Section 6.3.2.

Relating this to Figure 6-1, the provider service session (PSS) and the domain usage service session (D_USS) correspond (by subclassing) to the domain session, and the domain usage service session bindings corresponds (by subclassing) to the domain session binding.

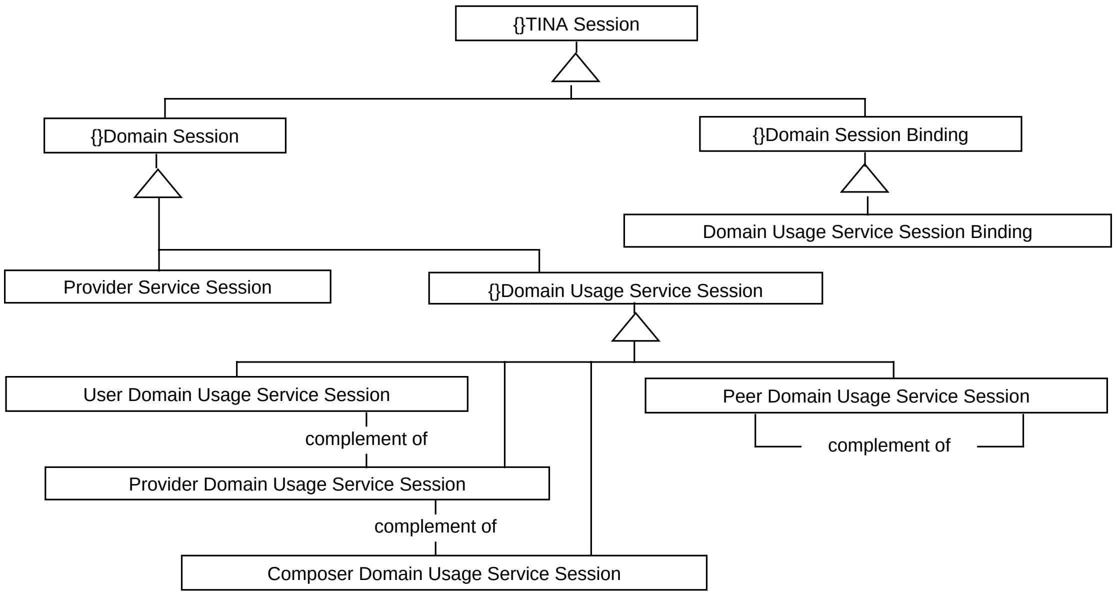

{}TINA Session
{}Domain Session
{}Domain Session Binding
Domain Usage Service Session Binding
Provider Service Session
{}Domain Usage Service Session
User Domain Usage Service Session
Peer Domain Usage Service Session
complement of
complement of
Provider Domain Usage Service Session
complement of
Composer Domain Usage Service Session

Figure 6-6. Classification of the service session.

6.3.2 Service Session

The service session related objects and their relationships are shown in Figure 6-7.

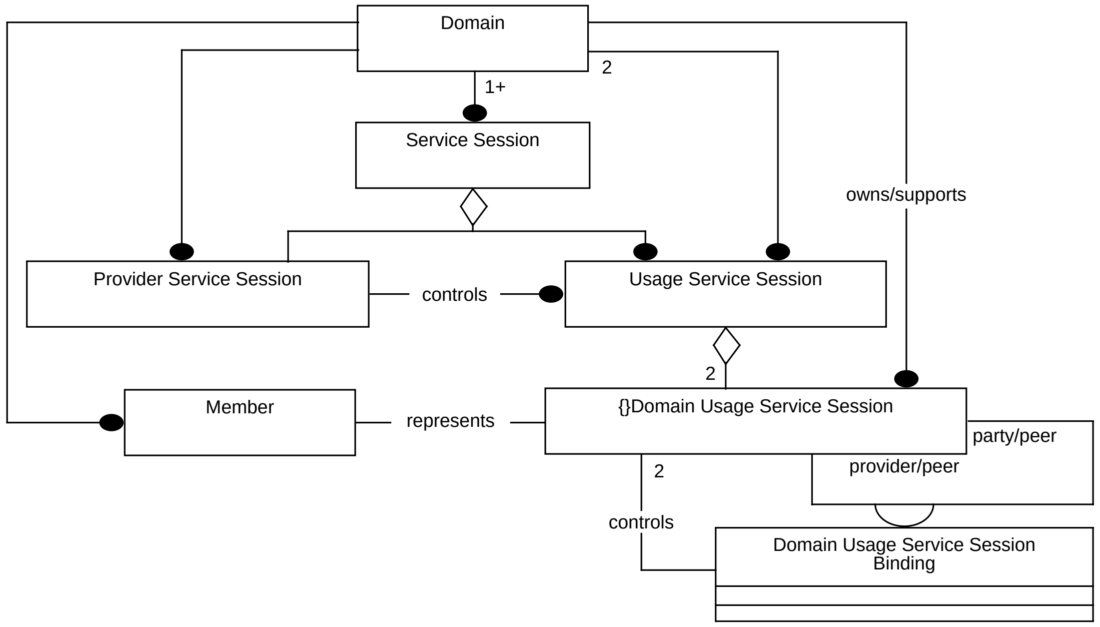


Figure 6-7. Service session related objects and their relationships.

The service session consists of usage and provider service sessions. Each usage service session can extend over two domains and is composed of two complementary domain usage service sessions (D_USSs), where a complementary D_USS is one that can interact with another. Each member of a session, i.e. an end-user, resource, or associated session, is associated with a usage service session.

The type of D_USS supported depends on the perceived role of that member in the service session. Figure 6-6 shows the possible D_USSs. The service architecture supports the following generic usage roles:

- Usage Party: Resembles an “end-user” of a service, an active role which may make requests and is sent notifications of session changes.
- Usage Provider: Provides a service or acts as a resource to another entity. This is a passive role in the sense that it cannot initiate actions, but responds to requests or notifications of changes.
- Usage Peer: This is a symmetric type role in which both entities can take actions on the other and share information about themselves. This role supports federation and some kinds of composition paradigms.

More roles are possible. These include service specific roles, which are outside of the scope of the architecture, and control and management roles to support relationships in composed services (see Section 4.2.2).

When a service session is started, or when a new member joins a service session, it acquires the relevant user profile information from the access session or domain access session (e.g. service description) for the member. This constrains the usage service session and potentially the provider service session. In the case of multi-member service sessions, an individual’s user profile or current usage configuration may affect the whole service session, depending on the nature of the service and its management policies.

A service session can be instantiated by an access session, domain access session, or another service session. The initiator of a service session associates it with its members(s). Members may have different responsibilities within the session (e.g. management or purely interaction with the service content and general session control). If a service session is the responsibility of an access session, the service session can remain active while that access session is active. When an access session is ended, related usage service sessions must be ended, suspended, or transferred to another access session or domain access session.

6.3.2.1 Provider Service Session (PSS)

It contains a central view of the service session, including all members and any additional provider information and logic necessary to execute service requests and maintain the session. Support of this session is the responsibility of the provider. A provider service session represents the service capabilities common to multiple members. Generally the provider service session holds information objects related to the management view of the service (e.g., accounting) or system related information of the service.

6.3.2.2 Usage Service Session (USS)

This is the member’s (e.g., an end-user’s) customized view of a service. It hides service complexity from the member and ensures that the member’s preferences and environment are supported by the service. It hides the heterogeneity of each usage configuration from the provider service session. It will be further decomposed into domain usage parts.

6.3.2.3 Domain Usage Service Session (D_USS)

This object is an abstract object which represents generic information required to establish and support a service between two domains for an associated session member. Each domain usage service session is associated with a particular domain and a particular service session. As this object is an abstract object, it is not instantiable. The D_USS is specialized into four types of new information objects as described in the following sections.

6.3.2.3.1. User Domain Usage Service Session (UD_USS)

This is the functionality and information present in an end-user domain, e.g. a consumer domain, to support the usage service session and allow the end-user to interact with the service. In all cases, the UD_USS is associated with the usage party role. It is the responsibility of the end-user domain to deploy and manage this session. However, certain deployment and management responsibilities of resources in the UD_USS may be assigned to the provider domain by agreement.

6.3.2.3.2. Provider Domain Usage Service Session (PD_USS)

This is the functionality and information present in a provider domain (e.g. retailer, third party provider) to support the usage service session in a usage provider role. It hides the complexity and specifics of the other domain from the PSS and isolates party specific activity from the general activity of the PSS, which is common to all participants in the service session. The domain in the usage provider role is responsible for deploying and managing the provider domain usage service session.

6.3.2.3.3. Peer Domain Usage Service Session (PeerD_USS)

This is the functionality to support a peer relation between two sessions in a composition or federation relation. It hides the complexity of the other domain from the PSS and isolates the peer member from the general activity of the PSS, which is common to all participants. Each domain in the peer relationship is responsible for deploying and managing a peer domain usage service session to support the relationship. It is supporting the usage peer role in a provider domain.

6.3.2.3.4. Composer Domain Usage Service Session (CompD_USS)

This is the functionality and information present in a domain to support a usage party role in a composition relation between service session. This type of relationship is supported by a usage service session.

6.3.2.4 Domain Usage Service Session Binding (D_USS_Binding)

It represents the dynamic information associated with binding two D_USSs. The D_USSs control this information. The information contained here is determined by the type of D_USS participating in the binding, and the session model(s) and feature set(s) associated with the binding.

TINA uses the service session graph model to express the information associated with the binding, (see Section 6.4).

6.4 The Service Session Graph Information Model

The information model of the usage part of the TINA Service Architecture is mainly based on the concept of *Service Session Graph*[4] (SSG). Figure 6-8 illustrates how the service session information model relates to the service session graph information model that is explained in more details in the following sections.

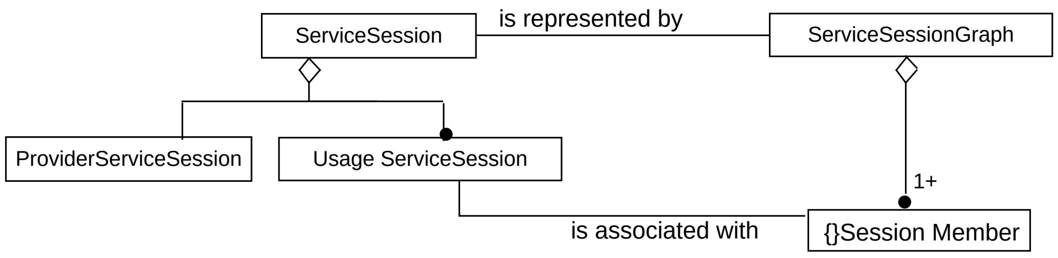


Figure 6-8. Relationship between overall session information model and service session graph.

The service session graph has to be considered as the overall framework on which the different feature sets in TINA are based. This means that a service session does not need to use all these concepts if only some are needed. A highlevel description of the feature sets can be found in Section 3.6.2 and more details in Section 7.2.6. The relationship between TINA session model features sets and the session graph information objects are explained in Section 6.4.7.

The basic concept used to describe the information contained in and exchanged within service session control is the Service Session Graph (SSG). It is important to make the distinction between a particular instance of a service session graph and the concept itself: the concept provides the tool with potential operations and elements which are generic (service independent), while a particular instance is a specific instantiation or activation of one or more of those elements. The SSG information model is valid for both the local (user) views and the central (provider) view.

6.4.1 Overall View of the Service Session Graph

The service session control's SSG information model is described below using OMT notation, by means of the diagram in Figure 6-9.

First of all, the SSG is composed of (aggregates) several objects as shown in Figure 6-9. The figure just gives a *first idea* of the model and *is neither complete nor self explanatory*. The classes, relations, and cardinalities and their usage will be explained in more detail in the document: “Service Component Specifications” [14].

With respect to Service Architecture 4.0 the peer session member and the composing session relationship have been added in order to support service composition and federation. See Section 4 for further details on these issues.

The service session graph model defines all the generic classes that are necessary to fulfil its mission: offering a generic framework to describe information in service sessions. The concept is used to model and control the state of a TINA service session. An instance, at a certain point of time, of a service session graph models a “snapshot” of the resources, the parties, the peers and the relationships established into the service session. Note that in Figure 6-9 the cardinality of most relationships

4. Sometimes the shorter term session graph will be used instead of service session graph.

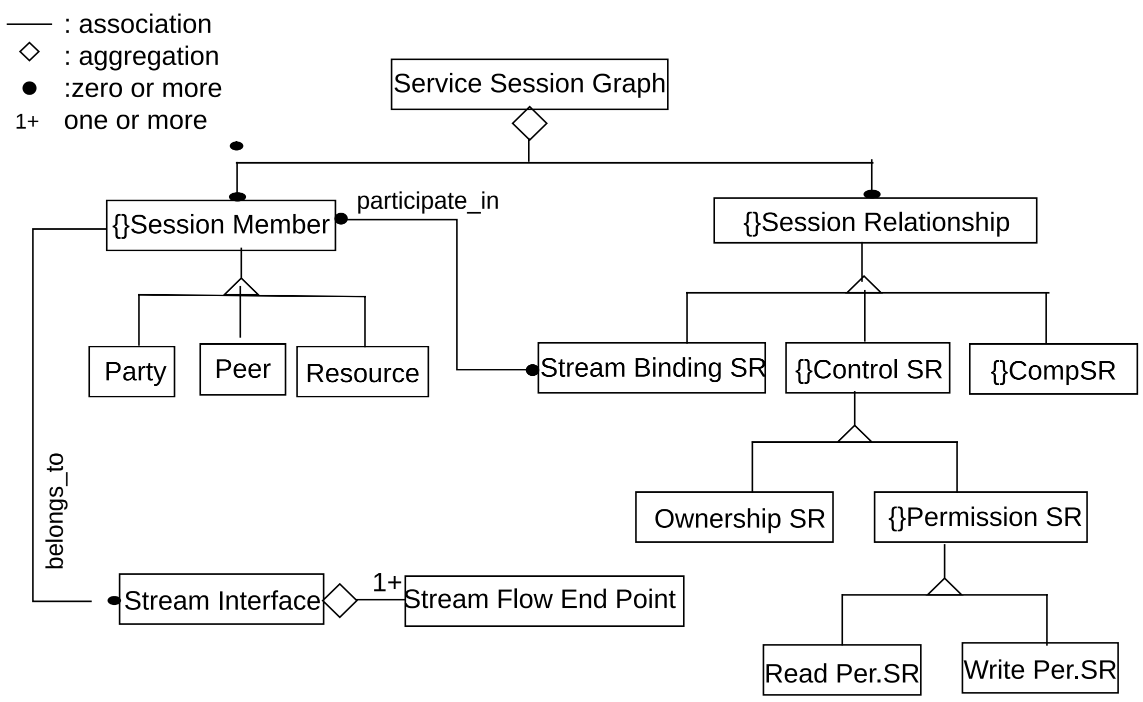


Figure 6-9. Service Session Graph aggregation.

is defined as “zero or more” for reasons of flexibility. Most of the classes described in the model are optional (zero instances when not used), or can be used many times (for multi-party and multi-media requirements, as many instances of the classes will be required as parties and media involved). This flexibility allows the reuse of the model for every feature set that has been defined. Details of the information model are given in the following sections.

The service session information objects are manipulated by the User Application, User Service Session Manager, Composer Usage Session Manager, Service Session Manager and Peer Usage Session Manager components, which are defined in Section 7.2.6: they provide the interfaces for modifying the service session. The detailed interface specification (including computational operation signatures) is within the scope of the service component specifications document (SCS [14]).

In the information viewpoint this corresponds to operations defined for each IO class in the session graph model.

These high-level operations[5] are:

- **Creation**: This type of operation will initiate the creation of a new object in the service session; it includes its configuration, i.e., the need for specifying the objects on which this new object depends, and setting its attributes. Default value mechanisms will be foreseen. The relation’s cardinality is checked at creation time;
- **Modification**: This type of operation will be used to modify an object's attribute value at any time in the service session's lifetime;
- **Deletion**: This type of operation will be used to remove one or more objects and their dependent objects from the SSG;

5. Sometimes these operation are not offered to external clients.

- **Suspension**: This type of operation will be used to put an active object into the suspended state;
- **Resume**: This type of operation will be used to put a suspended object into the active state.

Remark: the creation/deletion/modification of an object can cause side effects in the session graph, such as the implicit creation/modification/deletion of other objects. This is the case where an imported member is created: it implies the creation of a Subsidiary Session Relationships and vice versa (these relationships are defined in Section 6.4.6). Besides, the deletion of a Session Member generally implies the implicit deletion of the Session Relationships it is participating in.

The following sections provide a high level description of all SSG information objects classes.

6.4.2 Service Session Graph Class

The **service session graph** class is the top class of the information model, and represents the service session as a whole. It contains (aggregates) all other objects in the model. It can contain information for scheduling the entire service session.

6.4.3 Session Member, Party, Resource and Peer Classes

Party. It is defined as an active[6] (initiating) entity taking part in the service session: it can be either an end-user, a subscriber, or a service or resource provider. It models a (potential) “user” in the service session. The party object is used to maintain information about end-users in the service session and/or any service sessions participating in the service session in the "session party" role. Parties have a party name attribute that uniquely identifies them.
An end-user or service session can request to take part in the service session by requesting to become a party in the SSG. A party can invite another one into the service session. If there are already other parties involved in the same service session, they might have to confirm the invitation while the service session negotiation is taking place. A party can request the deletion of another party (or itself) from the service session. Again, this might involve negotiation.
In general, any negotiation of any operation handled by the service session will have a behavior that depends on the presence of control session relationships. Parties may be negotiating entities and act as controllers in control session relationships. The meaning of negotiation in this context is further explained in Section 6.4.5.
A SSG contains zero or more party objects. There are as many parties as end-users or sessions in the session party role. The party is the only class that is present in every feature set.

Peer. It represents a session participating in another session, where both sessions can initiate actions on the other. A peer may issue requests and act on the session graph of the session in which it participates. Peers are generally associated with composition and federation relationships. They may not be requested directly, but may result from an invitation to an end-user in another retailer domain, or a request to merge or associate with another session. The service logic determines when such a member is necessary. Peers can be related to a number of subsidiary session members via a Composing Session Relation, see Section 6.4.6. A subsidiary cannot be acted on without interaction with the other member (peer or resource). The modifications on a subsidiary member are either delegated or notified to the peer, depending on the specific type of composing session relationship established between peer and subsidiary objects (further details can be found in Section 6.4.6). This relationship allows a session to act on the session graph of another session via the peer. A peer does not usually participate in Control Session Relationships, but can be used as an intermediation object in a nego-

6. If multipartyIndFS is supported it is also a “negotiating” entity (see Section 7.2.6.2 for further details on Feature Sets).

tiation process. This would be the case when a subsidiary member owns a non-subsidiary member. Peers may only participate in these relationships when they are not related to any subsidiary object[7]. A SSG contains zero or more peer objects. There is a peer for each peer related session visible to the session. This class may be associated with federation and composition feature sets.

Resource. It models a source of support for the execution of the service (session). It can model many types of resources to be identified or shared in the service session (e.g., a file to be retrieved, a shared pointer in that file, a conference bridge, a service subscription file, a VoD server, or another service etc.). The resource is identified as a part of the service session upon request of parties or the service logic itself.
A resource plays a “passive” role in relation to parties. It cannot initiate requests or participate in negotiation or voting. Resources may only respond positively or negatively to requests. Resources can, however, notify the session of changes in their own status. If a resource represents some subsidiary service, then it may support composing session relations with other elements of the session graph.
The SSG contains zero or more resource objects; there will be as many resource objects as resources used in the service session. Resource is the main information object class of the Resource Feature Set. It may also be used in the stream binding and some composing feature sets.

Session member abstract class: The party objects, the resource objects and the peer objects that have been described above have several commonalities. First, they both will require interconnection: connections between parties (and peers) for audio and video communication, connections between parties (or peers) and resources for information retrieval, connections between resources for information transfer, etc. This shows the need for defining means of communication for both the party, peer, and the resource class. Another commonality is that the party, the peer and the resource will both require scheduling. In order to fully exploit the strength of object oriented modelling and enable the service designers and implementors to reduce redundancy, a class generalization of the party, peer, and resource classes will be defined: the Session Member (SM) class. This class is abstract because it cannot be instantiated: only the specialized classes can. Since party, peer, and resource classes are now defined as specializations of the session member class, they will be more completely named as, respectively, Party Session Member (PartySM), Peer Session Member (PeerSM) and Resource Session Member (RSM)[8].

Sometimes it is useful to group parties and/or resources or other session members in order to be able to define relations in the service session that are valid for each member of such a group. The following groups are defined: Session Member Group (SMG)[9], Party Session Member Group (PartySMG)[10], Peer Session Member Group (PeerSMG) and Resource Session Member Group (RSMG)[11]. These classes are not depicted in the overall diagram of the TINA session model; more details can be found in the Service Component Specification.

7. In this case the peer is the only visible entity from the ‘peer related session’.

8. The terms party and PartySM will be used as synonyms and likewise for resource and RSM.

9. **SMG:** it can contain any number of session members of any kind (indifferently parties, resources and peers, even mixed together).

10. **PartySMG**: it is a specialization of the SMG because it will contain parties only; note that this class could carry Closed User Group (CUG) identification.

11. **RSMG**: it is a specialization of the SMG because it will contain resources only.

6.4.4 Stream Binding Related Parts of the Service Session Graph

In order for a session member to express its ability to communicate with other session members by means of streams[12], the session member is associated to stream interfaces belonging to it, as shown in the overall figure (Figure 6-9, page 85). The stream binding session relationship class represents the binding between parties or their corresponding stream interfaces and stream flow endpoints.

Stream Flow Endpoint (SFEP): It represents the termination of a single flow by an application. In other words, it can be seen as the logical representation of a physical stream device I/O port of a terminal (e.g. the outlet of a camera). Further details can be found in NRA[8].

Stream Interface (SI): It represents a dynamic grouping of SFEPs. The session member will associate as many stream interfaces as required (eventually none, when the stream binding feature set is not supported); the cardinality is shown in the Figure 6-9. It's worth noting that the session member - stream interface relationship is inherited by the SM's specialized classes: party, peer and resource. The stream interface class used here is defined in NRA (see [8]).

Stream flow connection (SFC): It describes point-to-point or point-to-multipoint connections between SFEPs (see Section 6.5). Figure 6-10 describes stream flow connections and the relationships between SM, SIs, SFEPs and SFCs.

Stream binding session relationship(SBSR)[13]: It can be described by a number of different information models. The stream binding information model is related to the setup of the binding (i.e., it is used to describe the stream binding we want to set up) and any explicit setup and modification control operations. Many models are possible, but we are most interested in those that relate the stream binding to SI and SFEPs, since these have computational aspects that may be supported by a DPE. TINA has defined three different ways to provide stream binding; both of them are based on the concept of stream flow connections.

TINA has defined three different ways to express stream bindings, all of which are related to Figure 6-10:

Participant oriented stream binding: In this high level approach, stream bindings are described in terms of participating session members, type and QoS. Each participant is then requested for suitable stream interfaces which are to be bound by the stream binding. This high level description can be mapped by the service logic to a number of stream flow connections. These stream flows may be of different types and QoS. Special resources, such as bridges, may also be introduced to support the binding. The stream binding may be manipulated by modifying the type, QoS and participants.
This type of stream binding specification allows a high level request of multi-party-multi-media stream bindings[14]. The type and QoS parameters may be used to implicitly specify multimedia requirements[15].

Flow oriented stream binding: This approach to stream bindings uses stream flow connections (SFCs) to specify the stream binding. In this case, a stream binding may be considered an aggregation of SFCs, and the session members participate indirectly by exporting their SI information before any request to establish a stream binding is made.

12. As opposed to using operational interfaces.
13. Stream binding and stream binding session relationship (SBSR) will be used as synonyms. Normally 'session relationship' will be suffixed when the emphasis is on SBSR as a subclass of session relationship (SR).
14. (i.e., one stream binding representing a multipoint-to-multipoint binding, that maps onto a number of stream flow connections, each representing point-to-multi-point connections between SFEPs).
15. However, it is also suitable for simple stream bindings with little overhead (i.e., this approach allows for a computationally 'lightweight' way of setting up stream bindings).

A mapping is no longer required between the stream binding and the SFCs that support it. Since there is no mapping, service level special resources must be specified by the stream binding requester. The stream binding may be manipulated by adding, deleting and modifying the SFCs. Each SFC is specified in terms of type, branches, state, and QoS. The SFCs may be manipulated by adding, removing and modifying branches.

Simple flow oriented stream binding: this is a simplification of the previous one, where only one SFC is allowed in each stream binding[16].

Depending on the type of service, there may be different needs for explicit manipulation of individual SFEPs and SFCs. Hence, TINA supports all three models. There might also be cases where SBSR groups may be defined[17]. The different feature sets for the three different approaches are listed in Table 7-2, page 112. An example of the use of the participant oriented stream binding setup is given in Section 7.4.6.

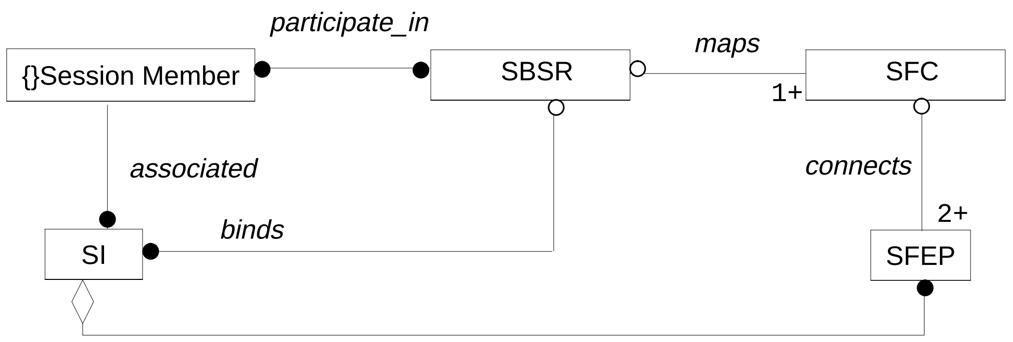


Figure 6-10. Stream binding model

6.4.5 Control Session Relationship

The **Control Session Relationship (CtrSR)** class allows enhanced control to be performed on the service session objects. It supports the mechanism for negotiation and voting, and therefore it is the basis for the Control and Voting Feature Sets. In other words, if the Control Features Set is not supported, this information object is not present and neither negotiation nor voting can take place.

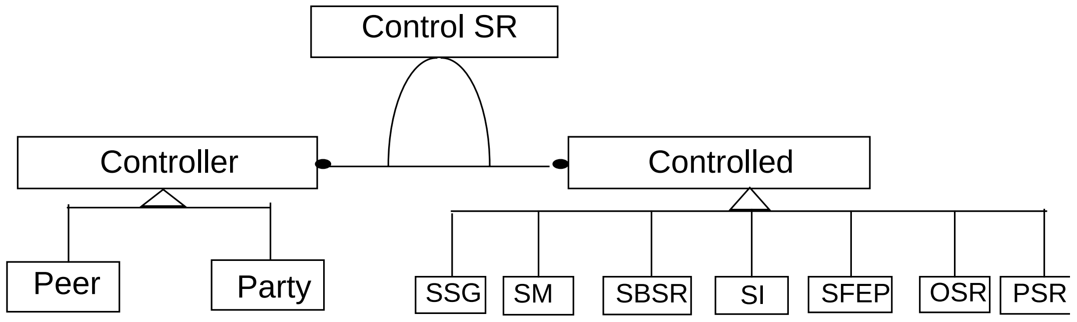
Several specialized classes are defined for the CtrSR, for each specific control requirement, as shown in Figure 6-9 page 85. Their usage is explained below.

When a control session relationship has to be established between two objects in the SSG, there will always be one object having the “controller” role, and another having the “controlled” role (as depicted in Figure 6-11).

In a similar way, as was done for the session members, parties, resources etc., it might be useful to define concepts allowing for grouping of session relationships. It may also be necessary to group classes for control SR only, and for SBSR only. These new grouping classes are called: *Session Relationship Group* (SRG), *Control Session Relationship Group* (CtrSRG), *Stream Binding Session Relationship Group* (SBSRG) and *Composing Session Relationship Group (CompSRG),* respectively. They are quite intuitive and will not be shown in formal OMT diagrams in this document.

Control Session Relationship can be further specialized in Ownership and Permission Session Relationships.

16. This simple one will thus only allow for point-to-multi-point communication.
17. SBSRs may also be grouped together with CtrSRs into SRG.



Note: The ‘zero or more’ cardinality bullets in OMT do not relate to the describing class for the relation (in this case control SR class), so the correct reading of this diagram is:
a control SR instance must always have one controller and one controlled class instance;
a controller instance can be related to zero, one or more control SR instances;
a controlled instance can be related to zero, one or more control SR instances.

Figure 6-11. The control SR.

The **Ownership SR (OSR)** associates a party SM or a peer SM (a controller class object) to a controlled class object in the SSG. It is used to force multi-party negotiation when an operation is requested on an object that is owned by one or more other parties. The Ownership SR specifies which party SM or peer SM owns that object instance in the SSG. The ownership of that object instance includes its eventual attributes, its attachment to associations / relations with other objects, the objects it eventually aggregates, and the OSR itself. The OSR allows multiple party and peer SMs and/or party SMGs to be owners of an object instance at the same time (multi-party ownership).[18]

When one or more controller objects have an OSR relation with a controlled object, they will be called “owners” of that object, and the controlled object will be “owned”. As a consequence of this OSR relation, the behavior of the negotiations in the service session will be modified. When a party (e.g., an end user) requests a modification of the controlled (owned) object, the service session will:

- authorize the modification if the requesting party or peer is one of the owners,
- start a negotiation with the owners if the requesting party or peer is not an owner.

If a negotiation is required, each owner will be informed of the request, and will have to answer either positively or negatively. An attribute of the OSR will indicate the voting rule in effect for this OSR instance.

The possible “modifications” of the owned objects include the following:

- a modification of one or more of the attributes of the owned object,
- a modification of the associations of the owned object
  (adding or removing such an association),
- adding or removing any object aggregated by the owned object,
- adding to or removing the owned object from a “group” object.

18. Every party/peer owner of an object has “full-authority” on the owned object, i.e. no negotiation is required to fulfil owner requests. Joint-authority is only supported when the owner is a SMG, i.e. negotiation among the parties in the owner group has to take place before actions (on the owned object) can be done.

The OSR control SR allows services to customize negotiation on a service session, and enables the concept of “third party control” for any operation taking place in the service session.

The OSR is meant to be established by the owning entities themselves.

Default situation: when no OSR is specified in the SSG, it means that all parties (and party groups) have a shared ownership of all the object instances in the service session; no negotiation is required. The only exception to that rule is that a party implicitly owns itself, which includes its eventual attributes, its attachment to associations / relations with other objects and the objects it eventually aggregates. These default rules are important because they mean that the initial situation in a service session (when no OSR has been defined yet) is a shared ownership. This way most of the operations can take place without negotiation.

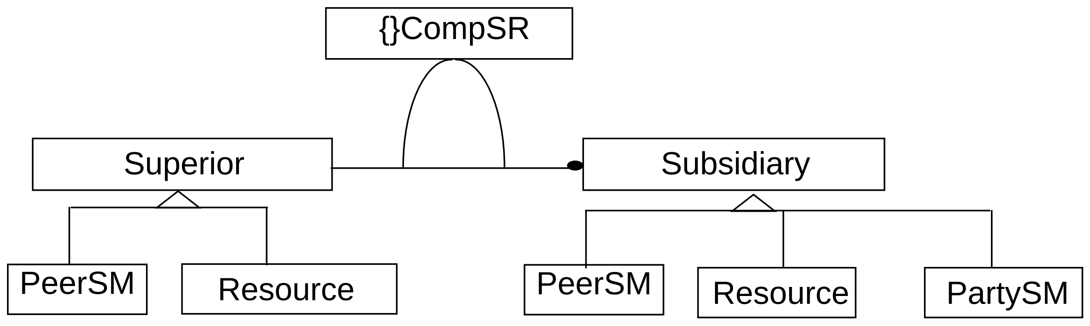
The **permission SR** class can be further specialized in read-permission and write-permission so that they can be modeled separately (see Figure 6-9). Their usage is briefly described below.

- **Read Permission Session Relationship (RPSR)**: It associates a party/peer or party-SMG (a controller class object) to a controlled class object in the SSG. The read permission SR specifies whether or not that party/peer or party-SMG has the ability to see that object instance in the SSG. The visibility (or non-visibility) of that object instance includes its eventual attributes, its attachment to associations / relations with other objects, the objects it eventually aggregates, and the RPSR itself. This means that if the RPSR specifies a “read: no”, the RPSR itself will not be visible to the concerned party/peer or party-SMG, just like the controlled class objects it is hiding. The RPSR is meant to be established by a “third party” entity (party/peer), in order for that third party to be able to hide parts of information from other parties in the service session.
  Default situation: when no RPSR is specified, it means that all parties/peers (and party groups) have read permission on all the object instances in the service session. If information hiding is required, specific RPSRs have to be established.
- **Write Permission Session Relationship (WPSR):** It associates a party/peer SM or party SMG (a controller class object) to a controlled class object in the SSG. The write permission SR specifies whether or not that party/peer SM or party SMG has the ability to write on (modify) that object instance in the SSG. The modification ability (or non-ability) on that object instance means the ability to modify or delete it, which includes its eventual attributes, its attachment to associations / relations with other objects, the objects it eventually aggregates, but not the WPSR itself. The WPSR can be established by a “third party” entity (e.g., a PartySM), in order for that third party to be able to inhibit other parties to modify some parts of the service session.
  Default situation: when no WPSR is specified, it means that all parties have write permission on all the object instances. If write inhibition is required, specific WPSRs have to be established.

6.4.6 Composing Session Relationships

The Composing Session Relationship (CompSR) provides the required support for representing composition and federation of services in the session graph. It relates an associated session, represented by a superior object (peer or resource), with a subsidiary session member, as shown in Figure 6-11.Modifications on the latter are forwarded or notified to the former. Note that in a CompSR a subsidiary can only have one superior.

.



Note: The ‘zero or more’ cardinality bullets in OMT do not relate to the describing class for the relation (in this case control SR class), so the correct reading of this diagram is:
a CompSR instance must always have one Subsidiary and one Superior class instance;
a superior instance can be related to zero, one or more CompSR instances;
a subsidiary instance can be related to one CompSR instance.

Figure 6-12. The Composing Session Relationship.

The CompSR is an abstract class. As shown in figure 6-13, it is specialized into two classes that represent the possible relationships between session members and session composers (peer or resources): Share Session Relationship (ShSR) and Subordinate Session Relationship (SubSR)[19].

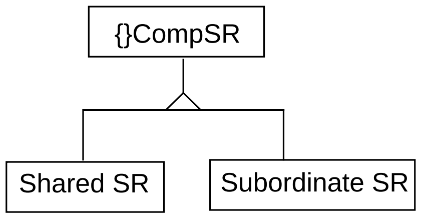


Figure 6-13. Specialization of the ComposingSR.

- **Shared Session Relationships (ShSR):** The ShSR associates a subsidiary session member with a superior object representing a session that needs to be informed of changes to the subsidiary session member. It may also indicate that a particular session, via the superior object, can issue modification requests on a specific subsidiary session member. The ShSR can be created independently from the session member. The creation of a ShSR in a SSG represents that a particular session member has been made known to an associated session. This implies the creation of a CompSR to a corresponding member in the SSG of the superior's associated session. The deletion of a ShSR represents that the subsidiary is being hidden from the associated session and implies its deletion in the superior's associated session SSG. The deletion of the superior or subsidiary causes the deletion of the ShSR.
- **Subordinate SR (SubSR):** The SubSR relates a subsidiary session member with the superior member that represents the session which is in charge of it. Any modification on the subsidiary session member is forwarded to the superior session. The superior mem-

19. Examples on the use of Composing Session Relationships can be found in the Annex (Section A-4.4).

ber notifies the session of any changes or requests to change on the subsidiary members. There is a mutual existence dependency between the SubSR and the subsidiary session member.
The SubSR can only be created together with the subsidiary object. A subsidiary object cannot exist without being related through a SubSR to a superior object. This means that a member belonging to another session cannot appear in a SSG without indicating which session is in charge of it. Because of that, a subsidiary session members also depends on the existence of the superior session. The deletion of a superior session member will imply the deletion of all the subsidiary members.
The existence of a Subordinate SR in a session implies the existence of a ShSR for the corresponding subsidiary member in the associated session.

CompSRs have a direct influence on the SSG of a service session associated with the superior member. When the Superior of a CompSR is a peer, a corresponding CompSR in the other service session is implied. For a SubSR, this will always be a ShSR. However, a ShSR may correspond to either a SubSR or ShSR. Due to this fact, both session graphs are said to be correlated. This correlation keeps the required consistency between SSGs in different sessions participating in a service (session).

Parties cannot explicitly create CompSRs. The service logic, following the policies for service composition and federation, creates them after establishing a relation with the associated session. They are created when a session member is imported from or exported to the associated session[20].

6.4.6.1 Inherent Constraints

These constraints represent the implications of the modification of SRs in a session in the SSG of the associated (federated or composing) sessions. These constraints have been defined to keep consistency between SRs in federated or composed sessions. This is important to ease the voting procedure when it involves parties in more than one session.

Relationships between OSRs in different sessions: When an OSR is defined and it has its own[21] SM as controller and an imported SM as controlled, the following actions take place:

- If the own SM is exported, the OSR is also implicitly exported, i.e. the OSR will also appear in the SSG of the associated session;
- If the own SM is not exported, the OSR will appear in the SSG of the associated session, but the controller role will be transferred to the Peer representing the session which the non-exported SM belongs to.

Correlation between CompSRs:

- The deletion of a ShSR causes the deletion of the subsidiary SM in the session the Superior SM belongs to;
- The deletion of a SubSR causes the deletion of the subsidiary SM and the deletion of the complementary ShSR in the session the Subsidiary SM belongs to.

20. From now on, a SM will be considered ‘imported’ when it is participating in a SubSR as a Subsidiary; and ‘exported’ when it is acting as Subsidiary in a ShSR.
21. ‘Own’ is here used as non-imported.

6.4.7 Relationship between features sets and Session Graph Information Objects

The following table summarizes the relationship between the information objects described above and the Features Set of TINA session model as described in Section 7.2.6.

Table 6-1. Feature sets and relations to SSG information objects

Feature Set	SSG information object classes	Allowed operation
BasicFS	Party (1)	suspension deletion resume
BasicExtFS	Party (1)	suspension deletion resume
MultipartyFS	Party (1+)	creation suspension deletion modification
MultipartyIndFS (with indications)	Party (1+)	creation suspension deletion modification resume
ParticipantSBFS (Participant oriented stream binding)	SB session Relationship Stream Flow End Point Stream Interface Party (Resource)	creation suspension deletion modification resume
SFlowSBFS (Flow oriented stream binding)	SB session Relationship Stream Flow End Point Stream Interface SFC	creation suspension deletion modification resume
StreamInterfaceFS	Stream Flow End Point Stream Interface	creation deletion modification registration withdrawal
ControlRelation-shipFS	Control SR OwnershipSR PermissionSR (Write & Read) Party (1+) Peer	creation deletion modification

Table 6-1. Feature sets and relations to SSG information objects

Feature Set	SSG information object classes	Allowed operation
ResourceFS	Resource	creation suspension deletion modification resume
VotingFS	Party (1+)	creation suspension deletion modification
CompositionFS	Peer Resource Party Comp SR	creation suspension deletion modification resume

6.5 Communication Related Information Model

The communication session(s) and its relation(s) to service session(s) is described in the latest version of TINA NRA [8]. NRA section 4.1 gives an overview of both the service and the network architecture and how they fit together. Though the main focus is not on the service layer, it gives useful information to readers of this document. Furthermore, the NRA section 4.3.1 gives the information models related to mapping from stream bindings to stream flow connection (SFCs), and further, to Terminal Flow Connections (TFCs) and Network Flow Connections (NFCs). The same chapter also gives mapping from logical connection graph (LCG), speaking in terms of SFCs, to Nodal Connection Graph (NCG) and Physical Connection Graph (PCG), speaking in terms of TFC and NFC, respectively.

Figure 6-14 shows how SFC as point to multi-point connection uses the terms flow connection branch, root and leaf/leaves.

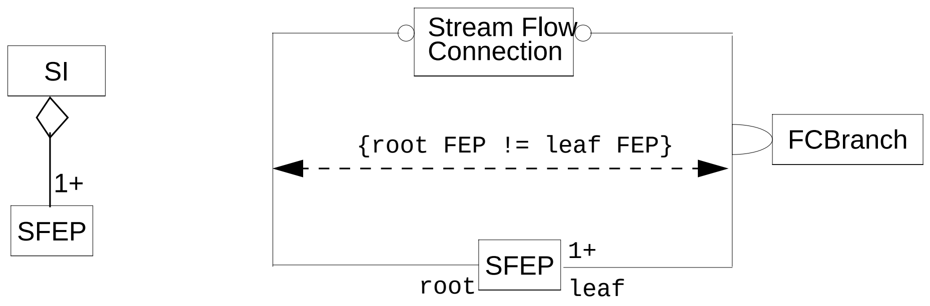


Figure 6-14. Stream flow connection information model

7 Computational Modeling Overview

This chapter defines the components of the service architecture, which are termed service components. These components are defined in the computational viewpoint. They map to computational objects or computational object groups, as defined in Computational Modeling Concepts (CMC) [5]. However, the concept of component is more general: it does not force any particular mapping to computational objects, leaving this open to designers to choose. This freedom is required to provide the desired flexibility to fulfill requirements stated in Chapter 1.

This chapter first defines what a service component is, then presents the service components as defined by TINA.

7.1 Service Components

This section defines the service component concept, and its relationship to computational objects and object groups of the TINA computational modeling concepts [5].

Service components are defined in the ODP computational viewpoint, and the definition of an individual service component is a computational specification. The TINA computational modelling concepts document defines computational objects (COs) and computational object groups (COGs) for computational specifications.

The service architecture defines a set of components which provide a framework for segmenting the functionality of TINA systems. Components can be used together to provide some functionality of the TINA system. The service components, which are defined in Section 7.2, "Overview of Service Architecture Components", are high level abstractions, which can be decomposed into COs and COGs. It is up to the designer of the TINA system to decide precisely how each service component is decomposed into COs and COGs. However, the architecture should not place any restrictions on how the service components are decomposed and deployed.

The problem is that TINA computational modeling concepts distinguish between COs and COGs.

COs are defined as a unit of distribution over a DPE node. If a service component were defined as a CO type, then all the functionality represented by the service component would have to be supported by a single DPE node. However, the architecture should not force a service component instance to be supported by a single DPE node.

COGs are not defined as a unit of distribution. However, COGs are only defined in terms of their internal COs and COGs. (A COG type is specified by listing the CO types that are part of the group.) COGs do not encapsulate (or hide) their internal structure. The architecture should not define the internal structure of any of the service components. (The internal structure of service components is decided by the system designers.)

So the service component concept is defined to allow a component to be decomposed into any combination of COs and COGs. Therefore, service components map to these entities: COs and COGs; interfaces of service components also map to interfaces of COs and contracts of CO groups.

In the future, the TINA CMC [5] will be updated to remove these restrictions, (i.e. COs can be defined at different levels of abstraction, and can be decomposed, and distributed over multiple nodes if necessary).

This decomposition is not visible as the service architecture is defined in terms of services components.

A TINA Service Component (SC) is an entity with the following properties:

1. It encapsulates data and functionality in the computational viewpoint;
2. It offers computational interfaces to other SCs and uses computational interfaces of other SCs;
3. It can map to the following:
    - a computational object,
    - a set of interacting computational objects,
    - a computational object group,
    - a set of interacting computational objects and computational object groups;
4. All CO mappings of service component type are equivalent, i.e. they can be seamlessly interchanged with no effect on the external world;
5. The service component is defined in ODL as a single computational object, with all its interfaces defined in IDL. This is a *CO representation* of the service component and is used to unambiguously define the interfaces of the component. (It does not mean that the service component is restricted to distribution on a single DPE node, as a CO is.)

There may be several CO mappings for an SC, thus enabling flexible design (e.g. in terms of distribution); in other words, there are several CO mappings for an SC. The representation defined in point 5 is chosen uniquely in TINA, thus enabling each component to be unambiguously specified in TINA-ODL language (defined in [6]).

Figure 7-1 shows 3 examples of different mapping for a service component S. All of these mappings are equivalent at the service component level and can be seamlessly interchanged. Each example offers the same external interfaces for the service component.

Each mapping may provide some benefit to the system designer, (e.g. improving system performance by allowing different deployments of COs, or allowing better re-use of CO code). Other mappings are also possible. None of the mappings can be distinguished at the service component level, (Unified View). Cases 1 to 3 represent possible *mappings* of service component S to structures of computational objects and object groups.

7.1.1 Using service components

The TINA service architecture provides a framework to offer a set of generic functionalities, in a stakeholder-independent, service-independent, and interoperable way. This is achieved by defining service components: a set of generic components and their interfaces. These components are defined in Section 7.2 and specified in the forthcoming Service Components Specifications document [14]. Reference points are also defined between some service components. These reference points define points of interoperability between domains and within a domain. Conformance to reference points is described in [3]. Reference points never occur within a service component, only between some of them.

TINA system designers will also want to provide additional functionality to that defined by the service architecture, in order to provide some competitive added value for their system This can be achieved by making it possible to derive new components from existing ones; this is done according to two paradigms:

a. specialization,
b. specialization and composition.

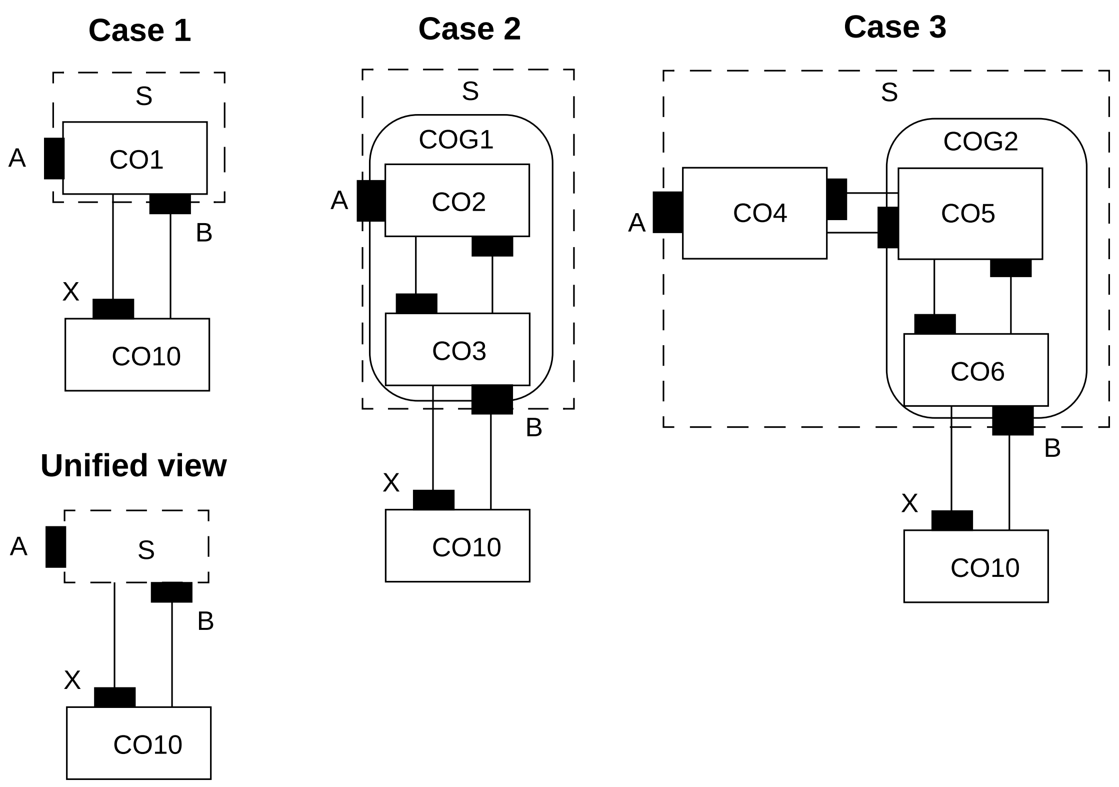


Figure 7-1. Service components and their relationship to computational objects and computational object groups

Specialization allows a service component to be extended through inheritance. A specialized component inherits all the functionality of the ‘parent’ service component, and can add some new functionality. However, the specialized component must still provide all the functionality of the parent component. This functionality must be assessable through the same service component interfaces. The specialized component must provide all of the interfaces provided by the parent component. These interfaces must either be subtypes of the interfaces on the parent component, or be of the same type. (The specialized component may also provide additional interfaces not supported by the parent component.)

Specialization ensures compatibility between specialized components, by ensuring that they can be treated exactly as if they were their parent components. To check that a component is a specialization of a parent component, the CO representation of the two components must be considered. If the CO representation of the specialized component is a specialization of the CO representation of the parent component, then the specialized component is definitely a specialization of the parent component. (For each interface supported by the CO representation of the parent component, either that interface type, or its subtype must be supported by the CO representation of the specialized component.)

Specialization can be used to design service components the provide the same functionality as TINA service components, but provide subtypes of the interfaces to support both TINA and provider specific functionality.

Composition allows a service component to be extended through aggregation. Paradigm b. above refers to “specialization and composition”, not “composition” alone, because a component cannot have functionality added and still be the same type of service component, it must be a subtype (or possibly a completely different type).

Aggregation allows COs and COGs to be added to a service component. When this adds new external interfaces to the service component, then the service component becomes a specialization of the parent service component, i.e., the new aggregated component has all of the interfaces of the parent component, plus some additional interfaces. It can be treated exactly as a parent component.

Composition through aggregation can be used to design SCs that contain generic components defined by TINA and additional specific components. The new SCs provide all the same functionality of the TINA components, and the same interfaces, but also provide additional interfaces to support provider specific functionality.

In summary, the same computational specification for a service component (i.e. the ODL specification of its CO representation) can correspond to several structures of COs and CO groups (for example, case 1 to 3 in Figure 7-1), potentially distributed over several DPE nodes. Such structures can be interchanged without any impact, at the computational level, on the clients and servers of the component in question. Furthermore, the particular structure to adopt can be left up to the designer without sacrificing accuracy of the specification.

7.2 Overview of Service Architecture Components

This section gives an overview of the computational model of the TINA service architecture in terms of TINA service components. More detailed specifications of the TINA service components will be given in a forthcoming version of the Service Component Specifications document [14].

To support the separation principles from Section 3.3, "Access, Service and Communication Separations", the service components are categorized as follows:

- **Access** related components, supporting interfaces which provide user's universal access to services;
- **Usage** related components, supporting interfaces which allow users to use (interact with) a service;
- **Communication** related components, supporting interfaces which control communication services (stream flow connections).

Table 7-1. Service Components in TINA Service Architecture

Category	Domain Role	TINA Service Component	Abbreviation
Access Session related	User Role	User Application	as-UAP
		Provider Agent	PA
	Provider Role	Initial Agent	IA
		User Agent	UA
		Named User Agent	namedUA
		Anonymous User Agent	anonUA
	Peer Role	Initial Agent	IA
		Peer Agent	PeerA
Service Session related	Party Role	User Application	ss-UAP
		Composer Usage Session Manager	CompUSM
	Provider Role	Service Factory	SF
		User Service Session Manager	USM
		Service Session Manager	SSM
	Peer Role	Service Factory	SF
		Peer Usage Session Manager	PeerUSM

Table 7-1. Service Components in TINA Service Architecture

Category	Domain Role	TINA Service Component	Abbreviation
Communication Session related	User Role[a]	Terminal Communication Session Manager	TCSM
	Provider Role[a]	Communication Session Manager	CSM

a. Other roles may occur here, as federation on the communication level is not covered in this version of the architecture.

Service components (SCs) are also separated according to the business administration domain's role for access and usage. Table 7-1 lists the TINA service components defined by the TINA service architecture. Definitions of these components can be found in Section 7.2, in particular Section 7.2.5, "Access Session Related Components" and Section 7.2.6, "Service Session Related Components". All the TINA service components have to be mapped onto computational objects and groups, as explained in Section 7.1. The way this mapping is achieved is not part of the prescriptive architecture. A suggested mapping for some components is given in Annex A-5, "Descriptive Refinements of Service Components".

7.2.1 Example of User-Provider Roles

Figure 7-2 gives an example of how some of these service components can interact, categorized as above. This figure shows a simple case of two users using a service in a provider domain. The following overview describes the service components in a scenario where the provider domain acts in both an access provider role and a usage provider role. (This scenario is typical of the Ret reference point[13]). Scenarios where domains take different roles in the access and usage parts are also possible, and are described later.

Access session related components provide a framework for offering secure and personalized access to services and for supporting mobility. The **Initial Agent (IA)** is the initial contact point for the **Provider Agent (PA)** wishing to interact with the provider, and is used to gain an access session with the **User Agent (UA)**.

The PA and UA components interact within a secure, trusted relationship between the user and the provider (an access session). They support authorization, authentication and customization of the user's service access and provide a secure mechanism for starting and joining sessions. In terms of the access session, the user domains take access user roles; the provider domain takes an access provider role.

The **access session related User APplication (as-UAP)** provides the user interface for the user to interact with the provider. It interacts with the PA to perform user requests, e.g. to establish an access session, and use services.

Service session related components provide a framework for defining services which can be accessed and managed across multiple domains. In the provider domain, **Service Session Managers (SSMs)** and **User Service Session Managers (USMs)** are instantiated by **Service Factories (SFs)** based on requests from UAs. An SSM and USM provide session control capabilities — an SSM supports those shared among the users, and a USM supports those dedicated to a user. The **service session related User Application (ss-UAP)** in the user domain allows a user to interact with a service session and acts as an end point for session control. In terms of the service session, the user domains each take a usage party role; the provider takes a usage provider role.

The communication session related components provide end-to-end connectivity. Figure 7-2 shows a **Communications Session Manager (CSM)**, using a **Terminal CSM (TCSM)** to establish a stream binding between two stream interfaces on the users' UAPs. Details on CSM and TCSM can be found in TINA NRA, and so can the details of connection related components.

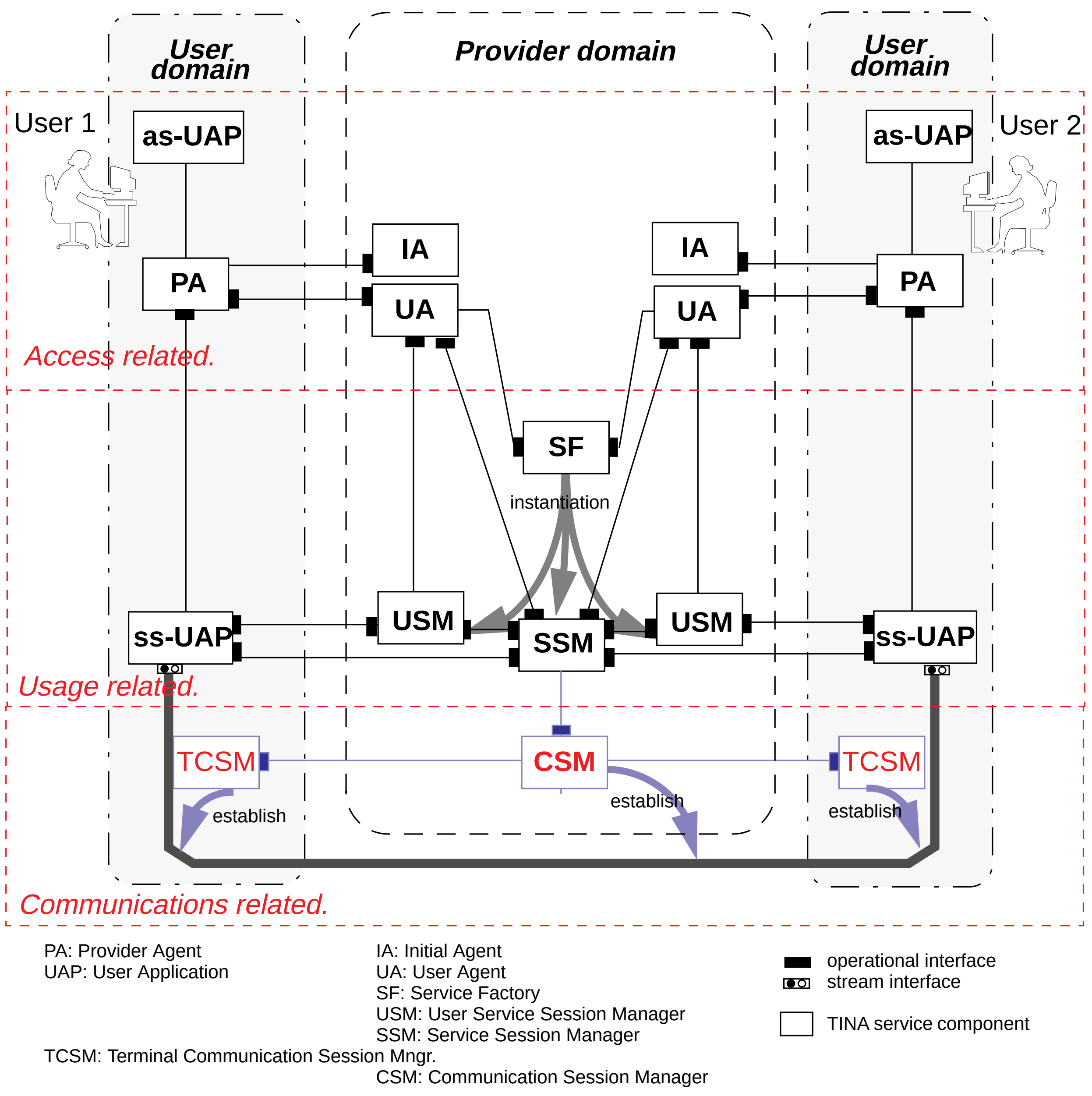


Figure 7-2. Example of User-Provider roles using service components

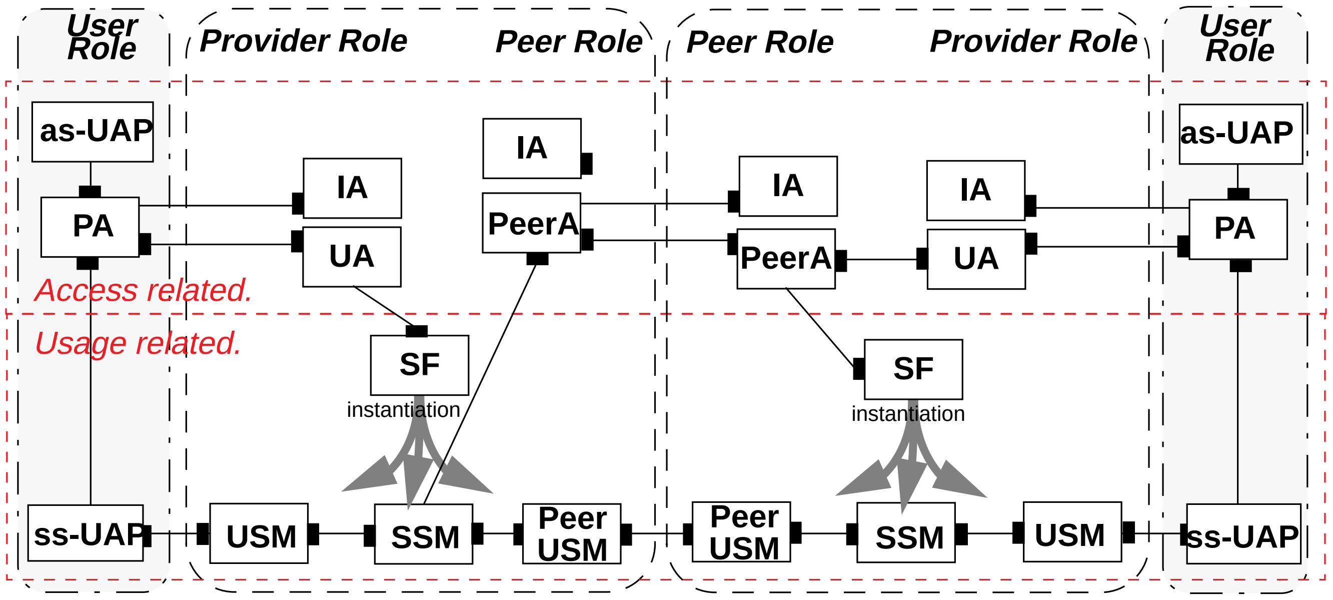
Other service specific components may be necessary in addition to those in Table 7-1. For example, for a video-conferencing service, it may be useful to model the video bridge as a service specific component, separated from the service session components. For example, the video bridge may have both operational interfaces (to control the bridge, i.e. the composition of video pictures or the sound level) and stream interfaces (e.g. as sinks of flows from the conference participants and sources of merged video flows). Such components are collectively termed Service Support Components (SSCs). The definition of individual components is outside of the scope of TINA.

Examples of the dynamics of the interactions between the components is given in Section 7.4, "Examples". More detail on the components' functionality and interfaces will be given in the forthcoming Service Components Specifications document. Some objects support more than one interface. Separate interfaces have not been shown in this diagram.

Communication session related components, in grey, are introduced briefly in Section 7.2.7, "Communication Session Related Components", and are defined in TINA NRA [8].

7.2.2 Example of Peer-to-Peer Roles

Figure 7-3 gives an example of peer-to-peer roles, and how these are related to service components.



PeerA: Peer Agent
PeerUSM: Peer Usage Session Manager

Figure 7-3. Example of peer-to-peer access roles, and peer-to-peer service sessions

This figure shows two domains co-operating to provide a service session between two users. It is typical of a conference service, where two of the participants are consumers of different retailers. The retailer domains both act in provider roles to the consumer domains, acting in user roles. (As in the previous example).

The retailers act in peer roles towards each other at the access level. Both domains are able to initiate an access session with the other domain, request the use of services, and forward invitations to consumers of the other domain. The Initial Agent is again used as the initial contact point for the other domain wishing to establish an access session. A **Peer Agent (PeerA)** of one domain wishing to interact with the other domain contacts the IA and uses it to gain an access session with the PeerA of the other domain.

The PeerAs interact within a secure, trusted relationship between the domains (an access session). A PeerA in one domain represents the other domain in this domain, and allows the other domain to start and join sessions, send invitations, etc.

In this example, the retailer domains are interacting to set-up a federated service session. The service sessions are the same as the previous example, except that a **Peer Usage Session Manager (PeerUSM)** is used by each of the domains to represent the other domain in the service session. The PeerUSM represents the consumer of the other domain to the local service session. The PeerUSMs interact in a peer usage role, to keep the other service session up-to-date with changes in the local service session.

This example showed 2 domains acting in peer roles at both the access and usage levels. Although domains can have the same role in access and usage, this is not mandatory.

7.2.3 Example of Composition

Figure 7-3 gives an example of using composition to combine 2 service sessions.

This composition example reuses the components from the previous examples and introduces one new component, the Composer Usage Session Manager (CompUSM).

Access session related components are used to establish a relation between the domains involved in a composition. These may either be PA and UA, used in the consumer example, or the PeerA used in the peer example. The components are determined by the reference point between the domains.

In this example, a service session in one domain initiates the composition by making a request to start (and use) to the access components. This causes a service factory in the second domain to instantiate a service, in a similar way to the user-provider example. When the initial request is confirmed, a service factory in the first domain instantiates a CompUSM, which allows the first service to use (i.e. act as a usage party) to the new service.

Note that the relation between the services could be reversed, with the new service adopting a usage party role to the requesting service. This would be reflected in the usage session components instantiated in each domain.

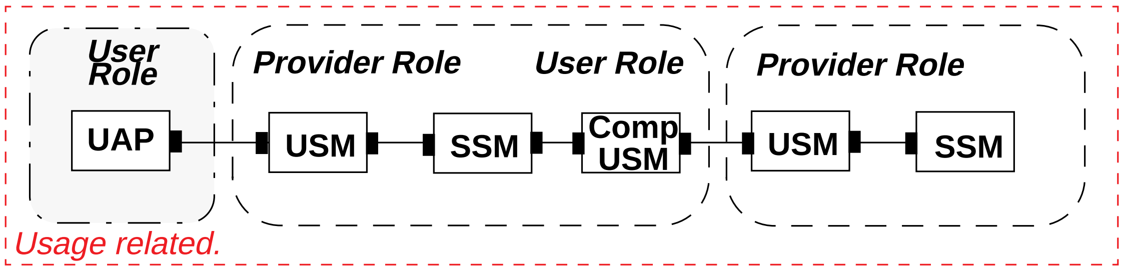


Figure 7-4. Example interactions for a compound service session

7.2.4 Service Components and Domains

As described in Section 3.4, "The Session Role Concept", domains can assume a number of access and usage roles. For access, a domain can support user, provider or peer roles, depending on reference point requirements. For instance, across the Ret reference point, the retailer supports the access provider role and the consumer supports the access user role. For example, for an end-user (consumer) wishing to use services provided by a retailer, the consumer domain takes a user role, and the retailer takes a provider role.

For usage, a domain can support usage party, usage provider, and usage peer roles. For example, for an end-user (consumer) using a library service provided by a retailer, the consumer domain takes a usage party role, and the retailer takes a usage provider role.

However, a domain can support a number of different usage roles across the same reference point, and these roles need not be the same as their corresponding access roles. For the Ret reference point, an access user is usually also a usage party (as in the examples above), but, for RtR, the access user may be a usage party, provider or peer.

Components are defined according to the roles that they take. The components are generic, so that they are applicable to whichever role the business administration domain takes. However, this may also require that components are specialized depending on the administrative domain in which they are used.

The domain encompasses the business's hardware and software systems. These systems may vary considerably depending on the administrative domain and the role they take. The service architecture does not specify details of the internal structure of the domain systems. Modeling of them is limited to identifying capabilities required to use services. However, it does assume that the TINA components interact through a DPE and make use of other TINA components and services, inside and outside the domain.

For example, a domain taking the access user role may be a single end-user, with the user's systems consisting of a single terminal with a direct physical connection to a connectivity provider. Or it may be a large site, encompassing a network of many terminals and other communications and computing resources, with one or more physical connections to one or more connectivity providers. The components defined are generic enough to be applicable in both situations.

7.2.5 Access Session Related Components

The access session related components support the access related sessions[1]. They support both the functionality and common session operations defined for the session concepts. A mapping between the session concepts and the service components is given in Section 7.3, "Relationship to Information Model".

Access sessions can be symmetric or asymmetric, as discussed in Section 6.2.2, "Access Session" The type of access session is determined by the reference point between the domains. A symmetric session requires components in different domains to support user and provider roles, while an asymmetric access session requires both domains' components to support peer roles. This section starts by considering the components necessary to support an asymmetric access session. The user role is supported by the Provider Agent, while the provider role is supported by the Initial Agent and User Agent. A UAP is also considered, which support end-user needs.

The symmetric access session is supported by Peer Agents in each domain acting in the access peer role. Peer domains also support Initial Agents to support initial contact from another domain, before an access session is established.

1. Access Session (AS), Provider Domain Access Session (PD_AS), and User Domain Access Session (UD_AS).

7.2.5.1 User Application

The User APplication (UAP) SC is defined to model a variety of applications and programs in the domain. A UAP SC represents one or more of these applications and programs. A UAP can be used by human users, and/or other applications in the user domain. A UAP can be either or both an access session related and service session related SC. The access session related UAP is defined below. The service session related UAP is defined in Section 7.2.6, "Service Session Related Components".

As an access session related SC, the UAP enables a human user, or another application, to make use of the capabilities of a PA or PeerA, through an appropriate (user) interface. An access session related UAP supports part of the domain access session. The UAP provides capabilities for:

- request authentication information from the user, required by the PA (or PeerA) to set-up an access session with a UA (PeerA),
- the user to request the creation of new service sessions,
- the user to request to join an existing service session,
- alerting the user to invitations, which arrive at the PA or PeerA.

An access session related UAP may also support the following optional capabilities, when they are also supported by the PA or PeerA:

- allow the user to search for a provider, and register as a user of the provider’s services;
- allow the user to search for services and identify providers providing those services.

Zero or more stream interfaces [5] can be attached to a UAP. The stream interfaces can be bound to those in user systems and/or those in providers’ domains.

A user or peer domain contains one or more access session related UAPs. Any access session related UAP can request a PA or PeerA to establish an access session. One or more UAPs interact with a PA or PeerA to use its access session related capabilities within an access session.

A UAP instance may support only access session related capabilities or only service session related capabilities; or it may support both. Access session related UAPs may be specialized by a domain to interact with a specialized PA or PeerA.

7.2.5.2 Provider Agent

The Provider Agent (PA) is a service independent SC, defined as the user’s end-point of an access session. The PA is supported in a domain, acting in an access user role. The PA supports a user accessing their UA and making use of services, through an access session. The PA supports the user domain access session, in conjunction with access session related UAPs, and other user domain infrastructure.

Capabilities supported by a PA:

- set-up a trusted relationship between the user and the provider (an access session), by interacting with an Initial Agent[2], and gaining a reference to a UA[3];
- within an access session:
    - convey requests (from a user to a UA) for creating new service sessions,
    - convey requests for joining existing service session,
    - receive invitations to join existing service sessions (from a UA) and alert the user[4],
    - anonymously make use of a provider’s services,

- deploy new components into the user's domain,
- support access to terminal configuration information from a provider's domain,
- register to receive invitations sent when no access session exists.

Operations supported by a PA are service independent.

For each concurrent access session a user has with a provider, there is one PA instance in the user's domain. Each PA may be associated (through an access session) with the same UA, or separate UA instances. A single PA is only ever associated with one UA through an access session. (When no access sessions exist, a user domain can still support a PA. It can be used to initiate an access session, and may receive invitations if registered.)

7.2.5.3 Initial Agent

An Initial Agent (IA) is a user and service independent SC that is the initial access point to a domain. An IA is supported by domains taking both the provider and peer roles. An IA reference is returned to a PA or PeerA when it wishes to contact the domain. The IA supports capabilities[5] to:

- authenticate the requesting domain and set up a trusted relationship between the domains (an access session) by interacting with the PA or PeerA,
- establish an access session, but allowing the requesting domain to remain anonymous. The type of UA accessed in this way is an anonymous user agent.

Operations supported by an IA are service independent.

An IA supports requests from one PA / PeerA at a time. The PA / PeerA requests to contact the domain and is given a reference to an IA. When the PA / PeerA has interacted with the IA to establish an access session with a UA / PeerA, the reference to the IA will become invalid[6]. Subsequently, the IA may be contacted by another PA / PeerA.

7.2.5.4 User Agent

A User Agent (UA) is a service-independent SC, that represents a user in the provider's domain. It is supported by a domain acting in the access provider role. It is the provider domain's end-point of an access session with a user. It supports the provider domain access session. It is accessible from the user's domain, regardless of the domain's location.

A UA supports capabilities to:

2. The PA may use a location service to find an interface reference for the IA, or some other means. The PA will provide the retailer name, and possibly other information to scope the search of the location service. The capability of the location service to return this interface reference is an important part in enabling access irrespective of location, which is one important feature of personal mobility. This assumes that the location service can indeed be contacted, irrespective of location. Also, it may imply that interworking between location services in different domains is required. How the location service gains an interface reference of an initial agent is undefined. It is likely that the location service has to interact with an object in the provider's domain in order to gain the reference. This interaction is not defined at present.
3. This capability is an important element in support of personal mobility, as it allows a user to access a provider domain from various locations.
4. Using an access session related UAP.
5. These capabilities are available irrespective of the location of the PA with which the IA interacts, and therefore are an important part of personal mobility support, i.e. allowing the user access irrespective of location. For a discussion of invitations, see Section 7.4.4 and Section 8.2.3.5.
6. That is, the PA should not retain a reference to an IA after it has established an access session. An implementation may enforce that the reference to the IA is not usable once the access session has been established.

- support a trusted relationship between the user and the provider (an access session) by referencing the user's PA,
- within an access session:
  - act as a user's single contact point to control and manage (create/suspend/resume) the life-cycle of service sessions and user service sessions,
  - create a new service session (by requesting that a service factory creates a USM and SSM),
  - join in an existing service session by creating a new user service session (via a service factory creating a USM),
  - resolve the service execution environment of the user, allowing them to use services from many different types of terminals. This requires resource configuration information of the user system (which includes terminals and their access points being used by or available for the user). Access to this information may be restricted by the user/PA,
  - provide access to a user's contract information with the provider,
  - resolve interaction problems between service usage requests.

The UA is defined as an abstract (i.e. non-instantiable) component type. Two instantiable subtypes are defined:

- Named User Agent (namedUA);
- Anonymous User Agent (anonUA).

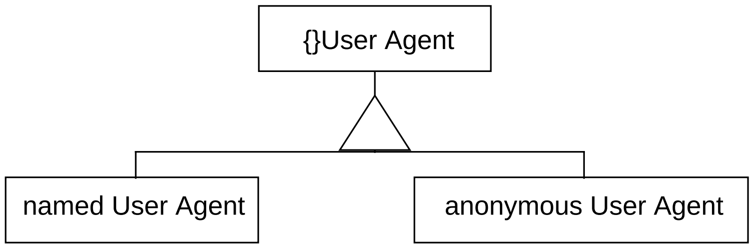


Figure 7-5. Inheritance hierarchy for User Agent.

Instances of the subtypes are created to represent different types of users. The subtypes of UA support all the capabilities which are defined for UA.

The namedUA is a UA, specialized for a user that is an end-user or subscriber of the provider.

The anonUA is a UA, specialized for a user that does not wish to disclose their identity to the provider.

Definitions for these subtypes are given in the sections below.

7.2.5.5 Named User Agent

A Named User Agent (namedUA) is a service independent SC, that represents a user in the provider's domain. The namedUA is a specialization of UA for a user that is an end-user or subscriber of the provider. It is the provider domain's end-point of an access session with a user. It is accessible from the user's domain, regardless of the domain's location.

The namedUA supports all of the capabilities which are defined for UA. In addition, it supports the following capabilities:

- within an access session:
  - Act as a single contact point to control and manage (create/suspend/resume) the life-cycle of service sessions and user service sessions, taking into account restrictions posed by subscribers and the user;
  - Suspend/resume existing user service sessions and service sessions. This includes support for session mobility;
  - Manage the user's preferences (choices or constraints) on service access and service execution (This is supported by starting a provider specific service session.);
  - Resolve the service execution environment for the user, allowing them to use services from many different types of terminals. This requires resource configuration information of the user system, (which includes terminals and their access points being used by or available for the user. Access to this information may be restricted by the user/PA.) This includes support for personal mobility;
  - Register user at a terminal to receive invitations. This includes support for personal mobility;
  - Allow the user to define user private/public policies. (This is supported by starting a provider specific service session.);
  - Negotiate the session models and feature sets supported by a service session, in order for it to interact with a UAP in the user's domain.
- accept invitations from users to join a service session;
- deliver invitations to a terminal, previously registered by the user with the namedUA. No access session would be required to allow this delivery of invitations.

The namedUA may support the following optional capabilities:

- perform actions on the users behalf, when the user is not in an access session with the namedUA;
- initiate an access session with a PA;
- support additional authentication of the user. This may be tailored to the user, and usage context.

Operations supported by a namedUA are service independent.

A namedUA may support one or more access sessions concurrently[7]. Each access session is with a single, distinct PA.

7.2.5.6 Anonymous User Agent

An Anonymous User Agent (anonUA) is a service independent SC that represents a user in the provider's domain. The anonUA is a specialization of UA for users that do not wish to disclose their identity to the provider. It is the provider domain's end-point of an access session with the anonymous user.

The anonUA supports all of the capabilities which are defined for UA. In addition, it supports the following capabilities:

7. NamedUAs must be able to support one access session, and may optionally be able to multiple concurrent access sessions. NamedUAs continue to exist when there is no access session.

- Support a trusted relationship between the user and the provider (an access session) by referencing the user's PA. The provider does not know the identity of the user. ('Trust' is not guaranteed by identifying the user, as for the namedUA, but may be ensured by, for example, pre-payment.);
- within an access session:
  - Suspend/resume existing user service sessions and service sessions within an access session. (Suspended sessions cannot be resumed in a different access session.[8]);
  - Manage the user's preferences (choices or constraints) on service access and service execution. (These would have to be determined during the access session, and could not be re-used in a separate access session.);
  - Provide access to a user's contract information with the provider. (This contract would be defined at the start of the access session and terminated at the end of the access session.);
  - Define user private /public policies. (This may be supported by starting a provider specific service session. This information would only be maintained during this access session.);
  - Allow the anonymous user to register as a user of the provider (i.e., set-up a contract with the provider for longer than a single access session);
  - Negotiate the session models and feature sets supported by a service session, in order for it to interact with a UAP in the user's domain.

The anonUA provides no support for personal or session mobility.

7.2.5.7 Peer Agent

The Peer Agent (PeerA) is a service independent SC that represents a peer in another peers domain. It is supported by a domain acting in the access peer role. It is this domain's end-point of an access session with the peer domain. It supports a peer domain access session. It also represents another domain, or a member of another domain, to this domain, and holds the agreed contract between the domains. It is accessible from the peer's domain, regardless of that domain's location.

The PeerA should support the combined external capabilities of a UA and a PA. It may provide other functionality to ensure the initiator of the access session and responder both receive appropriate references to each other. Also, it ensures the coordination of these external capabilities and maintains consistency of requests and responses.

7.2.6 Service Session Related Components

The service session related components defined in this section follow the TINA session concepts from Section 6.3. Service sessions can be supported over multiple domains. The service related sessions[9] are supported by these components.

The interactions between the domains depend on the role the domain takes in the service session. A domain acting in a usage provider role will support a Service Session Manager. It will also support User Service Session Managers (USMs) for each domain acting in a usage party role. The party domain supports a service session related UAP to interact with the USM.

8. It is assumed suspended sessions are ended by the provider if the access session is ended.
9. Service Session (SS), Provider Service Session (PSS), Usage Service Session (USS), Provider Domain Usage Service Session (PD_USS), User Domain Usage Service Session (UD_USS), Peer Domain Usage Service Session (PeerD_USS), and Composer Domain Usage Service Session (CompD_USS).

A domain may be perceived as a party domain by the provider domain, while it is actually composing this service session with its own; it provides a Composer Usage Session Manager to interact with the USM.

Domains, acting in a usage peer role with each other, provide a Peer Usage Session Manager (PeerUSM) to interact with the other domain.

Session models define how service session components in each domain can interact in a generic manner. These session models allow components, which have been designed and implemented separately, to interact to support the service session.

TINA defines a single session model, the TINA Session Model. This model allows service session components to make requests about: ending and suspending the session, the parties involved, set up and modification of stream bindings between parties, for example.

Service session related components may support one or more of a variety of session models. These session models may be defined by a variety of organizations. Each session may support a number of session models, or may only support a single model. Services may decide not to support the TINA Session Model. This is acceptable, following the statement in [19], that “the access part includes the possibility to negotiate alternative usage interfaces”.

7.2.6.1 The TINA Session Model

The TINA session model defines a number of feature sets and interfaces which can be used to interact with a session. The session components may also support other service specific interfaces, (and probably will) and may support other session models. The architecture assumes that the session models act independently, and only describes TINA session model interactions.

The TINA session model defines a basic feature set which must be supported by all sessions which support the TINA session model. The basic feature set allows a client application in the user’s domain to do the following: to discover the interfaces, session models and feature sets supported by a session; to retrieve the interfaces supported by the session (including those specified by supporting a particular feature set); to register the client’s own interfaces and session models with the session; and to end and suspend the session.

It also defines additional feature sets which may also be supported by the session. The feature sets which a session supports are available through the access part of Ret RP, and through operations on the interface defined by the mandatory basic feature set.

7.2.6.2 TINA Session Model Feature Sets

The table below provides a brief description of each of the feature sets, and identifies any feature sets that it is dependent on.

Roles may determine which feature sets are offered and which interfaces in a feature set that a component supports or requires between domains. A session in a usage provider role offers usage provider type interfaces, and requires usage party interfaces from the other domain session component.

Current (May ‘97) feature sets are defined for the usage party and provider roles only. Feature sets for peer roles have not yet been defined. Other more specialized roles may have specific feature sets to support those roles, e.g. a domain in a manager role may require a management feature set (not defined).

Table 7-2. TINA Session Model Feature Sets

Feature Set	Description	Dependent on
BasicFS	Support end and suspend session requests. Allows the users' UAPs to discover interfaces and session models supported by the session, and register their interfaces with the session.	Mandatory for TINASessionModel
BasicExtFS	Allows the provider domain session components to discover interfaces and session models supported by the user domain UAPs.	BasicFS
MultipartyFS	Allows the session to support multiparty services. Supports requests for: - information on other parties - ending/suspending a party in the session - inviting a user to join session - announcing the session.	BasicFS
MultipartyIndFS	Allows the session to indicate requests that are to be processed to the user domain UAPs.	MultipartyFS
VotingFS	Supports UAPs voting to determine if a request should be accepted, and executed.	MultipartyIndFS
ControlRelationshipFS	Supports parties having ownership, and read/write rights on session entities, (i.e. parties, resources, streams, etc.)	BasicFS
CompositionFS	Supports the composition and federation of service sessions between domains.	ControlRelationshipFS
ResourcesFS	Supports requests and use of resources needed for the provision of a service.	BasicFS
MgmtCtxtFS	Supports the manipulation of management contexts. (See Section 5 for details on management contexts)	BasicFS
ParticipantSBFS	Participant type stream binding feature set: provides high level support for setting up stream bindings. Stream bindings are described in terms of session members' participation.	BasicFS
ParticipantSBIndFS	Participant type stream bindings with indications.	ParticipantSBFS
StreamInterfaceFS	Stream interface feature sets: allows the manipulation and passing of Stream Interface (SI) and Stream Flow Endpoint (SFEP) information.	BasicFS
SFlowSBFS	Stream flow type stream binding feature set: provides a detailed support for setting up stream bindings. Stream bindings are described as an aggregation of Stream Flow Connections (SFCs).	StreamInterfaceFS
SFlowSBIndFS	Stream flow type stream bindings with indications.	SFlowSBFS
SimpleSBFS	Simple stream binding feature set: a SFC defines a stream binding.	StreamInterfaceFS
SimpleSBIndFS	Simple stream binding with indications.	SimpleSBFS

The session must also support all the feature sets that the other feature set is dependent on. (E.g. MultipartyIndFS is dependent on MultipartyFS. If a session were to support MultipartyIndFS, then it must also support MultipartyFS. And since MultipartyFS is dependent on BasicFS, the session would also have to support BasicFS.)

7.2.6.3 User Application

The User APplication (UAP) SC is defined to model a variety of applications and programs in the user domain. A UAP SC represents one or more of these applications and programs in the TINA computational model. A UAP can be used by human users, and/or other applications in the user domain. A descriptive CO mapping of the UAP is given in Annex A-5, "Descriptive Refinements of Service Components". A UAP can be either or both an access session related and service session related SC. The access session related UAP is defined in Section 7.2.5.1, "User Application", under “Access Session Related Components”. The service session related UAP is defined below.

As a service session related SC, the UAP enables a user to make use of the capabilities of a service session, through an appropriate user interface. It acts as an end-point of a service session, by supporting the User Domain Usage Service Session (UD_USS). The capabilities provided by particular UAPs are specific to the UAP, and any service session it is part of. UAPs may provide some of the following generic capabilities to the user[10], such as:

- starting / ending the session,
- inviting other users to join the session,
- joining an existing service session,
- adding / removing / modifying stream bindings and the users’ participation in them;
- establishment of control session relationships and other changes in the service session
- suspending the user’s participation in the session, or the whole session,
- resuming the user’s participation in the session, or the whole session.

Zero[11] or more stream interfaces [5] can be attached to a UAP. The stream interfaces can be bound to those in other user systems and/or those in the provider domain.

The user’ s domain contains one or more service session related UAPs. One service session related UAP can be involved in one or more service sessions. For each service session the UAP is involved in, it interacts with a user service session manager, or a service session manager[12].

The UAP may also support a particular session model, such as the TINA Session Model, and Feature Sets and interfaces associated with this. The USM/SSM uses these interfaces to share information with the UAP on parties and resources in the service session. If a UAP is interacting with more than one USM, there may be one set of feature set interfaces supported for each USM[13], or all the USMs may be supported by the same set of feature set interfaces.

10. Which of the capabilities that are actually offered are determined by the feature sets supported.
11. Some services don’t require stream interfaces on the UAP.
12. Services which only support a single user in a service session can provide only an SSM (no USM), and allow the UAP to interact directly with the SSM. These services will be restricted to only ever having a single user in a service session, and should not be able to invite other users to join the session, as no USM for the invited user could be made available.
13. These USMs belong to separate service sessions.

A UAP instance may support: only access session related capabilities; only service session related capabilities; or it may support both. Service session related UAPs will be specialized to the service session(s) they interact with. In order to use a service session, a UAP specialized to the service type must be present in the user’ s domain (or be deployed, e.g., by downloading) before the session is used.

7.2.6.4 Service Factory

A Service Factory (SF) is a service-specific SC, which creates the service session components for a service type.

A request to create a service session of a particular service type will result in the creation of one or more SC instances[14]. The SF will create and initialize the instances according to rules imposed by their implementation. The SF will return to the client one or more interface references to these components. (The SF is used to create instances of all the service session related components defined in this document: USM, SSM, CompUSM and PeerUSM.)

Requests are typically made by UAs or PeerAs. Other clients may also be able to request the creation of a service session. The client must have an interface reference to the SF and issue an appropriate request. A SF which supports more than one service type would typically provide separate interfaces for each service type.

A SF supports capabilities to:

- create SCs for one or more service types upon request. (This includes choosing the session models supported by the service session, although this may be fixed by the service type.)

SF may support optional capabilities to:

- create a SC (typically USM, or CompUSM) to be used in conjunction with other SCs (typically a SSM & USMs) created by a different factory instance;
- continue to manage the created SCs. It may provide a list of sessions managed by it, and may ‘clean up’ some sessions if requested;
- may include mechanisms to schedule the activation of a session at a specific date and time. (This mechanism includes resource reservation.);
- support suspension/resumption of a service session.

The SF assembles the resources necessary for the existence of a component it creates. Therefore, the SF represents a scope of resource allocation, which is the set of resources available to the SF. A SF may support an interface that enables clients to constrain the scope.

7.2.6.5 Service Session Manager

The Service Session Manager (SSM) is a service component which comprises the service-specific and generic session control segments of the Provider Service Session (PSS). An SSM supports service capabilities that are shared among members (parties, resources, etc.) in a service session. Information related to a particular member of the service session are encapsulated in Member Usage Service Session Managers (MUSMs). SSMs support (some or all of) the following capabilities:

14. Typically, the USM and SSM.

- keep track and control the various resources shared by multiple users in a service session. This can be done just by having references to other objects (like a CSM) which really maintain the context of usage for a specific kind of resources;
- hold the state of the service session and support suspension/resumption of the service session;
- support adding / inviting / removing users to/from the service session by interacting with the corresponding UAs;
- support adding / removing / modifying stream bindings and the users' participation in them;
- support the negotiating capabilities among the users interacting with the USMs. SSM will serve as a control center of consensus building (such as voting procedures);
- support management capabilities associated with the service session (e.g., accounting).

Zero[15] or more stream interfaces [5] can be attached to an SSM. The stream interfaces can be bound to other stream interfaces in this or other domains.

An SSM is created by a SF, one per request for the corresponding service type. It is deleted when all the users leave the service session, or when quit by a user or SF. The lifespan of an SSM is the same as the corresponding provider service session.

7.2.6.6 Member Usage Service Session Manager

The Member Usage Service Session Manager (MUSM) is an abstract service component, which comprises the service-specific and generic session control segments of the Domain Usage Service Session (D_USS) that interact with the PSS. The generic segments of the MUSM correspond to the TINA session model feature sets. It is specialized according to the role of session member supported by the D_USS:

- User Service Session Manager (USM) represents the UD_USS;
- Composer Usage Session Manager (CompUSM) represents the CompD_USS;
- Peer Usage Session Manager (PeerUSM) represents the PeerD_USS.

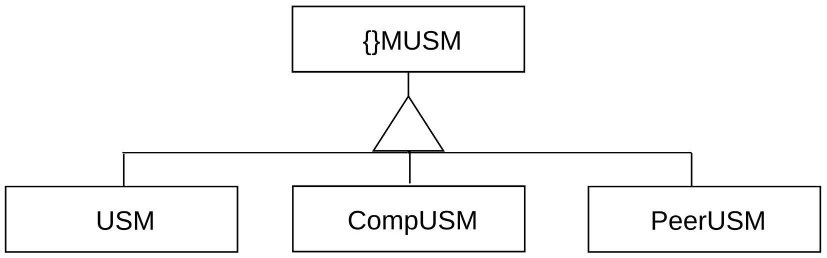


Figure 7-6. Inheritance hierarchy for Member Usage Service Session Manager.

The MUSM represents and holds the context of a member (party, resource, provider or peer) in a service session. Its main characteristics are the following:

- It contains the information and service capabilities which are local to the member. If an operation involves activities that are purely local to the member, the MUSM controls and manages the activities by itself. If not, the MUSM interacts with the SSM to perform the

15. Typically stream interfaces are offered by Service Support Components (SSCs) associated with an SSM.

operation. The SSM may interact with the MUSM in response to operations from other members (or due to service logic) that affect this member. Such interactions are dependent on the role of the member and the feature sets supported;

- It keeps track of and controls the exclusive (non-shared) resources used by the member in a service session. This can be done just by having references to other objects (e.g. a communications session manager) which really maintain the context of a member for a specific kind of resource;
- It may be configured to preferences of the member. This may be done during initialization, and dynamically during the session.

As the MUSM is an abstract service component, no instances are created. Instances of the appropriate specialized service component are created to represent specific session members.

7.2.6.7 User Service Session Manager

The User Service Session Manager (USM) is a service component which comprises the service-specific and generic session control segments of the Provider Domain User Service Session (PD_USS). It is a specialization of the MUSM which represents and holds the context of a party[16], or resource in a service session. It has the same characteristics as the MUSM (with member replaced by party or resource as applicable):

- It holds the state of the PD_USS and supports suspension/resumption of the party's participation in the service session;
- It supports the different roles of the party in the service. The role of a party will be service dependent (e.g., chairman in a conference).

Zero[17] or more stream interfaces can be attached to a USM. The stream interfaces can be bound to other stream interfaces in this or other domains.

A USM is created by the SF, one per request for the corresponding service type (per PD_USS). It is deleted when the party leaves the service session. The lifespan of a USM is the same as the corresponding PD_USS.

7.2.6.8 Composer Usage Session Manager

The Composer Usage Session Manager (CompUSM) is a service component, which supports composition of service sessions. Composition is asymmetric with one domain taking the usage party role, and the other domain taking the usage provider role. It is a specialization of the MUSM and supports the Composer Domain Usage Service Session (CompD_USS).

The CompUSM allows a service or resource to act as a usage party in a service session in another domain. It supports usage party interfaces to the service session in the other domain (i.e., to the other service session, the CompUSM appears as if it were a UAP).

7.2.6.9 Peer Usage Session Manager

The Peer Usage Session Manager (PeerUSM) is a service component which supports peer to peer relationships between service sessions in different domains. Federation is symmetric with both domains taking usage peer roles. It is a specialization of the MUSM and supports the Peer Domain Usage Service Session (PeerD_USS).

16. A party can be either an end-user, or a service session acting in a usage party role (see CompUSM).
17. Typically stream interfaces are offered by Service Support Components (SSCs) associated with a USM.

The PeerUSM allows a service session to interact with another service session in another domain. Both service sessions are peers, and both support a PeerUSM to interact through.

7.2.7 Communication Session Related Components

Communication Session related components are outside the scope of this document. The descriptions below are given for clarification only and are defined by TINA NRA [8]. Note that services not using stream bindings will not have communication sessions and will not need these components. However, as the components are service independent, it is likely that a CSM or TCSM will be present in most domains.

7.2.7.1 Communication Session Manager

The Communication Session Manager (CSM) is a service-independent SC which manages application-level, end-to-end bindings between stream interfaces (stream flow connections). A stream flow is an abstraction of a connection. The characteristics and configuration of stream flows are described by Logical Connection Graphs (LCGs). The CSM provides LCG interfaces to allow USM/SSMs to set-up, modify, and remove stream flows. The CSM decomposes the requested connection into two parts, the nodal part and the transport part. It requests TCSM to take care of the nodal part and requests other connection management objects to take care of the transport part.

The CSM provides the following capabilities:

- creation and control of stream flow connections(SFCs), end to end.

7.2.7.2 Terminal Communication Session Manager

The Terminal Communication Session Manager (TCSM) is a service-independent SC, which manages Terminal Flow Connections (TFCs) (intra-nodal flow connections) within the user's domain. The TCSM provides an interface to the CSM, to allow the CSM to request the TCSM to set-up, modify, and remove connections in the user's domain.

The TCSM provides the following capabilities:

- creation and control of flow connections (TFCs) within the user domain.

7.3 Relationship to Information Model

Table 7-3 and Table 7-4 provide a mapping between session concepts and objects in the information model, and service components in the computational model.

Session concepts, which are mapped to a service component, means that the service component supports the functionality and state of the session, and controls the resources which are part of the session. If a session concept is mapped to several components, then each of the components support part of the functionality and state, and control some of the resources of the session.

Information objects which map to a component mean that the information in the information object is contained within the component, and that the component may provide access to that information to other components/objects.

Table 7-3. Mapping between access session related components

Session Concept / Information Objects	TINA Service Components
Access Session (AS) with User-Provider Roles	PA and UA
Access Session (AS) with Peer-to-Peer Roles	PeerA and PeerA
User Domain Access Session (UD_AS)	PA
Provider Domain Access Session (PD_AS)	UA
Peer Domain Access Session (PeerD_AS)	PeerA
User Profile with User-Provider Roles	UA
User Profile with Peer-to-Peer Roles	PeerA
Contract with User-Provider Roles	PA and UA
Contract with Peer-to-Peer Roles	PeerA and PeerA

Table 7-4. Mapping between service session related components

Session Concept / Information Objects	TINA Service Components
Service Session (SS)	Service session related UAP, USM, and SSM
User Service Session (USS)	Service session related UAP, and USM
User Domain Usage Service Session (UD_USS)	Service session related UAP
Provider Domain Usage Service Session (PD_USS)	USM
Provider Service Session (PSS)	SSM
Composer Domain User Service Session (CompD_USS)	CompUSM
Peer Domain Usage Service Session (PeerD_USS	PeerUSM
Service Session Graph (SSG) model IOs	Service session related UAP, USM, SSM, CompUSM and PeerUSM

The Service Session Graph (SSG) objects may be supported by many of the service session related components. The SSG objects supported by each component depend on the feature set supported by the component. Table 6-1 describes the information objects which are associated with each feature set. In general this means that in a service session supporting a feature set, all components in the session support the information objects associated with that feature set.

7.4 Examples

Example scenarios are described in the following sections to illustrate how computational objects can interact in the following cases:

- Contacting a provider,
- Logging in to a provider as a known user,
- Starting a new service session,
- Inviting a user to join an existing service session,
- Joining an existing service session,
- Creating a stream binding in an existing service session.
- Composition: a service starts another service and acts in the usage party role
- Service federation: a new user joins a service session

Note that the scenarios are examples and that they assume all the operations are successfully completed (no error, no fault, and no rejection) for simplicity. Detailed explanations of those scenarios can be found in [14].

7.4.1 Contact a Provider

This example shows the user A making contact with a provider. This scenario supports user mobility by allowing the user to contact a specific provider from any terminal.

Preconditions:

A PA must be present in the user's domain.

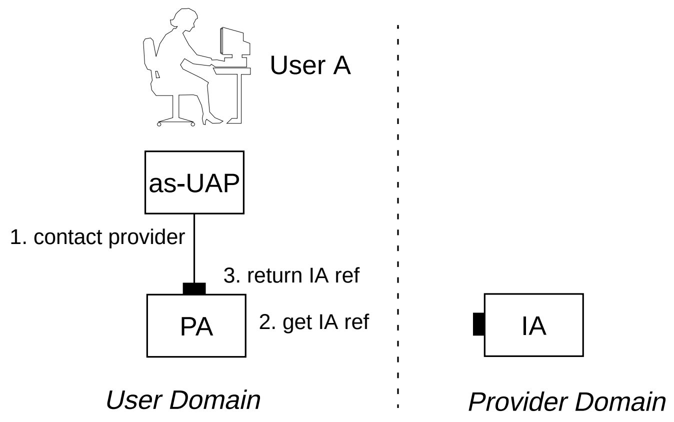


Figure 7-7. Contacting a provider

1. User starts an access session related UAP. He provides the retailer name he wishes to contact. UAP requests the PA to contact the provider, giving the retailer name.
2. PA gains a reference to an interface of an initial agent of the provider[18].
3. PA returns success to UAP.

18. The PA may use a location service to find an interface reference for the IA, or some other means.

Post-conditions:

The PA has an interface reference to the IA. The user has not setup an access session between the PA and IA. The IA does not support use of a user's services, only operations to set up an access session as a known user (See Section 7.4.2) or an anonymous user.

The IA has no knowledge of any interfaces on the PA or in the user's domain.

It is possible for the provider to download a provider specific PA to the user's domain, once an interface reference to the InitialAgent has been gained by the PA. This helps to support user mobility. No scenario describing how this is achieved is defined at present.

7.4.2 Login to a Provider as a Known User

This example shows the user A establishing an access session with their named user agent of the provider. The user wishes to make use of the provider's services which the user has previously subscribed to (See Section 7.4.3). This scenario supports user mobility by allowing the user to establish an access session with a provider from any terminal.

Preconditions:

The user has contacted the provider (as in Section 7.4.1), and the PA has an interface reference to an initial agent of the provider.

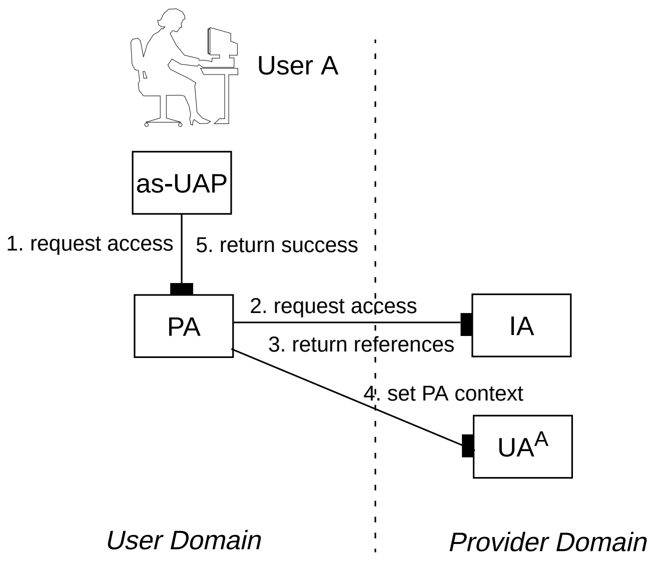


Figure 7-8. Login to a provider as a known user

1. User A uses an access session related UAP to login to the provider, as a known user[19]. The user has then requests the PA to login to the provider, as a known user. The UAP supplies the security information to the PA.
2. PA requests that an access session is set up with the namedUA of the user. PA provides the username of the user to the IA. The PA has passed the user security information to the security services supported by the DPE. The security services interact with the provider domain in order to authenticate the user[20].

19. UAP may ask the user for their username and other security related information, e.g., password.

20. If security services were not supported by the DPE, then the PA would first have to send authentication information to the IA.

3. IA has already authenticated the user through the DPE security services, and an access session has been established. It returns the interface reference of the user's UA.
4. PA sends information about the user domain to the UA. This information is termed the PA context, which includes references to interfaces on the PA, and possibly terminal information.
5. PA returns success to UAP.

Post-conditions:

User has setup an access session between the PA and namedUA. The namedUA is personalized to the user, and has knowledge of interfaces of the PA.

Any interface references of the IA held by the PA will be invalid.

Note that in the sequence of events, personal mobility is allowed due to the following capabilities:

a. The capability of the naming service to be contacted irrespective of location and to return a reference to the IA;
b. The IA to establish a trusted relationship with the user, that is independent of the physical location of that user;
c. The IA to return a reference of the user's own named UA to the (PA acting on behalf of the) user.

It is possible that once an access session has been established, the provider may download a provider specific PA to the user's domain. This helps to support user mobility. No scenario describing how this would be achieved is defined at present.

7.4.3 Starting a new Service Session

This example shows a user starting a new service session (in this example, a video conference service, but the interactions would be the same for all types of service). The user is assumed to be in an access session with the provider and to have a valid subscription to the service (the service type is videoConference234). The service session related UAP is assumed to be present on the user's terminal.

Preconditions:

An access session exists between the PA (user A) and UA (in provider domain). An access session related UAP shows the user the services which he can start.

Scenario:

1. The access session related UAP requests a list of services from the PA, which the user has been subscribed to. The PA makes the same request to the UA, which returns the list. The UAP displays the list to the user. The user selects a service to start[21]. The UAP requests PA to start the service.
2. The PA starts the service session related UAP[22], associated with this service session, and informs it of the service type that it should start (videoConference234).
3. The service session related UAP requests a new service session of service type videoConference234, from the PA. (The UAP may pass information about itself to the PA, including session models and feature sets supported and references to its operational and stream interfaces.)

21. The preceding interactions are not shown in the figure.
22. If the service session related UAP is not available in the user's domain, PA may attempt to download the UAP and continue.

4. PA requests to start a new service session of the service type (videoConference234), to (user A's) UA. (It may also pass the information about the UAP.)
5. UA may perform some actions, which are not prescribed by TINA, before continuing. For example, the UA checks the new session request against the user's subscription profile[23], to verify that the user has subscribed to this service and that it can be used with the terminal configuration of the user. Other decisions may also be taken. UA raises an exception to the PA, if the UA declines to start the service session.
6. UA gets a reference to a service factory which can create service session components for the service type (videoConference234). [24]

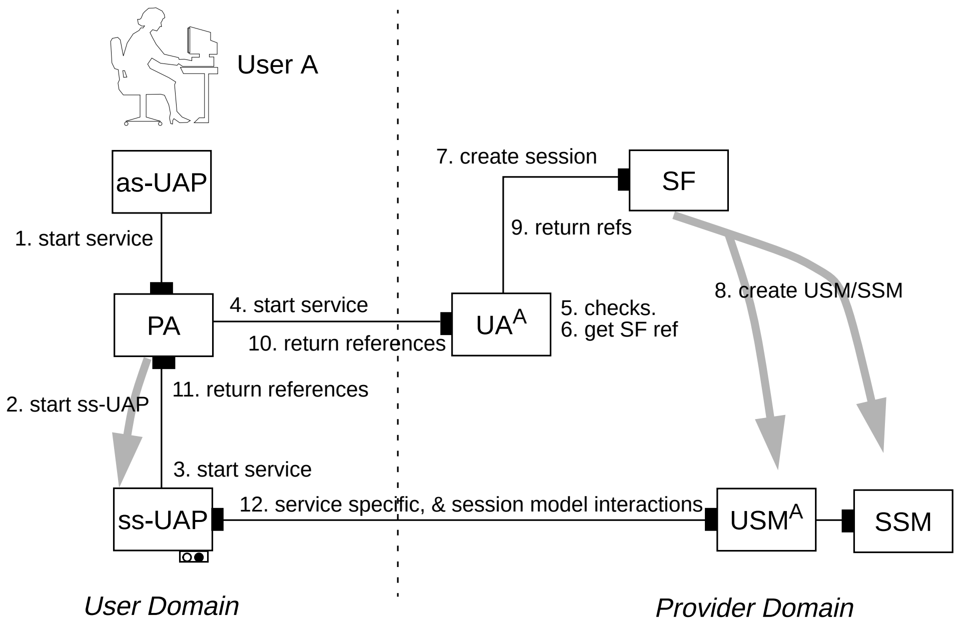


Figure 7-9. Starting a new service session

7. UA requests that a new session of the service type (videoConference234) is created by the Service factory.
8. Service factory creates an SSM and a USM[25] and initializes them.
9. Service factory returns interface references of the USM and the SS to the UA.
10. UA returns references of the USM and SSM to the PA.
11. PA returns references of the USM and SSM to the service session related UAP.
12. The service session related UAP and USM (and SSM) can interact using service specific interfaces or interfaces defined by session models, including the TINA session model. Some interactions between these components may be necessary before the user can use the service.

23. The user's subscription profile may define preferences and constraints on the invocation of a service. These preferences/constraints may be dependent on the user's current location. This provides support for personal mobility.

24. The UA may use a location service to find an appropriate service factory, or some other means. The UA may also provide other information to scope the search for the service factory, such as terminal configuration information. Other means include: the subscription information could potentially contain an interface reference to the service factory to use.

25. The Service Factory creates the computational objects which comprise the service session. These may include the USM and SSM. Other computational objects are also possible.

At this point User A is the only user involved in the service session. Some services may be single user only services, or may be used by a single user. As this is an example of a video conference service, user A probably wants to invite some more users to join in the session.

7.4.4 Inviting a User to Join an Existing Service Session

This example shows user A inviting another user (B) to join in the service session. The example ends when the invitation has been delivered to user B. The example of user B actually joining the session is given in Section 7.4.5, "Joining an Existing Service Session".

This example assumes that the invited user B is a named user, and is represented in the provider by a named user agent. Anonymous users, represented by an anonymous user agent, cannot be invited to join a service session because it is not possible to locate the specific user, as anonymous user agents do not publish the identity of the user (and may not even know the user’s identity).

This scenario supports user mobility by allowing a user to be invited to join a session, irrespective of their location. (This does not mean that they will automatically be able to join the session.)

Preconditions:

An access session exists between the PA and UA of the user sending the invitation (user A). It is NOT necessary for an access session to exist between a PA and the UA of the user receiving the invitation (user B), but for this example assume that an access session does exist for user B.

User A is using a service session related UAP and has a service session established with a USM and SSM. User A wishes to invite user B to join this service session. User A is ‘active’ in the service session, i.e. they have not suspended their participation.

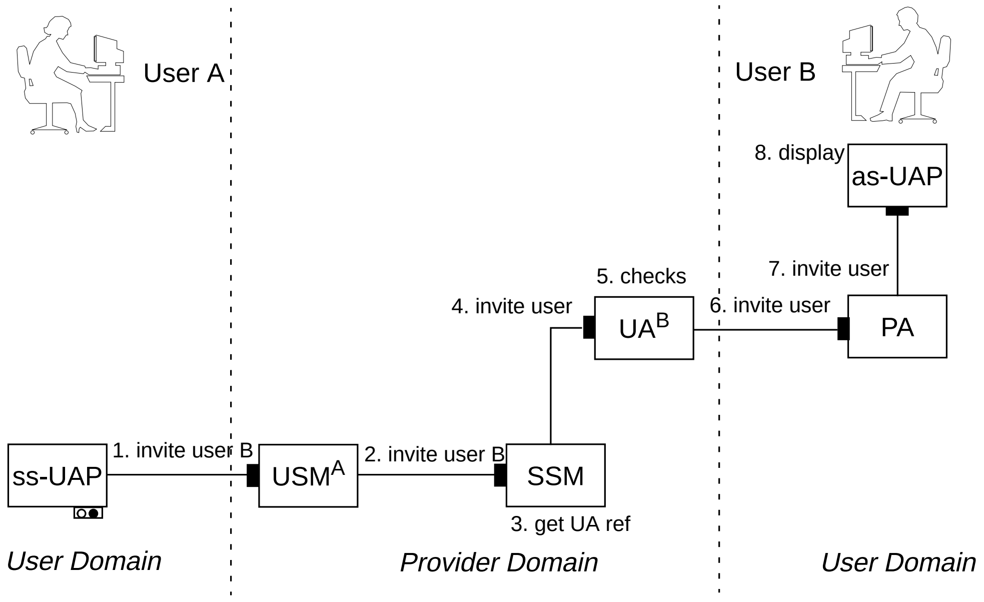


Figure 7-10. Inviting a user to join an existing service session

1. User A uses UAP to invite another user (invitee) to join a session. (User A supplies the user name of the invitee, or a user defined alias which can be resolved by the inviter’s UA.) The UAP requests[26] the USM to invite user B to join the session.

26. This request can be made using the Multiparty Feature Set interface on the USM, if the TINA session model is being used; or on a service specific interface.

2. USM requests[27] the SSM to invite a user to join the session. (Both the USM and SSM may check that User A is allowed to invite User B. These checks are not defined by TINA.)
3. SSM gets a reference to an invitation interface of user B’s UA[28].
4. SSM sends an invitation using the invitation interface of the user B’s UA.
5. Invitee’s UA may perform some actions, which are not prescribed by TINA, before continuing. For example, the UA may check the user profile within the UA for a policy on invitations. The policy will then determine the UA actions and interactions with other objects. UA may raise an exception to the SSM, if the UA declines to deliver the invitation.
6. In this example an access session exists between user B’s UA, and the PA on User B’s terminal. The invitation is delivered to the PA, by using an invitation interface on the PA.
7. PA sends the invitation to the access session related UAP.
8. The UAP displays the invitation to user B.

The invitation to join the service session has been delivered to user B’s UA, PA, and is displayed by the UAP. The invitation contains sufficient information for the UA to locate the service session, and allow the user to join it (as described in Section 7.4.5, "Joining an Existing Service Session"). Only some of this information will be passed to the P and UAP.

In the example above an access session already existed between user B’s PA and UA. If user B is NOT currently in an access session with the UA, then there are several alternatives as to what happens. It is not currently defined which of these alternatives must be supported as a mandatory capability and which are optional. The alternatives are:

- UA stores the invitation until the invited user establishes an access session. When he does establish an access session, the invitation is delivered as above;
- UA delivers the invitation to a registered terminal. (The terminal would have been selected by the user to receive invitations when no access session was present)[29];
- UA returns the address of a registered terminal to the SSM;
- UA forwards the invitation to another UA. (This UA would have been selected by the user to receive invitations when no access session is present. The UA may be in a different provider’s domain.);
- UA returns the address of another UA;
- UA starts a service session of a specified type. (The invitation may be sent to the service session, as part of its configuration, or later.).

7.4.5 Joining an Existing Service Session

This example shows a user B joining an existing session, after receiving an invitation to join the session.

User B is assumed to be in an access session with the provider and to have a valid subscription to the service (the service type is videoConference234). The service session related UAP is assumed to be present on user B’s terminal.

27. This request can be made using the Multiparty Feature Set interface on the SSM, if the TINA session model is being used; or on a service specific interface.
28. The UA may use a location service for locating user B’s UA, or some other means.
29. This case is required to support personal mobility. (See Section 8, "Personal and Session Mobility".)

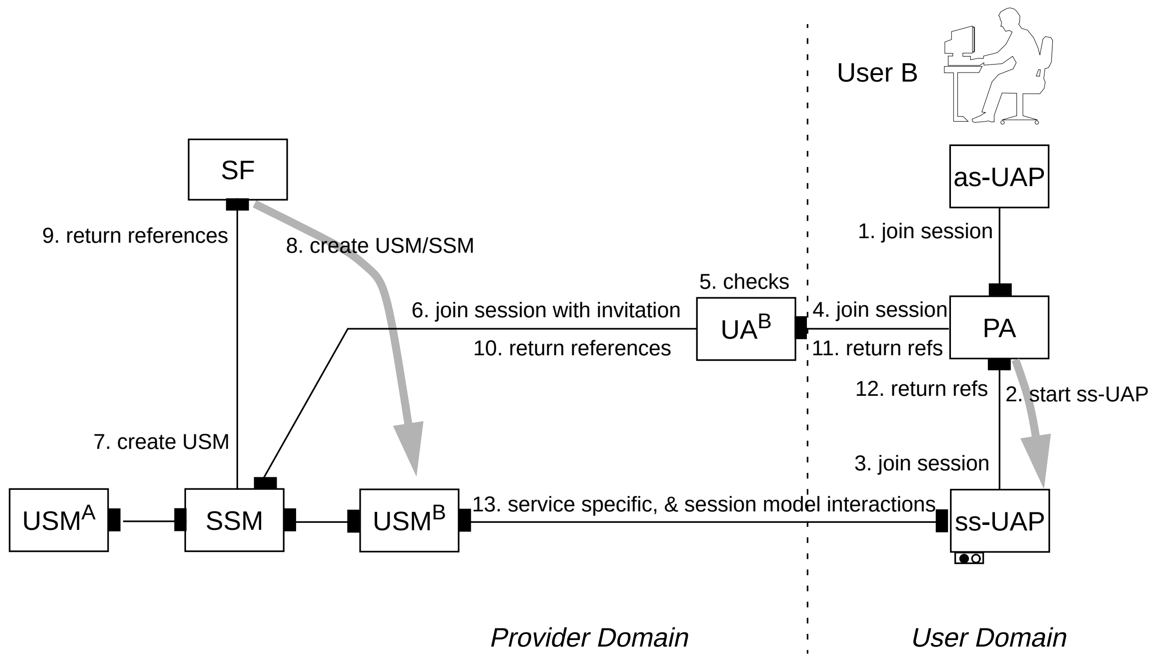


Figure 7-11. Joining an existing service session

User B can join this session from any terminal, from which he has established an access session. When the access session is established, the PA requests a list of the invitations received by the UA. The PA can then request to join any of the sessions. However, in this example we assume the invitations have been delivered to USer B's PA and as-UAP as described in Section 7.4.4.

Preconditions:

An access session exists between the PA (user B) and UA (in provider domain). User B's UA and PA have received the invitation to join the service session, and an access session related UAP shows the user the invitation which he has received.

Scenario:

1. The access session related UAP displays a list of invitations to join service sessions. The user selects an invitation to join the service session. The UAP requests PA to join the service session, giving the invitation id.
2. The PA starts a service session related UAP[30], associated with this type of service session, and informs it of the invitation id that it should request to join.
3. The service session related UAP requests to join the service session, giving the invitation id, from the PA. (The UAP may pass information about itself to the PA, including session models and feature sets supported and references to its operational and stream interfaces.)
4. PA requests to join the service session, giving the invitation id, to (user B's) UA. (It may also pass the information about the UAP.)
5. UA may perform some actions, which are not prescribed by TINA, before continuing. For example, the UA may check the invitation id against the user's current invitations, as well as the

30. If the service session related UAP is not available in the user's domain, PA may attempt to download the UAP.

user's subscription profile[31] to verify that they are subscribed to this service, and that it can be used with the current terminal configuration, etc. UA can decline the user to join the session.

6. UA gets a reference to the SSM[32] and requests to join the session. (It may pass some information in the invitation to confirm that the SSM invited this user to join the session.)
7. SSM requests its service factory to create a USM for user B.
8. Service factory creates a USM and initializes it.
9. Service factory returns interface references of the USM to the SSM.
10. SSM returns references of the USM and itself to the UA.
11. UA returns references of the USM and the SSM to the PA.
12. PA returns references of the USM and the SSM to the service session related UAP.
13. The service session related UAP and USM (and SSM) can interact using service specific interfaces or interfaces defined by session models, including the TINA session model. Some interactions between these components may be necessary before the user can use the service.

At this point both user A and user B are involved in the service session. As this is an example of a video conference service, either user may invite other users to join the session. There may be some service specific policy to decide whether a particular user in the session is allowed to invite other users to join.

7.4.6 Request and Establishment of a Stream Binding

This example shows the set-up of a stream binding between UAPs in consumer 1 and 2's domains. Consumer 3 requests the establishment of a stream binding in which consumer 1 and consumer 2 shall participate, but doesn't himself participate in the stream binding[33].

Precondition:
The 3 consumers have already been invited to the service session and are now parties in the service session. Consumers 1,2 and 3 support the participant oriented stream binding feature set (role: party) for stream bindings[34],while the service provider supports the participant oriented stream binding feature set (role: provider).

Post-condition:
A stream binding is established between the two participants, party1 and party2, and party3 has a reference to the stream binding in order to control it further.

The scenario shows a successful setup, but at certain points it is made clear that different decisions could have been made.

31. The user's subscription profile may define preferences and constraints on the invocation of a service. These may be dependent on the user's current location. This provides support for personal mobility.

32. The invitation may contain a reference to an interface on the SSM to use to join the session. Or the UA may use information in the invitation along with a location service to find the SSM, or some other means. The UA may also provide other information to the SSM, such as terminal configuration information, and application information.

33. This example is chosen to illustrate the separation of service session and communication session. Of course other examples are possible, e.g., one (more 'POTS-like') where consumer 1 acts as the initiator of the service session and also as the initiator of the communication session. The separations of access and usage and of service session and communication session still have meaning and add value (e.g., support mobility aspects).

34. Since no separate feature set exists at the moment for controlling a stream binding (without participating), it is likely that a new role controller will be introduced, and then consumer 3 could support the participant oriented stream binding feature set in the controller role instead.

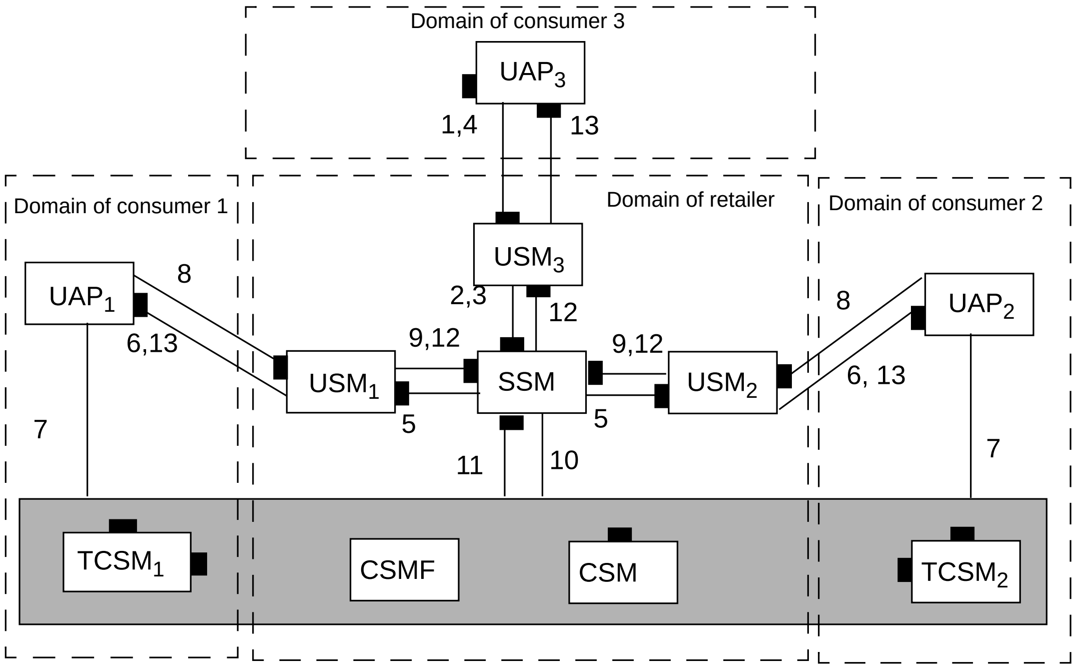


Figure 7-12. Stream binding request and setup
3 parties in the service session, communication session involving 2 of them

1. UAP_3 requests the setup of a stream binding with party1 and party2 as the participants. USM_3 may optionally make the necessary checks to make sure consumer 3 is allowed to setup the requested stream binding.
2. USM_3forwards the request to the service session's SSM.

. Optionally, SSM may check for permission to set up this stream binding; if necessary it may negotiate for permission with the other session members in the session graph. (This will then involve the voting feature set (not shown)).

3. If permission is obtained, SSM returns a stream binding identifier, as well as a request identifier for later confirmations to USM_3.
4. USM returns a stream binding identifier, as well as a request identifier, for later confirmations to UAP_3.

. (The following can be done in parallel to '1' and '2', hereafter called 'i')

5. SSM requests USM_i to join the stream binding.

. USM_i may optionally make decisions, e.g., about non-participation on behalf of consumer i.

6. USM_i forwards the request to UAP_i.
7. UAP_i starts an application setup scenario to get the NFEP(pool)s related to the stream interface user by consumer i in this service.
   This may have been done already[35], or it must be done now in order for the consumer to participate in this stream binding.
8. UAP_i accepts and returns this acceptance to USM_i, together with a description of consumer i's terms of participation in the stream binding, as well as a stream interface descriptor.
9. USM_i forwards this acceptance and the associated information to SSM.

. Depending on the answers from the participants (and specific logic for this service), the SSM may choose to give up establishing the stream binding, or the stream binding request will result in a request for communications (this is what is shown).

10. SSM requests to setup a communication session (if not already existing), and then requests the stream flows associated with the stream binding to be set up.

11. Final notifications back to SSM.

12. Final notifications back to USMs for consumer 1,2 and 3 (or just some of them, depending on their reply in step 9 (for each consumer i).

13. Final notifications back to consumers 1, 2 and 3 (or just some of them, depending on the reply from UAP in step 9, and/or depending on the behavior of USM_i [36].)

7.4.7 Composition: a service starts another service and acts in the usage party role.

This example shows one service using another to create a compound service. In this example, retailer domain 1 and provider domain 2 are peers, and use PeerA's to interact. However, a PA and UA could also be used for this scenario, if the domains were acting in user-provider access roles.

Preconditions:

A service session exists in retailer domain 1. To fulfill a user request, it needs to make use of another service of type B. Provider 2 supports service type B.

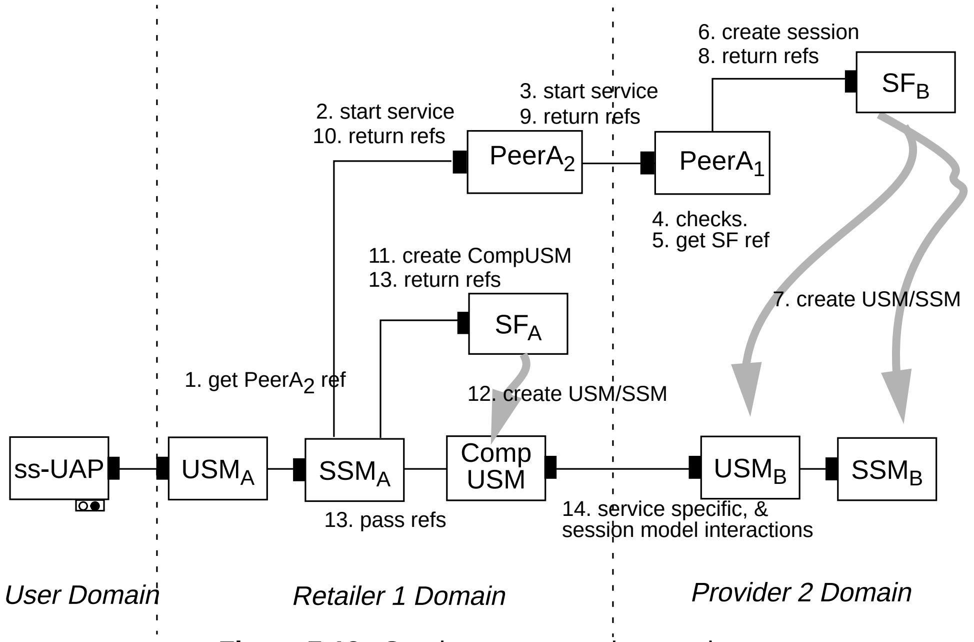


Figure 7-13. Starting a new service session

35. Depending on the service type and the terminal capabilities, there might be several cases: If the application is the only one ever using streams, the nodal (terminal) part of the stream binding may be hardwired. It might also be the case that, when receiving the invitation, consumer 1 already knows (or finds it likely) that he will be asked to participate in a stream binding, and starts to prepare the terminal internal actions needed to get hold of a stream interface (e.g., ask another application to release a stream interface or kill some applications in order to increase performance). This shows specialization of behavior of the UAP.

36. It is possible that the USM gets the notification, but does not forward it to the UAP; this is similar to what is explained in step 5, where USM takes decisions (e.g., 'screens') on behalf of the user / UAP.

Scenario:

1. SSM_A, the service A service session manager, locates the access component, $PeerA_2$, of a provider offering service B (it may also be a PA, depending on the specific composition relationship between the domains).
2. SSM_A requests a new service session of service type B from $PeerA_2$. (The SSM passes information, including required session models, usage role - provider in this case, feature sets supported, management context information and interface references).
3. $PeerA_2$ requests to start a new service session of the service type (B), to $PeerA_1$, the domain access component representing the Retailer in the provider's domain (depending on the case, it may be a UA). It also passes the requested model, role, feature sets, management context, and other service information.
4. $PeerA_1$ may perform some actions, which are not prescribed by TINA, before continuing. For example, $PeerA_1$ checks the new session request against the contracts between these domains, to verify that the request is valid. Other decisions may also be taken. $PeerA_1$ raises an exception to $PeerA_2$, if $PeerA_1$ declines to start the service session.
5. $PeerA_1$ gets a reference to a service factory which can create service session components for the service type (B). [37]
6. $PeerA_1$ requests that a new session of the service type (B), supporting the required session model, usage role and feature set, is created by the service factory.
7. Service factory creates SSM_B and USM_B and initializes them.
8. Service factory returns interface references of USM_B and SSM_B, to $PeerA_1$.
9. $PeerA_1$ returns references of USM_B and SSM_B to the PeerA $_2$.
10. $PeerA_2$ returns references of USM_B and SSM_B to SSM_A.
11. SSM_A requests that its SF create a composer usage session manager component, CompUSM, supporting the same feature sets and service model as requested for service B, but exporting the usage party role.
12. The SF creates and initializes the CompUSM to support interactions between the services.
13. The SF returns the CompUSM interface references to the $SSM_{A.}$
14. SSM_A passes the references to USM_B (and SSM_B if necessary) to the CompUSM.
15. The CompUSM and USM_B can now interact using specified session model and feature set interactions. The service session A takes the usage party role and can use service B much like an end-user.

7.4.8 Service Federation: A New User Joins a Service Session

In this example, no federation is formed until the invited user joins the session.

37. The PeerA may use a location service to find an appropriate service factory, or some other means. The PeerA may also provide other information to scope the search for the service factory, such as terminal configuration information. Other means include: the subscription information could potentially contain an interface reference to the service factory to use.

7.4.8.1 Inviting a user in another retailer domain to join an existing service session.

This example shows user A inviting another user (B) to join in the service session, when B is not a user in the same domain as A. The example ends when the invitation has been delivered to user B. As before, B is assumed to be a named user.

Preconditions:

An access session exists between the PA and UA of the user sending the invitation (user A). It is NOT necessary for an access session to exist between the domains of Retailer 1 and Retailer 2, and between a PA and the UA of the user receiving the invitation (user B). However, for this example it is assumed that an access session does exist between the Retailers, and between Retailer 2 and user B.

User A is using a service session related UAP, and has a service session established with a USM and SSM. User A wishes to invite user B to join this service session. User A is 'active' in the service session, i.e. they have not suspended their participation. This example reuses the simple invitation case, inserting extra steps for passing the invitation between retailer domains. In this alternative, the Peer Agent, PeerA, is responsible for locating an appropriate UA for the user and passing on the invitation

.

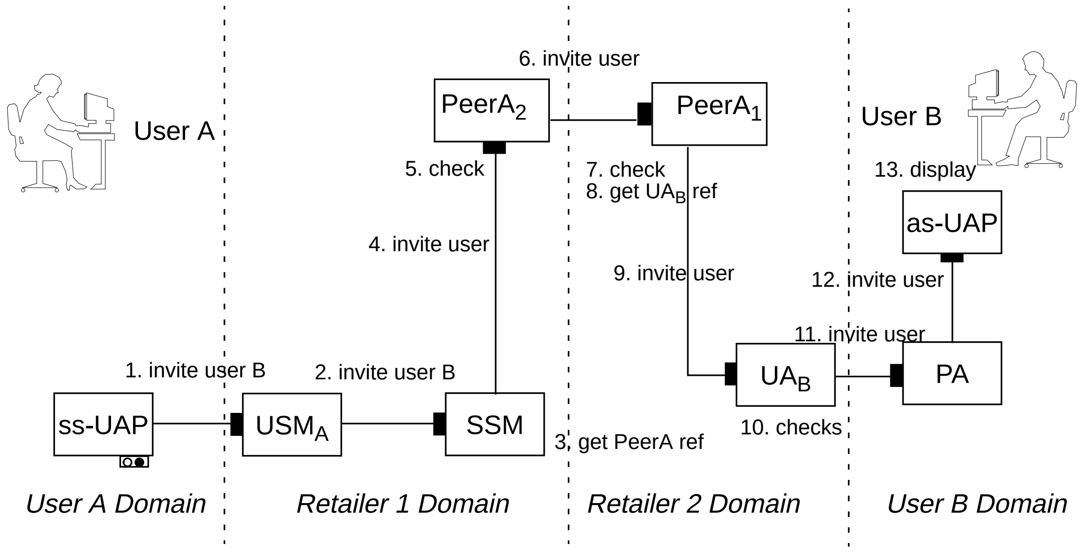


Figure 7-14. Inviting a user to join an existing service session

1. User A uses UAP to invite a user (invitee) to join a session. (User supplies the user name of the invitee, or a user defined alias which can be resolved by the inviter's UA). The UAP requests[38] the USM to invite user B to join the session.
2. USM requests[39] the SSM to invite a user to join the session.
3. SSM gets a reference to an invitation interface that can act for user B. The SSM need not be aware that this is in fact an interface of the PeerA acting for retailer 2.

38. This request can be made using the Multiparty Feature Set interface on the USM, if the TINA session model is being used; or on a service specific interface.

39. This request can be made using the Multiparty Feature Set interface on the SSM, if the TINA session model is being used; or on a service specific interface.

4. SSM sends an invitation using the invitation interface of PeerA$_2$.
5. PeerA$_2$ may perform some actions, which are not prescribed by TINA, before continuing. For example, it may check that this is a user that it knows about or that the invitation conforms to the federation contract (e.g. for a known service type). The policy will then determine the PeerA$_2$ actions and interactions with other objects. PeerA$_2$ may raise an exception to the SSM, if the PeerA$_2$ declines to deliver the invitation.
6. PeerA$_2$ sends the invitation to PeerA$_1$, the peer agent representing supporting this access session in Retailer 2's domain, using its invitation interface.
7. PeerA$_1$ may also check the validity of the invitation, raising an exception with PeerA $_2$ if it needs to refuse the invitation request.
8. PeerA $_1$ locates UA$_B$, the UA for User B.
9. PeerA $_1$ sends the invitation to UA$_B$, using its invitation interface.
10. In this example an access session exists between user B's UA, and the PA on User B's terminal. UA$_B$ may perform some actions, which are not prescribed by TINA, before continuing. For example, the UA may check the user profile within the UA for a policy on invitations. UA may raise an exception to the SSM, if the UA declines to deliver the invitation.
11. The invitation is delivered to the PA by using an invitation interface on the PA.
12. PA sends the invitation to the access session related UAP.
13. The UAP displays the invitation to user B.

The invitation to join the service session is delivered to user B's UA, PA, and is displayed by the UAP. The invitation contains sufficient information for the UA to locate the service session and allow the user to join it, (as described in Section 7.4.5, "Joining an Existing Service Session"). Only some of this information will be passed to the PA, and UAP. As for the simple case, this is not the only action the UA may take on receiving an invitation.

7.4.8.2 Joining a federated service session

This example shows a user B joining an existing session in another domain via federation, after receiving an invitation to join the session. User B is assumed to be in an access session with the provider and to have a valid subscription to the service. The service session related UAP is assumed to be present on user B's terminal. The first five steps are unchanged from the simple case.

Preconditions:

An access session exists between the PA (user B) and UA (in provider domain). User B's UA and PA have received the invitation to join the service session, and an access session related UAP shows the user the invitation which he has received.

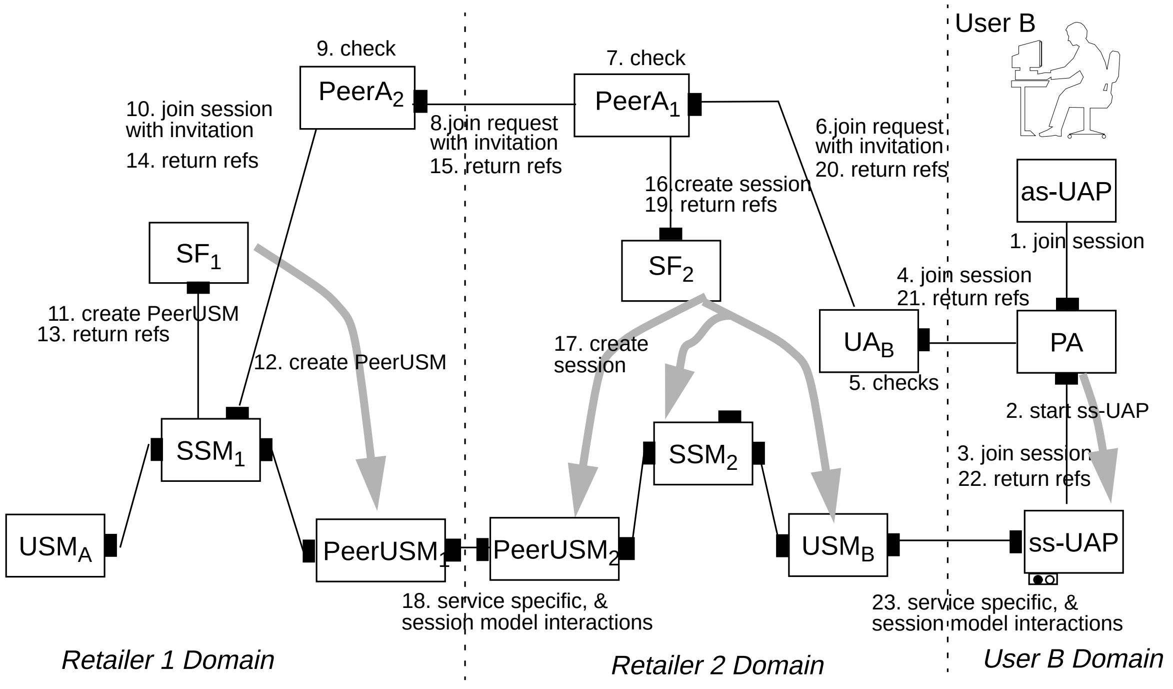


Figure 7-15. Joining an existing service session

1. The access session related UAP displays a list of invitations to join service sessions. The user selects an invitation to join the service session. The UAP requests PA to join the service session, giving the invitation id.
2. The PA starts a service session related UAP[40], associated with this type of service session, and informs it of the invitation id that it should request to join.
3. The service session related UAP requests to join the service session, giving the invitation id, from the PA. (The UAP may pass information about itself to the PA, including session models and feature sets supported and references to its operational and stream interfaces.)
4. PA requests to join the service session, giving the invitation id, to (user B's) UA. (It may also pass the information about the UAP.)
5. UA may perform some actions, which are not prescribed by TINA, before continuing. For example, the UA checks the invitation id against the user's current invitations. And the UA may check the user's subscription profile[41] to verify that the user has subscribed to this service, and that it can be used with the terminal configuration of the user. Other decisions may also be made. UA raises an exception to the PA, if the UA declines to allow the user to join the session.

40. If the service session related UAP is not available in the user's domain, PA may attempt to download the UAP and continue.

41. The user's subscription profile may define preferences and constraints on the invocation of a service. These preferences/constraints may be dependent on the user's current location. This provides support for personal mobility.

6. UA gets a reference to $PeerA_1$[42], and requests to join the session, using its request interface.
7. $PeerA_1$ checks the join request and realizes that a federation is required to support it.
8. $PeerA_1$ creates a federation join with invitation request (at this point negotiation over the context is possible) and passes it to $PeerA_2$.
9. $PeerA_2$ may perform some actions, which are not prescribed by TINA, before continuing. For example, it may check the invitation id. $PeerA_2$ may raise an exception to $PeerA_1$, if $PeerA_2$ declines to accept the join.
10. $PeerA_2$ passes on a federated join with invitation request to $SSM_{1.}$
11. SSM_1 makes a peer usage manager component request to its service factory, $SF_{1.}$
12. SF_1 creates a peer usage manager component ($PeerUSM_1$) and initializes it.
13. SF_1 returns its interface reference to $SSM_{1.}$
14. SSM_1 acquires the exported interface references for $PeerUSM_1$ and returns them to $PeerA_{2.}$
15. $PeerA_2$ passes the interface references to $PeerA_{1.}$
16. $PeerA_1$ acquires a suitable service factory reference (to SF_2). $PeerA_1$ requests SF_2 to create a service session of the required type and to support user B in a normal usage party role, and in addition, to support federated session 1 in a usage peer role.
17. The SF_2 creates and initializes the requested session components.
18. The two PeerUSMs can now interact with each other for service specific and session model purposes. To establish a federation, a number of interactions may be necessary. (Note that no contact between the service and user B is made at this stage.) Probably, a first interaction with $PeerUSM_2$ is required for $PeerUSM_1$ to pass it its interface references.
19. SF_2 returns the interface references to $PeerUSM_2$, USM_B and SSM_B to $PeerA_{1.}$
20. $PeerA_1$ returns the relevant interfaces ($USSM_B$ and possibly some SSM_B interfaces) to UA_B.
21. UA_B returns references of the USM and the SSM to the PA.
22. PA returns references of the USM and the SSM to the service session related UAP.
23. The service session related UAP and USM (and SSM) can interact using service specific interfaces, or interfaces defined by session models, including the TINA session model. Some interactions between these components may be necessary before the user can use the service.

At this point both user A and user B are involved in a federated service session. Alternatively, the session in retailer 2 could have been established before passing on the join with invitation request. This is shown in Annex A-4.3, "Joining an already federated service session".

The evolution of the service session graph along this example is shown in Annex A-4.4, "Evolution of the service session graph during a federation session".

42. The invitation may contain a reference to an interface on the PeerA to use to join the session. Or the UA may use information in the invitation along with a location service to find the PeerA, or some other means.

8 Personal and Session Mobility

8.1 Introduction

This section focuses on mobility in services. Two types of mobility are discussed within the scope of the service architecture: personal mobility and session mobility. Personal mobility is the ability of the end-user to access services and to receive invitations to services on different terminals in different locations. Session mobility is the ability for an end-user to maintain a session, independent of the equipment used, to support that session. Note: in this chapter the term “user” refers to the end-user (consumer) of the service.

Personal mobility and session mobility are inherently supported by the TINA service architecture; the purpose of this chapter is just to make explicitly clear which features of the architecture make them possible.

Section 8.2 focuses on personal mobility, while section 8.3 focuses on session mobility. For both cases a definition is given, requirements on the service architecture are outlined, and crucial functionality needed in the computational model to fulfill the requirements is explained.

Terminal mobility is not discussed in this chapter. Different ways of supporting terminal mobility in TINA are possible. It may be done partially in the service architecture, but it is also possible to handle terminal mobility fully outside the scope of the service architecture [23]. This choice has not been made yet, and therefore no further details are presently included in this document.

8.2 Personal Mobility

8.2.1 Definition

Personal mobility enables users to use services that are personalized with their preferences and identity ubiquitously, independently of both physical location and specific equipment. In this situation, the level of service obtained is dependent only on the capabilities of the access equipment and/or method, and restrictions imposed by the retailer and subscription.

8.2.2 Requirements

The requirements for personal mobility in TINA have been inspired by UPT (Universal Personal Telecommunication [39]). However, there are some important differences between personal mobility in TINA and personal mobility as offered by UPT. UPT is specifically designed as a separate service to support personal mobility in existing networks. It exists in addition to other services available in those networks. Furthermore, UPT has its own set of supplementary services and is not applicable to other supplementary services in the networks in which it is implemented.

TINA, on the other hand, should support personal mobility inherently, i.e., personal mobility support is a feature of a TINA system, rather than a separate service. It is (technically) applicable to any other service in a TINA system, including ancilliary services. As opposed to UPT, no services specific to personal mobility in TINA are specified, although any retailer has the freedom to do so.

Detailed requirements are:

- Users should be able to access the system from any terminal, subject to restrictions imposed by the retailer and by the subscriber[1], while being charged to the subscriber’s account. To “access the system” includes:
  - to initiate access sessions,

  - to initiate service sessions (and the associated communication sessions),
  - to receive invitations to service sessions at a specified terminal.
- Users should have the ability to register so that service invitations are sent to a terminal specified in the registration. That is, the registration will result in invitations to join a service arriving at the terminal specified. Conditions may be applied to this registration, for instance, a user may register at different terminals for different services, different callers, different times of day, etc. The architecture should allow the support of this conditional registration, but it is a retailer issue to define and implement the actual conditions that are supported.
- Users may respond to a service invitation on a different terminal than the one specified for getting the invitation.
- Restrictions to the user's ability to access and use the system independent of location (i.e., the used terminal) can be imposed in the following ways:
  - According to the contract the subscriber has with the retailer. This is defined in the subscription.
  - According to the agreements between subscriber and user. This is defined in the user profile.
  - According to agreements between the user and the owner of the terminal used. That is, the owner of the terminal may or may not allow the user to use that terminal.
- The ability for a user to use services from any terminal may be restricted according to the capabilities of the terminal and network access point. This limitation shall not apply to the access session, but may apply to the service session. If a terminal and network access point do not have all capabilities required for a service, the service may be offered with a restricted service profile and QoS. Nevertheless, the terminal should at least contain a PA.
- The user should be able to "access the system" according to the retailer's and user's own preferences, as much as possible, independent of the terminal used. The delivery of a service is subject to terminal and network capabilities.

The provision of personal mobility in a multi-retailer environment is also required: it should be possible for a retailer, different from the one that provides personal mobility to the user, to invite him/her to a service session. This issue is not addressed in the version of the service architecture.

8.2.3 Requirements Fulfillment

This section explains, from a computational viewpoint, how the requirements given in 8.2.2 are supported by the service architecture. For this exercise those requirements have been split into smaller ones, according to the following subsections.

8.2.3.1 Users Should Be Able to Access the System from Any Terminal

This requirement is fulfilled if the PA, present on the terminal that the user uses to access the system, is able to contact the user's own UA in a chosen retailer's domain. In this document, the following approach is used to realize this:

1. Remember that the role of a subscriber (having the contract with the retailer and paying the bills) is different from the role of the user (using the services). For instance, the subscriber could be a company, while the users would be the employees of that company.

- The user provides the PA with his/her user name and retailer name (either through the user's own knowledge, or through some device, e.g. a smart card[2]);
- The PA contacts a (DPE-) naming service, which is able to find the interface reference of an IA.
- The PA requests to the IA that an access session be set-up with the namedUA of the user, providing the user name to the IA. Security procedures are executed between PA and IA. The IA returns a reference of an access interface of the namedUA.
- The access session is then established. The PA can perform further requests (e.g., to start a service) to the UA.

Once an access session is active, the terminal and network may exchange interface references of other objects that are required to start a service session, but this is not specific to personal mobility.

8.2.3.2 Access and Invocation of Services is Subject to Restrictions Imposed by Subscription and User Profile

The ability of a user to access the system from any terminal is subject to restrictions imposed by the retailer and the subscriber. These restrictions are described in the subscription and user profile respectively. Different restrictions may be applicable to respectively access session and service session (e.g. the user may be allowed to start access sessions and certain types of service sessions from any terminal, while other types of services may only be started from a restricted set of terminals).

This feature is realized in the service architecture as follows:

- Upon a user's request to start an access session from a certain terminal, the UA will check the user profile to find out whether this access session is allowed from that terminal. The subscription does not need to be checked explicitly, as the subscriber should only be allowed to modify the user profile within the limits set by the subscription.
- Upon a user's request to start a service session from a certain terminal, the UA will check the user profile to find out whether this service session is allowed from that terminal. The subscription does not need to be checked explicitly, as the subscriber should only be allowed to modify the user's profile within the limits set by the subscription.

Note that this check for location restrictions is not visible as a separate operation, but only as a parameter of a more general check on the user profile.

8.2.3.3 Access Restrictions by Terminal Owners

Restricting a user from accessing a terminal should be handled by the terminal owner, e.g., by having a password on his/her terminal. This, however, is a matter of terminal implementation and not within the scope of TINA.

The best way to prevent a user from being invited at a certain terminal is to implement screening functions in the terminal itself. A retailer could also implement this type of functionality; however, nothing technical can prevent other retailers from *not* implementing this type of screening, i.e., sending invitations to "unwanted" users to that terminal anyway. Therefore, this type of function is best implemented in the terminal itself.

2. The interface between user and PA is outside the scope of TINA.

8.2.3.4 Usage of Services is Subject to Terminal and Network Access Point Capabilities

This point is not specific to personal mobility, because it is also true in case a user is always at the same location. Nevertheless, in the personal mobility case, it is likely to be seen as a problem, for instance of a user does not have capabilities on a “foreign” terminal that (s)he has on his/her “home” terminal. However, since it is not specific to personal mobility, the service session objects need a mechanism to be able to adapt service requests to the capabilities of the terminal and network access point. Therefore, we will not go into further detail in this chapter.

8.2.3.5 User Registration

Users should have the ability to register for service invitations to be sent to a terminal as specified in the registration. This requirement is realized by means of the procedure “user registration”. The registration will result in invitations to join a service arriving at the specified terminal.

The user registration procedure for TINA is illustrated in Figure 8-1. This procedure should be preceded by the establishment of an access session. The figure also shows the steps performed when, later, the user receives an invitation.

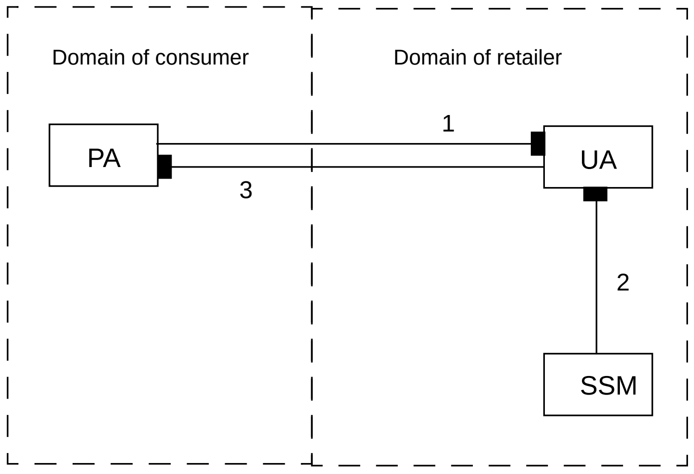


Figure 8-1. User registration and invitation

The user interacts with the PA in his terminal to perform registration at that terminal, providing it with the necessary information (see below).

1. The PA requests the user registration from the UA, providing it with the following information:
    - The identity of the terminal on which the user wants to register[3]. This information is needed by the UA to be able to contact the user during subsequent invitations, and is also used to check whether the user is allowed to register on that terminal.

3. More generic location information than the terminal identity could be provided, for instance, a room where a user can be reached. However, the interpretation of such aditional generic information is up to individual retailers to make, and not elaborated on in this chapter. The rest of this chapter only considers the terminal identity as location information. The terminal identity may coincide with a network access point.

- The condition that applies to the registration. This information is provider specific. For instance, it could consist of the service type, time of day, etc.

Note that the user may perform “remote registration”, i.e. specify another terminal for receiving invitations than the one used for executing the registration procedure. In that case the user has to explicitly specify the identity of the terminal, otherwise the PA may know it.

The UA checks with the user profile whether registration on the specified terminal with the specified conditions is allowed. The UA sets the user registration in the user’s usage context, where a list of locations (terminal identities), through which the user may be reached for different conditions, is kept. Then it returns to the PA the result of the procedure (success or failure).

If the procedure was successful, then from this moment on, the user should receive invitations at the terminal specified, according to the conditions indicated. What happens when the user receives an invitation is detailed in the following two steps:

2. The SSM of a service session forwards an invitation received from one of the session participants to the UA of the user who performed the registration.

   The UA looks up in the user’s usage context the identity of the terminal that should be used to invite the user, according to the valid conditions (e.g., time of day, caller-id, etc.), and contacts that terminal, to request the interface reference of its PA. Note that the UA should be able to contact the terminal on the basis of the terminal identity and the type name of a service on the terminal that can return a valid PA interface reference[4].

3. The UA may now send the invitation to the user (through the PA).

8.2.3.6 Uniform, Personalized Access and Usage of the System

A user should be able to access the system according to the retailer’s and the user’s own preferences, as much as possible independently of the terminal used. In other words, access to the system and the use of services should look, as much as possible, the same on every terminal used.

This requirement is clearly the most difficult one to realize, especially in the initial phase of access, as will be explained below.

There are three ways to tackle the problem of uniform access to services:

- Standardized services
  This approach could be applicable to a limited set of services, but is not the general approach that is taken for TINA, as it does not fit the future competitive multi-retailer environment, where retailers want to offer customized services.
- “Xwindows approach”
  In this approach there are no service specific components in the terminal, but instead there is a powerful command set available on terminals, that allows retailers to manipulate the user interface on terminals that support this command set. This approach is allowed in TINA, but not prescribed. Hence, a retailer cannot rely on this functionality to be available on a user’s terminal.

4. At present it is not clear whether this service on the terminal is offered by a new computational object (that has not been defined yet), or that the DPE could provide this service.

- Downloading of specific components
  In this approach, retailer, user, and/or service specific components (i.e., the PA and the UAP) are downloaded on a terminal when required. The PA is downloaded during the access phase, while a UAP (being service specific), may be downloaded at any time. A downloading mechanism is defined as part of the service architecture.

It seems that the most promising way to solve the requirement of uniform access is downloading of specific components. However, during the very initial phase of access retailer/user/service specific components are not available yet, so in this phase it is impossible to fulfil the requirement by means of the downloading mechanism.

Hence, to fulfill the requirement, a generic, “standardized” mechanism should be available for initial access of a user to a terminal. Note that, for example, for the service “UPT” the use of smart cards is envisaged in the future. The smart card may play an important role in solving the problem of initial access; for instance, if once inserted in a terminal, it can initiate the downloading of customized components from the retailer to the terminal.

8.3 Session Mobility

8.3.1 Definition

Session mobility enables a session to be maintained independently of the equipment used to support it, subject to capabilities of the equipment and restrictions imposed by retailer and subscriber.

Session mobility allows a user that has an active session on a particular terminal to move that session to another terminal (e.g., by suspending it on the first terminal, and resuming it on the second terminal).

Session mobility is, in principle, applicable to access session, service session and communication session, the latter not being within the scope of the service architecture.

8.3.2 Requirements

No special requirements are set for session mobility of an access session, because moving an access session from one terminal to another may very simply be realized by ending the access session on the old terminal, and starting a new one on the new terminal. This is possible because an access session can be viewed as consisting simply of a single state, once active. Hence, no specific support for mobility of an access session is needed.

However, the following requirements are defined for mobility of service sessions:

- The user must be able to suspend a user service session (i.e., to suspend his(her) participation in a service session) on a terminal and resume the participation using a different terminal[5] (which may be connected to a different NAP).
- A user service session can only be resumed if the service session is still active, i.e., a user cannot resume participation in a session that has been suspended globally. The user might be allowed to resume the whole session. This is not specific to session mobility, though.
- The ability to move a user service session is subject to restrictions imposed by the retailer and the subscriber.

5. Note that the participation in the service session may also be resumed at the same terminal. Such a case, however, is not relevant to session mobility, and therefore not further elaborated on in this section. Similarly, suspending a whole service session is possible, but not within the scope of this chapter.

- The terminal and network access point must meet a set of minimum requirements, similar to the personal mobility case. Specifically, the terminal must at least have a PA.
- The service architecture must assess how the requirements of the service match with the capabilities of the new terminal and access point, and act accordingly (e.g., adapt the way the service is delivered to the end-user if required). This might imply a renegotiation of the session model and of the FSs. As already discussed in Section 8.2.3, the way this is done is not specific to personal or session mobility, and therefore is not discussed further in this chapter.

8.3.3 Requirements Fulfillment

As mentioned in Section 8.3.2, only mobility of service sessions has to be handled. Session mobility relies on the procedures for suspending and resuming participation in service session, and suspending and resuming the whole service session. In addition, we should emphasize that session mobility also requires that communication resources are released on the “old” terminal, because the user moves to a new terminal. That is, during the “suspend” procedure, the SSM will request the CSM to release the stream flow connections to the UAP at the old terminal.

In principle the user could return to the service session, with the session being in the same state as it was before the user left the session. For this purpose the USM stores the local state of the session for the user. However, the global state of the session may change while the user moves from one terminal to another. This point may be resolved as follows:

Assume an active service session, with parties A, B, C, etc. Suppose party A suspends its session. Now suppose that, while A is suspended, party B asks for e.g., an audio component to be added. This request ends up in the SSM that has to check if negotiation is required. Negotiation will be required if B is not considered an owner for this operation (i.e., B is neither owner of the session, nor of the graph, nor of the session relationship (group) the audio component could be attached to) AND if there are other parties specified as owner for this operation. From A's point of view there are 2 cases:

1. A has to confirm the operation.
This is the case if all of the following conditions apply:

- B is not an owner,
- A is an owner, possibly together with other parties,
- The voting rule in the session graph indicates that A has to confirm (it is indeed possible that an agreement is obtained with the other owners, if the voting rule is not unanimity without A having to confirm explicitly).

2. A does not have to confirm the operation.
This is true in either of the following cases:

- B is an owner,
- B is not an owner, but A is not an owner either. (In this case it is still possible that other owners had to confirm but we suppose they did confirm, because the “refuse” case is of little interest for our problem.)

Now the approach to these two cases is as follows:

Case 1: the request for confirmation is issued by SSM to USM_A.

It is advisable that no operation remains hanging on an object. Therefore, because A is suspended, USM_A should react to the request on behalf of party A. There are two[6] possibilities for this reaction

1. All requests are rejected;
2. All requests are accepted.

The choice between these could be a USM configuration parameter. If the request is rejected there is no problem (from the point of view of A[7]). If the request is accepted, then the USM has to store the change (an “Info” operation) to forward it to A after it is resumed.

The simplest case is, of course, the one in which the USM refuses by default. The default acceptance should cause no difficulty. Both types of implementation can coexist in the same session.

Case 2: the operation is agreed without A's confirmation being required.

In this case either the SSM or the USM_A has to store the “Info” operation to be sent to the party A after the resume. For unification with the previous case, it is suggested that the USM_A stores the operation.

Note that the approach described above prevents inconsistency between the user service session and the global service session, when the state of the session changes while a user is suspended. However, it did not yet describe how the interaction to the communication session is dealt with during these possible state changes. This point can be solved as follows:

If party A suspends his/her service session, the SSM shall instruct the CSM to release the part of the communication session that involves party A. While party A is suspended, changes may occur in the service session (as described above), that may require changes in the communication session. The SSM will instruct the CSM to effectuate these changes, but only for the active parties, i.e., in our example not for party A. However, the SSM will keep the information required to establish the appropriate connections to party A, once party A resumes the session[8]. When party A resumes his/her service session, this will be notified to the SSM (by means of the “Resume_session” operation). The SSM will at that point instruct the CSM to set-up the appropriate connection(s) to party A, according to the session information it stores.

The two cases above describe a service generic way to solve the problem of a session changing state while the user moves from one terminal to another. But it must also be taken into account that, when a user wants to suspend his/her participation in a service session, there might be rules that depend on service specific logic forcing him/her to take some action first. Speaking in service session graph terms, this could include actions as diverse as:

- handing over control session relationships to other parties (and thus avoiding the need to communicate with the USM while the user’s session is suspended), withdrawing the associated stream interfaces,

6. Service specific solutions would allow the USM to choose based on the type of each request.
7. In case A suspended his/her participation later, while keeping control session relationships with voting rule making A perform “veto”, then there could be dead-lock situations.
8. Another service specific option here would be that the connections to party A are not automatically established, but instead A has the possibility of viewing the current configuration of the service session and deciding to establish them or not. This option might be relevant for services where the configuration is likely to change much while A is suspended. Of course, another option for those services is to quit, since the value (and price) of keeping the USM active may be questioned; the choice between suspending and quitting can be made by the user, or the service may reject suspensions or implement them as quit.

- grabbing control of the whole service session before leaving[9], thus ensuring the session remains unchanged: all requests for changes will go through his/her USM which will, in turn, stop them.

Figure 8-2 details the steps for a simple example of session mobility, where a user that is taking part in a service session suspends his participation and later resumes it, finding the session in the same state (s)he left it. The example assumes capabilities for multiple parties, end users' control (voting) , and stream binding session relationships.

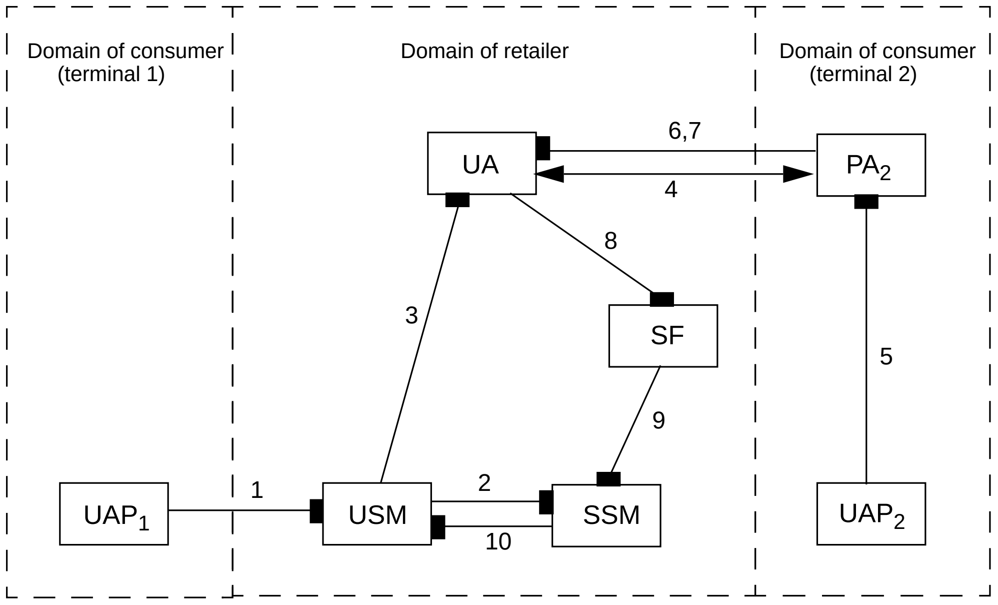

Domain of consumer (terminal 1)
Domain of retailer
Domain of consumer (terminal 2)
UA
PA2
6,7
4
8
3
SF
5
9
1
2
10
UAP1
USM
SSM
UAP2

Figure 8-2. Session mobility

The user has an access session with a retailer and is involved in a multiparty service session with active UAP (UAP_1), USM and SSM. He uses his UAP_1 to request the suspension of his participation in the service session.

1. The UAP_1 forwards this request to the user's USM.
2. The USM informs the service session's SSM of this request.

   The SSM may optionally notify other USMs of the participation suspension, and they may optionally forward the notification to their respective UAPs for the other users' information.

   As mentioned before, the SSM will request the CSM to release the part of the communication session that involves the party whose participation is going to be suspended. It will then return to the USM a confirmation, together with a session descriptor; this descriptor will be used when the participation of the user in the service session is resumed. The confirmation and descriptor will then be forwarded to the user via the UAP_1.

9. Although there are some aspects of control that can never be taken from a user, like the right to release his/her own communication resources and quit the session, controlling the "whole" session here means establishing an ownership control relationship with the top SSG.

3. The USM informs the UA of the session participation suspension and forwards the session descriptor to it.

Later the user wants to resume his participation in the service session. Since the different possibilities in case the state of the service session has changed have been outlined before, this example assumes it has not.

It will also be assumed that the user wants to resume his participation using a terminal that is different from the one where he was active in the service session before the participation suspension.This is the case that is relevant for session mobility.

4. The necessary interactions take place between the user's UA and the PA_2 in the new terminal to establish a new access session between the user and the retailer. Negotiation of FSs takes place.

   When the new access session is active, the user interacts with the UAP_2 and requests to resume his participation in a service session.

5. UAP_2 requests from PA_2 a list of suspended service session participations.
6. PA_2 forwards this request to the user's UA.

   The UA returns the list, which the PA_2 forwards to the UAP_2. Thus the list reaches the user who may choose the service session interactively with the UAP_2 and request it to resume it.

7. The PA_2 forwards this request to the UA, together with the service session's associated descriptor.

   The UA may check that he user is authorized to proceed. Then it uses a location service to request a reference to a service factory that can resume the USM that corresponds to the session descriptor.

8. The UA requests this SF to resume participation in the service session.
9. The SF forwards this request to the service session's SSM. Usage context information should be included in this request.
10. The SSM forwards the request and usage context information to the USM.

    When it receives a positive confirmation from the USM, the SSM may optionally notify other USMs that a user has rejoined the session; these USMs may optionally in turn notify their respective UAPs.

    Then the SSM requests from the CSM to set up the appropriate connection(s) to the new user.

    The confirmation is forwarded from the SSM to the SF, then to the UA (together with the necessary service session interfaces), then to the PA_2, to the UAP_2 and finally to the user.

9 Issues Requiring Further Work

This section identifies a reduced set of high-priority open issues, describing them in form of a shor problem statement.

The following main open issues have been identified:

- Internet-like services: this topic is planned to be one of the main focuses of the Core Team activity until the end of 1997; however, it is mostly relevant for the NRA (see below);
- Service management: this will likely be worked on by the core-team;
- Service composition and federation: this is likely to be addressed in the RtR and 3Pty reference points;
- Mobility: work on this topic is on-going in several TINA auxiliary projects and input is expected from them that may influence the service architecture;
- Role of the service factory component: this issue will be looked at by the Core Team, but it will not be included in the main working topics;
- Terminal capabilities;it is likely that auxilary projects will contribute to this issue;
- Service interactions: this issue is not likely to be addressed by TINA;
- Naming issue: this is not specific to the service architecture. This issue will be addressed by the naming work group that started its work April, 1997.

The issues are presented in more detail below.

9.1 Internet-like Services

The added value of using TINA for such services is to bring the TINA capabilities over the Internet namely: access session, service session, communication session, accounting, naming/typing and service compounding (telecom services and Internet services).

The most important aspect or characteristic of Internet services with regard to actual TINA services is that they use a **connection-less (network level) communication**. Currently only connection-oriented networks are modeled in the network resource architecture(NRA) [8]. The fact that something that is modeled as a stream will in fact use a connection-less transport network, is mainly outside the scope of the service architecture and will be solved in future versions of the network resource architecture(NRA). As seen from the service architecture and the computational modeling [5], this simply means that both operational interfaces and stream interfaces can be used by service components.

The fact that some services will only use operational interfaces and no stream interfaces is already handled in the service architecture by the defined feature sets.

Currently TINA supports the notions of explicit stream bindings and implicit operational bindings (as explained in this document, [8] and [5]). It is foreseen that both the implicit/explicit binding issue and the relationship between kTN and TN will be covered in new releases of NRA,CMC and DPE documents planned for 1997.

9.2 Service management

More work is needed on service management. It is likely that material that is currently in the annex part will be worked on by the core-team and moved to the main body in the next version.

9.3 RtR and 3Pty

Of course, the work on composition and federation is closely related to RtR and 3Pty reference point. There might also be some issues relating to retailer - retailer/3rd party service provider/content provider interactions that have not been addressed yet. It is likely that request for refinement and solutions for the reference points RtR and 3Pty will be issued, and that this process might give useful feedback to new versions of this document.

9.4 Service Composition and Federation

9.4.1 Composition

The folllowing issues have been identified:

- Impacts of composition at the communication level . Issues include the merging of established stream bindings and how CSMs may support this,
- Composition related feature sets,
- Composition supporting components: further definitions and examples,
- Composition and context management,
- Resolution of the relation of deployment and runtime issues that result when one or more services have a relation with a particular instance or implementation of another service or resource.

9.4.2 Federation

9.4.2.1 More than Two Retailers

The approach shown in Section 4.3.3 is for clarity and simplicity only. It only shows the setup of a federation between two retailers, because the situation where more than two retailers are involved, a number of issues must be further investigated.

However, the situation has been considered for the definitions given, and we believe the framework supports this situation. Specifically, a number of decisions must be made, for instance, whether every retailer must have a federation session (and thus communicate directly) with all other retailers in the federation, or whether a ‘chain’ of federation sessions, all based on the same contract, will suffice.

9.4.2.2 Management of Federation Session Profiles

Management of federation session profiles is to be considered and analyzed in depth as one of the most important aspects of the relationship between retailers. This includes: handling of management contexts through the RtR-RP, and consistency checking for and negotiation of management contexts corresponding to local service sessions participating in the overall service session.

9.4.2.3 One Retailer Hands-off the Service Execution to Another Service Provider

A special situation, usual in TINA where mobility is an intrinsic part of the architecture, should be considered in the next releases: one retailer hands-off the service execution to another retailer / service provider. This may be done for several reasons: congestion in his network, QoS not available, or the main one: the user is closer to the other retailer (attached to its supporting DPE).

In this main case, the access session is initiated with one retailer, but the service session is executed against other. This case has already been considered in the service composition framework. However, some functions (ancillary, management functions) can be still offered through the federation interface. For instance, those for accounting (to receive the final charge for the handed-off service), performance (to monitor the QoS provided to the user by the federated retailer), etc.

9.5 Mobility

Although the current service architecture provides a basic support for personal and session mobility as described in Section 8, some issues still need further work. They are summarized in the following list:

- Personalization: specification of subscription management components and procedures will provide the means for users to personalize their access and use of services. Work on this issue is on-going in the core team and PCS auxiliary project.
- Multi-retailer environment: the case where a service user or user's terminal moves to the domain of a retailer, different from the one providing him the service, needs to be considered. Both the core team and DOLMEN and PCS auxiliary projects are studying this problem and will provide solutions.
- Terminal mobility: [23] identified several possible solutions for the support of terminal mobility in TINA. They need to be developed further, and it should be decided which one(s) TINA will provide. Depending on the decision, the service architecture is expected to be affected to some extent. Input from DOLMEN and P608 auxiliary projects will be received on this issue.

9.6 Role of the Service Factory Component

In the event traces dealing with resume service and resume participation the service factory (SF) is employed. This is a slightly different but closely related role to that in service invocation, where the SF is responsible for ensuring the instantiation and the initialization of service objects. In the case of suspended service sessions or suspended participation, it is not clear whether the normal computational objects SSM and USM associated with the session actually exist, since it is a session that is suspended, not necessarily each implementation object supporting the session. Thus there is a role in the resume sequence for a ‘session manager’ that can take a reference to a suspended service session or user service session and instantiate and initialize the SSM and USM. This has been chosen to be the SF. This choice requires further refinement of the SF function and examination of what a suspend session has on the instantiated objects holding the session state.

9.7 Terminal capabilities

Mechanisms are needed to convey and manage (on-line) information about current terminal capabilities (and the possible changes to them) from the terminal to the USM/SSM in order to adjust the service profile, QoS, etc. It is likely that auxilary projects will contribute to this issue.

9.8 Service Interactions

How a USM could cooperate with a UA in order to solve possible service interaction problems (e.g., checks on the personal profiles) needs further investigation. However, these issues are highly service specific, and service interactions problems will be left for stakeholders to solve in a provider specific, service specific way, using specialized TINA components and interfaces.

9.9 Naming

A fundamental requirement of any information processing system is to be able to distinguish between instances of an entity type. The service architecture must therefore support a coherent framework for denoting, that is standing-in for, and referencing, that is pointing to, entity instances.

Denotation by means of identity is regarded as not tractable in the general case, but denotation by means of naming is tractable in controlled environments. Issues of name structure, naming authorities, name binding, time scales, location transparency, and location independence need to be resolved.

The referencing of entities and, in distributed environments, communicating with remote entities, generally requires some sort of reference resolution mechanism. Issues of aliasing, addressing, routing, and migration need to be resolved. Approaches to resolution mechanisms, such as directories, trading services, broadcasting, and local caching need to be discussed.

The primary function of naming, that is to support denotation or reference, is often informally overloaded to expose characteristics such as composition, ownership, description, and type. This form of name overloading needs to be discussed.

Existing systems support a number of naming architectures and name resolution mechanisms; federating with such systems is crucial to the success of the service architecture. Issues of name translation, security, and visibility need to be discussed.

10.Acknowledgement

The authors want to thank all authors of all versions of the service architecture during the life of the TINA core team. For the full list of contributors see page 165. Other members of the core-team during the last 4 and a half year have also contributed through discussions, comments etc.

This document has benefited from lots of suggestions and comments from many persons in the core team and in the TINA-C member companies. They have helped to improve the technical content and the presentation of this document.

The authors would like to thank particulary the external reviewers from the following companies that all provided useful comments:

- Alcatel: M. Mampay et al.;
  Bellcore: G. Brenner;
  BT: P.Loosemore;
  Deutche Telecom: C. Capellmann, T.Eckhardt, J.Fischer and U.Herzog;
  Fujitsu: F. Birch and C. South;
  KPN Research: Y.Lu et al.;
  Lucent Technologies: K. Kocan;
  NTT: H.Ishii and H. Kobayashi
  Sprint: A.Akram, M. Gettles and L.Matt;
  Telecom Italia: A. Limongiello;
  Telia Research: A.Lindgren et al.

Chelo Abarca
Alcatel Telecom, Madrid
Spain

Patrick Farley
BT
United Kingdom

Jan Forslöw
Ericsson Telecom
Sweden

Juan Carlos García
Telefónica
Spain

Takeo Hamada
Fujitsu Laboratories
Japan

Per Fly Hansen
Tele Danmark
Danmark

Stephanie Hogg
Telstra Corp.
Australia

Hiroshi Kamata
OKI
Japan

Lill Kristiansen
Telenor
Norway

Carlo A. Licciardi
CSELT
Italy

Harm Mulder
KPN
The Netherlands

Eiji Utsunomiya
KDD
Japan

Martin Yates
BT
United Kingdom

11 References

The TINA-C public documents may be acquired from the TINA-C public WWW page at:

http://www.tinac.com

The TINA-C internal documents may be acquired from the TINA-C internal WWW page at:

http://tinac.com:4070/root.html

The TINA-C web site provides a search engine. To find a particular document use the document number or the title of the document as search parameters.

TINA-C Documents

TINA-C Baseline documents

[1] *TINA Glossary of Terms,* Version 2.1, TINA-C, Jan. 1997; CT reviewed intermediate; public.
File: /u/tinac/97/integration/docs/glossary/v2.1/GLOSSARY.ps.
Authors: H. Mulder, All TINA-Consortium Participants,

[2] *Requirements Upon TINA-C Architecture,* Version 2.0,TINA-C, Feburary1995; TINA-C internal.
File: /u/tinac/94p2/viewable/requirements.ps.
Authors: D. Brown, S. Montesi.

[3] *TINA Business Model and Reference Points,* Version 4.0 , TINA-C, May1997; public.
File:/u/tinac/97/integration/viewable/bm_rp.ps.
Authors: M. Yates, W. Takita, R. Jansson, L. Demounem, H. Mulder.

[4] *Information Modeling Concepts,* TINA-C, April 1995; public.
File:/u/tinac/94p2/viewable/info.ps.
Authors: H. Christensen, E. Colban.

[5] *Computational Modelling Concepts,* Version 3.2, TINA-C, May1996; TINA-C internal.
File:/u/tinac/96/dpe/docs/computational_model/v3.2/cmc.ps.
Authors: T. Handegård, Many TINA-C Core Team Members

[6] .*TINA Object Definition Language Manual,* Version 2.3, TINA-C, July1996; TINA-C internal.
File:/u/tinac/96/dpe/viewable/odl_manual_v2.3.ps
Author: A. Parhar.

[7] *Service Architecture,* Version 4.0, TINA-C, Oct. 1996; public.
File: /u/tinac/96/services/viewable/sa96_v4.0/sa96.ps.
Authors: R. Minetti (ed.), C. Abarca, P. Farley, J. Forslöw, T. Hamada, P. F. Hansen, H. Hegeman, S. Hogg, H. Kamata, K. Kiwata, L. Kristiansen, M. Mampaey, H. Mulder, S. Pensivy, E. Utsunomiya, M. Yates.

[8] *TINA Network Resource Architecture,* Version 3.0, TINA-C, Febr. 1997; public.
File: /u/tinac/resources/viewable/nra_v3.0.ps.
Authors: F. Steegmans (ed.), C. Abarca, J. Forslow, T. Hamada, S. Hogg, H. J. Beom, D. S. Kim, H. Y. Lee, N. Natarajan.

Other Versions of the Service Architecture Document

[9] *Definition of Service Architecture,* Version1.0, TINA-C, December 1993; TINA-C internal.
File: /u/tinac/93arch/wab_db3.ps.
Authors: H. Berndt, D. Brown, M. Chapman, S. Fratini, H. Hammainen, R. Minerva, M. Nordin, K. Ohtsu, J. O'Neil, H. Yagi, D. Yun.

[10] *Service Architecture,* Version 2.0, TINA-C, March 1995; public.
File: /u/tinac/94p2/viewable/servarch.ps.
Authors: H.Berndt, C. Kim, S. Kim, H. Kobayashi, R. Minerva, K. Ohtsu, J. Pavón, F. Ruano, M. Wakano, H. Yagi.

[11] *Service Architecture Δ95,* TINA-C, April1996, TINA-C internal.
File:/u/tinac/95/services/95baselines/architecture/Versions/Ver1.0/architecture.ps.
Authors: C.Abarca, M. Bagley, H. Hegeman, H. Kamata, H. Kobayashi, M. Mampaey, R. Minetti, K. Moore, E. de Tournemire.

[12] *Service Architecture,* Version 4.1, TINA-C, January 1997, TINA-C internal.
File: /u/tinac/96/services/viewable/sa96dr_v4.1/sa96opd.ps.
Authors: P. F. Hansen, P. Farley (eds), C. Abarca, J. Forslöw, T. Hamada, P, H. Hegeman, S. Hogg, H. Kamata, K. Kiwata, L. Kristiansen, C. Licciardi, M. Mampaey, R. Minetti, H. Mulder, S. Pensivy, E. Utsunomiya, M. Yates.

Planned TINA-C Baseline Documents (relevant for Service Architecture)

[13] *The Ret Reference Point*, Version 0.3, TINA-C, 7th March, 1997 (interim version), TINA-C internal.
File:/u/tinac/97/integration/rfrs/RFR-96-01/interim/draft0.3/ret.bk
Authors: P. Farley, S. Hogg, L. Kristiansen, C. Licciardi, M. Mampaey, R. Minetti, S. Pensivy, C. Smith, E. Utsonomiya, M. Yates.

[14] *Service Component Specification.*

[15] *Developer's Guide to TINA*.

[16] DPE Architecture

Miscellaneous TINA-C Core Team Documents

[17] *Engineering Modeling Concepts (DPE Architecture),* TINA-C, Dec. 1994; TINA-C internal.
File: /u/tinac/94.p2/dpe/doc/eng/eng.ps.
Authors: P.Graubmann, W. Hwang, M. Kudela, K. MacKinnon, N. Mercouroff, N. Watanabe.

*[18] TINA Distributed Processing Environment (TINA-DPE), TINA-C, Dec. 1995; TINA-C internal.
File: /u/tinac/96/dpe/viewable/dpearch1.3.ps*
Authors: P.Leydekkers, K. MacKinnon, N. Mercouroff.

*[19] Ret Reference Point, Request for Refinements and Solutions, Version 2.0, TINA-C, August 1996; TINA-C internal.
File: /u/tinac/96/integration/rfrs/RFR-96-01/rfrs_ret.ps.*
Authors*: TINA-C Core Team.*

[20] *Service Deployment & Withdrawal,* TINA-C, December 1995; TINA-C internal.
File: /u/tinac/95/services/deploy/eng_note/bk_template.bk.ps.
Author: K. Moore.

[21] *Service Composition,* Version1.0, TINA-C, July1995; TINA-C internal.
File: /u/tinac/95/services/composition/Ver1.1/composit.ps.
Authors: M. Bagley, R. Gutierrez, H. Kobayashi.

[22] *Accounting Management Architecture,* Version1.2, TINA-C, March1996; TINA-C internal.
File: /u/tinac/95/resources/viewable/accounting.ps.
Author: T. Hamada.

[23] *Terminal Mobility*, Version 2.0, TINA-C, March 1997; TINA-C internal.
File: /u/tinac/97/resources/docs/mobility/v2.0/tm_v20.ps.
Authors: J. Hegeman, C. Abarca.

[24] *TINA naming framework*, Ver0.2(draft), December 1996, TINA-C internal.
File: /u/tinac/96/overall/docs/naming/reports/overall/Ver0.2/nameframe.ps.
Author: J. Forslow.

[25] *TINA Information Services & Resource Architecture***,** TINA-C, December 1996; TINA-C internal.
File:/u/tinac/96/services/docs/info_service/Report/Ver1.0/info.ps.
Authors: J. Forslow, L.Kristiansen, S. Pensivy.

Auxiliary Projects Documents

[26] *Initial Options for Evolving to TINA*, Deliverable 1, EURESCOM, February 1996. TINA-C and Eurescom internal.
file: /u/tinac/home_company_projects/auxiliary_projects/p508/EUtoTINA/deliverable/d1.ps.
Authors: EURESCOM Project EU-P508.

[27] *Migration Strategy for Interworking with Legacy Systems*, Deliverable 2, EURESCOM, February 1997. TINA-C and Eurescom internal.
file: /u/tinac/home_company_projects/auxiliary_projects/p508/EUtoTINA/deliverable/d2.ps.
Authors: EURESCOM Project EU-P508.

[28] *Demonstration of Information Browsing Based on Available Mobile Communications Technology*, DOLMEN TINA auxiliary project, August 96. TINA-C and DOLMEN internal.
File:/u/tinac/home_company_projects/auxiliary_projects/DOLMEN/dolmen1.ps
Authors: Raatikainen, Liljeberg, Helin.

[29] *TINA Data Management Framework*, France Telecom/CNET Auxiliary Project, November 1995. TINA-C internal.
File: /u/tinac/96/home_company_projects/auxiliary_projects/DataManagement/DMF.draft2.ps.
Author: Y. Lepetit.

[30] *Access and Service Session Control for VITAL v2 (parts A and B)*, ACTS Vital project, May 1997 TINA-C and Vital internal.
File: /u/tinac/home-companies/auxillary_projects/VITAL/workshop_9704/session.zip
Authors: P.Hellemans and P. de Ceuleners.

International Standards Documents

CCITT/ITU-T documents

Intelligent Networks

[31] ITU-T, Draft Recommendation Q.1221 *Introduction to Intelligent Network Capability Set 2*, 1995.

TMN

[32] CCITT Recommendation M.3010, *Principles for a Telecommunications Management Network*, 1992.

B-ISDN

[33] ITU-T, Recommendation Q.931 *ISDN User Network Interface Layer 3 Specification for Basic Call/Connection Contro*l, 1995.

[34] ITU-T, Recommendation Q.2931 *B-ISDN User Network Interface Layer 3 Specification for Basic Call/Connection Control*, 1995.

[35] ITU-T, Recommendation Q.2951, Q.2955 & Q.2957 *B-ISDN User Network Interface Layer 3 Specification for Supplementary Services*, 1995.

OSI Management

[36] ITU-T, Recommendation X.501 *Information Technology OSI The Directory: The Models*, 1993.

[37] ITU-T, Recommendation X.520 *Information Technology OSI The Directory: Selected Attribute Types*, 1993.

[38] ITU-T, Recommendation X.521 *Information Technology OSI The Directory: Selected Object Classes*, 1993.

Other

[39] ITU-T Recommendation F.850, *Principles of Universal Personal Telecommunication (UPT)*, March 1993

ISO/IEC documents

Open Distributed Processing (ODP)

[40] ISO//IEC 10746-2 / ITU-T Draft Recommendation X.902, *Basic Reference Model of Open Distributed Processing - Part 2: Foundations,* International Organization for Standardization and International Electrotechnical Committee, 1995.

IETF documents

[41] IETF Internet Assigned Numbers Authority, *Well-known Port Numbers* http://www.isi.edu:80/in-notes/iana/assignments/port-numbers.

[42] IETF, RFC 1738 *Uniform Resource Locators*, 1994. http://info.internet..isi.edu:80/in-notes/rfc/files/rfc1738.txt.

[43] IETF, RFC 1808 *Relative Uniform Resource Locators*, 1995. http://info.internet..isi.edu:80/in-notes/rfc/files/rfc1808.txt.

[44] IETF, Hypertext Transfer Protocol -- HTTP/1.1

OMG documents

[45] OMG, *Trading Object Service*, OMG RFP5 Submission, orbos/96-07-08

[46] OMG, *The Common Object Request Broker Architecture and Specification*, Ver2.0, July 1995

Network Management Forum documents

[47] Network Management Forum, SMART, *A Service Management Business Process Model*, Nov. 1994.

[48] NMF Component sets: CORBA/CMIP/SNMP Interworking, Draft, NMF, CS342, 1996.

X/Open

[49] Inter-Domain Management Specifications: Specification Translation (JIDM), X/Open, 1995.

Other Documents

RACE Documents

[50] RACE Project R.1093 (ROSA) Deliverable 93/BTL/DNR/DS/A/005/b1, RACE, *The ROSA Architecture*, Release Two, Version 2, RACE, May 1992.

Books

[51] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, *Object-Oriented Modeling and Design*, Prentice Hall, Englewood CliffsN.J., 1991.

12.Acronyms

3Pty	Third Party service provider (business relationship)
3Pty-RP	Third Party service provider inter-domain reference point
anonUA	anonymous User Agent
as-UAP	access session related User APplication
AS	Access Session
ASEP	Access Session End Point
ASR	Access Session Relationship
B-ISDN	Broadband-ISDN
Bkr	Broker (business relationship)
Bkr-RP	Broker inter-domain Reference Point
CompD_USS	Composer Domain Usage Service Session
CompSR	Composing Session Relationship
CompUSM	Composer Usage Session Manager
CM	Connection Management
CMC	Computational Modeling Concepts
CMIP	Common Management Information Protocol
CNM	Customer Network Management
CO	Computational Object
COG	Computational Object Group
ConS	Connectivity Service (business relationship)
ConS-RP	Connectivity Service inter-domain Reference Point
CORBA	Common Object Request Broker Architecture
CPE	Customer Premises Equipment
CPN	Customer Premises Network
CS	Communication Session
CSLN	Client-Server Layer Network (business relationship)
CSLN-RP	Client-Server Layer Network inter-domain reference point
CSK	Common Shared Knowledge
CSM	Communication Session Manager
CSMF	Communication Session Manager Factory
CtrSR	Control Session Relationship
CUG	Closed User Group
D_AS	Domain Access Session
D_USS	Domain Usage Service Session
D_USS_Binding	Domain Usage Service Session Binding
DAVIC	Digital Audio-Visual Council
DIT	Directory Information Tree
DN	Distinguished Name
DPE	Distributed Processing Environment
eCO	Engineering Computational Object
EML	Network Element Management Layer
FA	Federation Agent
FCAPS	Fault, Configuration, Accounting, Performance, Security

FEP	Flow End Point
FS	Feature Set
FSM	Finite State Machine
FTP	File Transfer Protocol
GDMO	General Definition of Managed Objects
GRM	General Relationship Model
GSC	Global Session Control
GSEP	Generic Session End Point
GSS	Global Service Segment
IETF	Internet Engineering Task Force
I/f	Interface
IA	Initial Agent
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IN	Intelligent Network
IO	Information Object
IOR	Interface Object Reference
IP	Internet Protocol
IR	Interface Reference
IRM	Information Resource Manager
ISDN	Integrated Services Digital Network
kTN	kernel Transport Network
LCG	Logical Connection Graph
LNFed	Layer Network Federation (business relationship)
LNFed-RP	Layer Network Federation inter-domain Reference Point
LSG	Local Session Graph
MgmtCtxt	Management Context
MUSC	Member Usage Session Control
MUSM	Member Usage Session Manager
MUSS	Member Usage Service Segment
namedUA	named User Agent
NAP	Network Access Point
NCCE	Native Computing and Communications Environment
NCG	Nodal Connection Graph
NEL	Network Element Layer
NFC	Network Flow Connection
NFEP	Network Flow EndPoint
NML	Network Management Layer
NRA	Network Resource Architecture
NRIM	Network Resource Information Model
NS	Naming Server
ODL	Object Definition Language
ODP	Open Distributed Processing
OMG	Object Management Group
OMT	Object Modelling Technique

OO	Object Oriented
OOM	Object Oriented Modeling
OSI	Open Systems Interconnection
OSR	Ownership Session Relationship
PA	Provider Agent
PCG	Physical Connection Graph
PD_AS	Provider Domain Access Session
PD_USS	Provider Domain Usage Service Session
PeerA	Peer Agent
PeerD_AS	Peer Domain Access Session
PeerD_USS	Peer Domain Usage Service Session
PeerSM	Peer Session Member
PeerSMG	Peer Session Member Group
PeerUSM	Peer Usage Session Manager
POTS	Plain Old Telephone Service
PartySM	Party Session Member
PartySMG	Party Session Member Group
PSR	Permission Session Relationship
PSS	Provider Service Session
QoS	Quality of Service
Rel	Relationship Object
Repo	Repository
Ret	Retailer (business relationship)
Ret-RP	Retailer inter-domain Reference Point
RFR/S	Request For Refinements/Solutions
RP	Reference Point
RPSR	Read Permission Session Relationship
RSM	Resource Session Member
RSMG	Resource Session Member Group
RtR	Retailer to Retailer inter-domain reference point
SA	Subscription Agent
SAG	Subscription Assignment Group
SBSR	Stream Binding Session Relationship
SBSRG	Stream Binding Session Relationship Group
SC	Service Component
SCS	Service Component Specifications
SF	Service Factory
SFC	Stream Flow Connection
SFEP	Stream Flow End Point
SGI	Session Graph Interface
ShSR	Shared Session Relationship
SI(1)	Stream Interface (informational entity)
SI(2)	Stream Interface (computational entity)
SIB	Service Independant Building block
SILC	Service Instance Life Cycle

SLC	Service Life Cycle
SM	Session Member
SMG	Session Member Group
SML	Service Mangement Layer
SNMP	Simple Network Management Protocol
SOE	Service Offer Evaluator
SP	Service Provider
SR	Session Relationship
SRG	Session Relationship Group
SRgs	Subscription Registrar
SS	Service Session
ss-UAP	service session related User APplication
SSC	Service Support Component
SSEP	Specific Session End Point
SSG	Service Session Graph
SSM	Service Session Manager
STH	Service Template Handler
SubM	Subscription Manager
SubSR	Subsidiary Session Relationship
ToM	Terms of Management
TCon	Terminal Connection (business relationship)
TCon-RP	Terminal Connection inter-domain reference point
TCSM	Terminal Communication Session Manager
TFC	Terminal Flow Connectiom
TINA [-C]	Telecommunications Information Networking Architecture [Consortium]
TMN	Telecommunications Management Network
TPSP	Third Party Service Provider
TSA	Terminal Service Adaptor
Tr	Trader
UA	User Agent
UA-GM	User Agent Group Manager
UAP	User Application
UCtx	User Context
UD_AS	User Domain Access Session
UD_USS	User Domain Usage Service Session
UDSEP	User Domain Session End Point
UPrf	User Profile
UPT	Universal Personal Telecommunication
URL	Uniform Resource Locator
URN	Uniform Resource Name
USC	User Session Control
USCI	User/Service session Control Interface
USCM	Universal Service Component Model
USM	User Service Session Manager
USS	Usage Service Session

VoD	Video On Demand
WPSR	Write Permission Session Relationship
WWW	World Wide Web

List of Figures

List of Tables

Contributors

The Service Architecture document contains the results of the work done at the TINA-C Core Team during its life until the time of this writing. Therefore, contributors to this document are not only the main authors listed in the cover page, but also the authors of previous versions of the TINA Service Architecture, produced in December 1993, in March 1995, April 1996, October1996 and JAnuary 1997.

The complete list of contributors with their affiliation (at the time of their contribution) is given below:

Chelo Abarca	Alcatel, Spain
Mark Bagley	BT, United Kingdom
Hendrik Berndt	Deutsche Telekom, Germany
Dave K. Brown	NEC America, USA
Martin D. Chapman	BT, United Kingdom
Eric de Tournemire	France Télécom, France
Patrick Farley	BT, United Kingdom
Stephen S. Fratini	Bellcore, USA
Jan Forslöw	Ericsson, Sweden
Nicola Gatti	Telecom Italia, Italy
Juan Carlos García	Telefónica, Spain
Takeo Hamada	Fujitsu, Japan
Heikki Hammainen	Nokia, Finland
Per Fly Hansen	Tele Danmark, Denmark
Hans Hegeman	KPN, The Netherlands
Matthias Horrer	Alcatel, Germany
Stephanie Hogg	Telstra, Australia
Hiroshi Kamata	OKI, Japan
Cheol Koo Kim	Samsung Electronics, Korea
Sang Kyung Kim	Korea Telecom, Korea
Kazuhiro Kiwata	NTT, Japan
Hidetsugu Kobayashi	NTT, Japan

Lill Kristiansen	Telenor, Norway
Carlo Licciardi	CSELT, Italy
Marcel Mampaey	Alcatel, Belgium
Roberto Minerva	CSELT, Italy
Roberto Minetti	CSELT, Italy
Kathryn Moore	BT, United Kingdom
Harm Mulder	KPN, The Netherlands
Mats Nordin	Telia, Sweden
Kazuyuki Ohtsu	Hitachi, Japan
Joseph O’Neil	AT&T, USA
Juán Pavón	Alcatel, Spain
Stéphane Pensivy	France Télécom, France
Fernando Ruano	Telefónica, Spain
Eiji Utsunomiya	KDD, Japan
Masaki Wakano	NTT, Japan
Hikaru Yagi	KDD, Japan
Martin J. Yates	BT, United Kingdom
Dong Sik Yun	Korea Telecom, Korea

The following contributors have also been editors of the Service Architecture document:

Chelo Abarca	Alcatel, Spain
Hendrik Berndt	Deutsche Telekom, Germany
Martin D. Chapman	BT, United Kingdom
Patrick Farley	BT, United Kingdom
Per Fly Hansen	TeleDanmark, Denmark
Hidetsugu Kobayashi	NTT, Japan
Roberto Minerva	CSELT, Italy
Roberto Minetti	CSELT, Italy