Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FACSIMILE: **+44 1223 359779**
COMPANY No: **2300071**
IINTERNET: **apm@ansa.co.uk**

ANSA Phase III

An Overview of the Description and Distribution of Services

John A Bull

Abstract

The benefits of open trade of products and services in a world wide context are well recognised. The challenge for information technology is to enable open trade of electronic information and electronic services over world wide networks. In general, this document is concerned with how to promote electronic business. It considers two facets of the problem: how to store information about services in a distributed and federated way such that providers can offer services and users can locate them; and how this information can be used to manage services throughout their lifecycle, from design to deletion.

The document provides a focus for ANSA Phase III Workprogramme activities, and proposes a number of issues to consider, tasks to perform, and questions to answer.

APM.1342.00.02 **Draft** Fri, Nov 25, 1994
Request for comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

Contents

1 Introduction

This document sets out the foundations for the ANSA Phase III work on "Federation and Lifecycle" as specified in [APM.1275]. It provides a technical focus for the whole of workpackage C1 (Federation and Lifecycle), and parts of workpackage C4 (Applications Programmer's Interface for distributed interactive multi-media). It also has a strong association with workpackage C3 (Engineering for distributed interactive multi-media).

The document is directed towards ANSA team members and sponsors. It is a living document and will be modified and revised as work progresses, particularly as earlier work completes, and requirements for later work are better understood.

This chapter presents the business and technical objectives of the work.

1.1 Business case

1.1.1 Considerable time, effort and money has been expended in promoting open trade; for example, throughout Europe with the creation of the EEC, throughout North America through NAFTA, and world wide through GATT. The business case for this is already well understood and accepted. Business can benefit further by open trade of electronic information and electronic services in a world wide network. The impetus is much the same as for non-electronic open trade, but with electronic interchange adding timeliness and efficiency. It also becomes possible to integrate services to support business endeavours that would otherwise have been impractical; for example, where a supplier is remote from the customer.

1.1.2 A distributed pool of service information (a Service Information Repository: SIR) is needed as a channel through which potential servers and clients may enter into trade agreements. The technical challenge is to make this repository available, its internal structure and content accessible, and its content comprehensive. Meeting this challenge would create significant benefits:

- Productivity: Dynamic re-use and sharing of information and services through repositories would improve service interoperability and service creation. This would improve productivity, both directly in services offered, and indirectly through the organisational structure reflected by them.

- Planning: The repository would reflect, and thus assist in the planning and evolution of the organisation's information technology. This would further assists when planning and evolving the business itself.

- Control: Automation of changes in the repository, to produce changes in the system, would allow greater control over the information base of the organisation. This would ease control of the business itself.

1.1.3 One of the greatest business challenges is to manage change. New opportunities arise, businesses merge and divide, and processes evolve. Consequently, electronic information and services are also continually evolving. The technical challenge is to allow dynamic integration and withdrawal of information and services, and to permit the exploitation of new technologies. A repository that is able to assist in the management of the full *life cycle* of all electronic system components would create significant further benefits:

- Evolution: The evolution of businesses would not be limited by the current essential electronic systems that support them. Moreover, a flexible repository that is sympathetic to the life cycle of business information and services would assist the process of change.

- Efficiency: More efficient access to, and evolution of the business information base would allow faster response to changing market conditions and a shorter time to market.

1.1.4 Organisations have their own standards, conventions, and processes, which create domains of interest within administrative boundaries. Inter-organisational trade depends on temporary bridges being built between organisations in response to business priorities and opportunities, thereby creating transient virtual companies. This is not controlled by any prior grand plan, nor central authority. The technical challenge is to permit similar transient lifecycles within the inter-organisational systems of electronic trading and interaction.

1.2 Technical challenge

1.2.1 The technical challenge is:

- to create and deploy scaleable, distributed, federated repositories;

- to structure information within those repositories to allow service trading;

- to allow for dynamic introduction, withdrawal and re-use of system elements.

1.2.2 The repository contains information concerning services. It needs to build on existing trading services and extend them:

- to store more complex information about services, including non-functional information, such as Quality of Service attributes and cost/price data;

- to describe services proposed but otherwise unavailable;

- to accommodate high level, application oriented descriptions that could hide changes in information structure, representation, protocols, etc, thus allowing services to be used in many unpredictable contexts;

- to describe itself, thus to become part of the system, and not to exist outside it.

1.2.3 It must be possible to distribute a repository within an administrative domain, and for a repository of one domain to refer to repository information of another. This allows any client to negotiate access to any service, anywhere. This degree of open trade requires:

- service descriptions that enable automatic generation (or identification) of gateways and interceptors as needed for access beyond domain boundaries;

- interfaces to be described such that they could be matched to similar interfaces of other domains;

- infrastructure services, such as naming or relocation services, to be described such that they could be matched to similar services in other domains;

- server offers and client requests to be described such that they could be matched, respectively, to client requests and server offers of other domains.

1.2.4 It is unlikely that a match between systems outside of a closed network could be exact, so it is necessary to specify minimum criteria which a service provider must satisfy. In addition to functional requirements, such as operations and transport protocols, the aim is to factor non-functional requirements, such as quality of service, cost, rights and obligations, into negotiations.

1.3 Base Position

1.3.1 The repository has to be developed within the context of products that exist, or are very likely to exist. The baseline is:

- OMG: [OMG 92]
  - CORBA and Object Services Infrastructure;
  - Naming;
  - Object management services;
  - Repository;
  - Life cycle management;
  - "Bridge" management (Universal Network Objects - UNO);
  - Lightweight, versus functionally rich ORBs;
  - Security.

- ANSA:
  - Trader; [APM.1005] & [APM.1140]
  - Federation; [APM.1280]
  - Information Model for Federation; [APM.1229]
  - Property Repository; [APM.1177]
  - Naming; [APM.1003]
  - Matchmaking Services; [APM.1280]
  - Interception;
  - ANSAware, particularly PREP C.

- ODP: [ISO 10746 Parts 2 and 3]
  - Naming;
  - Object Management Services;
  - Trader;
  - Interception;
  - Security.

- OSF/DCE: [OSF 92]
  - Naming;
  - Directory services;
  - Object Management Services.

- Microsoft Common Object Model - COM:
  - Naming;
  - Object Management Services;
  - Interception.

1.4 Goals

1.4.1 To develop a federation and lifecycle model. In particular:

- to provide federated object management through the integration of trading, repository and life cycle services;

- to develop ORB interoperability within the base context given above.

The first generation of OMG object services for life cycle and naming have been defined. Extensible schemes for federated trading and object management have proven to be more elusive. Current proposals suffer from technical problems that arise largely from the lack of a coherent architecture. [APM.1275 - 7.4.2.1]

1.4.2 To offer clean, high level infrastructure interfaces to application designers through *Abstract and Automate* principles [APM.1275 - 7.4.2.4]. In particular:

- to develop technology to support extensible repository interfaces
  - (eg, conformance based type checking);
- to develop technology to support an extensible repository implementation
  - (eg, an Abstract Syntax Tree);
- to track developments in multi-media engineering, and respond as necessary. [APM.1275 - 7.4.2.3]

1.4.3 To define an architecture which:

- follows ANSA principles;
- offers maximum compatibility with RM-ODP;
- offers maximum alignment with CORBA, and CORBA related object services;
- clearly separates interfaces from implementation;
- maximises the scope for federation, scaling, and flexibility.

1.4.4 To define a programming model which:

- is type safe, with maximum early checking. This will involve deciding:
  - what *semantic* type checking means;
  - how to manage type descriptions;
- provides selective domain transparencies. This will involve deciding:
  - how to split, merge, and bridge type repositories across domains;
  - the balance between application and infrastructure bridging;
- is extensible. This implies a programming model that can:
  - always allow data structures with dynamic bounds;
  - tolerate alternative concepts and programming styles.

1.4.5 To define an engineering model that provides for linking:

- property repositories and traders;
- type checkers and type/interface repositories;
- ORBs with dynamic skeleton and invocation interfaces (DSI & DII);
- object life cycle functions, including:
  - object factories;
  - garbage collection;
  - relocation facilities;
  - migrating objects;
  - groups/replication management;
- gateways and interceptors.

2 Description of Services

Clients may need to negotiate services from anywhere, including from domains with dissimilar information structures and conceptual models. This chapter considers the problem of matching client requirements to service offers, when an exact match between systems outside of a closed network is unlikely.

The work rests largely on a presumption against matching service requests and offers by prior assertion; that is, against direct checks for equality of pre-defined service elements. Direct equality would require total semantics being conveyed in element names; at worst, in a single service name. This would:

- create scope for error, when prior assertions are found not exactly to match;
- not scale, since global agreement on service semantics would not be achieved;
- be non-trivial to federate, since common knowledge would be presumed;
- be non-trivial to evolve, since version information would have to be integrated with names (otherwise adding version control would be an ad-hoc deviation from the prior assertion model).

Overall, an automatic process is needed for negotiating agreement between service offers and client requests based on criteria for necessity and sufficiency. In addition to allowing functional requirements, such as operations and transport protocols to be agreed, this would permit non-functional requirements, such as quality of service, cost, rights and obligations, to be introduced into negotiations.

The following sections list the issues to be considered:

2.1 Determine service negotiation principles

2.1.1 It has to be decided what, where and when client request against server offer negotiation should take place. Possible extremes are:

- that all services be allocated unique identifiers prior to on-line binding, so that the binding operation just involves selecting from identifier lists;

- that all negotiation takes place on-line when a client seeks a service.

Both of these extremes present problems, and the preferred position is likely to lie somewhere between. The primary task is to uncover the underlying concepts and principles that would point to the correct balance for any particular application. This will require detailed study of base components, with possible prototyping, and plausible scenarios within which concepts could be tested.

2.1.2 The outcome of this study is important to the timing and positioning of standards, which could range from anarchy (in keeping with Internet growth), through de-facto standards, to formal standards (eg, as would be needed early for allocation of unique identifiers). Standards are always needed, but there is an illusion to the contrary when they are deferred, or positioned elsewhere.

2.2 Examine the role of tools systems

There is a need to consider what part tools should play in any pre-binding negotiation, possibly to produce some partially bound service (but not going as far

as unique service identifiers). This would lead to considering the distribution and federation of software engineering tools systems as part of life cycle support.

2.3 Produce and integrate an Abstract Syntax Tree (AST)

An AST is a language that describes services using abstract data types; that is, neutral to language source representation and platform infrastructure. It describes the structure of an interface, and possibly its implementation, thus appearing as an enhanced integration of IDL and PREP C. Further work should consider:

- Direct representation of AST concepts (as AST structures) in source programs;
- use of ASTs in PREP C type matching, together with appropriate algorithms;
- tools based approaches for generating descriptions;
- C++ influence.

2.4 Develop type inferencing extension rules

When deciding whether a service requirement can match a service offer, type compatibility is necessary but not sufficient. The semantic *meaning* associated with the service has to be agreed. Possible ways to do this could be:

- explicitly defined algorithms, accessed by call and return;
- parametric service descriptions though a suitable language,
  - effectively providing a semantic equivalent of ASN data structuring;
- implicit casting for service descriptions,
  - equivalent to implicit casting of variables within programming languages, for example, such as integer to real.

2.5 Define interpretation of primitive types and constructors

This issue concerns the definition of the primitive data types which form the basis of all other data structures. There are two ways of defining primitives:

- by representation, such as: *An int is sign (bit zero) plus .... etc*;
- through operations, such as: *An int + int = > int.*

Some primitive representation is always necessary, but it has to be decided whether the base primitive set should be very small, or relatively wide. As a minimum, the relationship between different domains has to be defined, such as between ODP, CORBA IDL, and ANSA.

2.6 Compare and contrast inheritance and explicit sub-typing

There has been a long running debate about the relationship between the concepts of inheritance and sub-typing. This should be explored further, particularly in the context of life cycle management. The answers may be found in various papers already published, but these are highly technical. There is a need to:

- explain the practical significance of the issue;
- articulate the arguments to non language specialists.

3 Distribution of Services

The objective of this work is to consider how to match concepts and services at a domain boundary. Concern will range from real data items, such as bank accounts, to engineering concepts, such as methods for service relocation and naming. This work considers the following general areas:

- Invocation Reference management;
- Gateway management;
- Resource management issues relevant to gateways
  - including garbage collection;
- Role in binding (QoS issues);
- Optimisation;
- Monitor Binding and Multi-media work to check for any impact;
- The requirements of the repository;
- Federation of the service information repository;
- To consider how to enforce separation at a domain boundary
  - such as to partition a service information repository;
- To considers how to build a single domain from hitherto separate domains;
- To consider how to monitor a domain boundary.

Specifically, the task areas are:

1. To experiment with management of interception and the interception lifecycle;
2. To enhance present gateway design and implementation;
3. To move towards automatic generation of interface specific gateways;
4. To decide how to structure, create and deploy repositories.

3.1 Life cycle and management of interception

3.1.1 The aim of this work (which must be well advanced before any of the other work can begin) is to:

- Set up an interceptor between two domains;
- Decide responsibilities for interception, and gateway creation;
- Define relationships with:
  - trading;
  - factories;
  - domain managers;
  - previously established gateways;
  - clients and servers in a client/server model;
- Decide what translation activity is needed in gateways;
- Determine what information is needed from the service information repository
  - in order to deal with any of the above.

3.1.2 Federation involves linking domain services across domain boundaries, such as:

- naming;
- remuneration;
- domain management services;
- communication services.

Gateways have to be set up dynamically to handle transient links between these services. Of particular importance, will be to decide how Invocation References can be passed between domains that handle them differently.

3.2 Enhanced gateway implementation

The aim of this work is to consider the feasibility of engineering different models of federation. Some immediate prototyping will be important. Following paragraphs list components for experimentation:

3.2.1 Stubs for gateways
Data-driven marshalling and un-marshalling of invocation arguments:

At present the stubs are generated from the IDL definition of the interface and are static. Making the stubs data-driven is a step in the direction of generic gateways. Remote Procedure Call stays the same. The descriptions of the data types (derived from IDL of the interface) is passed as additional information.

3.2.2 Gateways for multiple servers:

At present a gateway acts as a front end to a single server. A single gateway could support connection to many servers of the same type.

3.2.3 Multi-threaded gateways:

In the current implementation [APM.1303], a process waits until a reply is received from one invocation, before servicing another. In a multi-threaded implementation it may be possible to service the next invocation in parallel with the previous one, but at the expense of increasing complexity of the gateway.

3.2.4 Error handling strategies in various circumstances:

- Translation of error messages across domain boundaries;
- Errors occuring in gateways:
  - eg, time out of internal gateway connections.

3.2.5 Gateway distribution:

A single logical gateway will consist of two physical gateways, one in each domain (with a possibility of it being null). Socket implementation, rather than pipes, will be considered, to allow two *half* gateways to be physically distributed.

3.2.6 Generic gateways:

- Gateways are currently interface type specific. Dynamic gateways generation based on supplied type information should be considered, such as:
  - Orbix: Dynamic Stub Interface (DSI) and
    Dynamic Invocation Interface (DII); [IONA 93a & 93b]
  - Other data driven gateways;
- Gateways as locations for monitoring.

3.3 Gateway tools to support automation

3.3.1 The above enhancements to the gateway implementation are not all necessary before work on gateway tools can start. Possible options to consider for automatic generation of interface specific gateways are:

- Generic stub compiler (GSC); [ARM 94]
- AST and DPL stub generators. [APM.1314]

3.3.2 The interception study is focused primarily on the engineering. However, there will be some discussion of Enterprise and Information concepts that span domain boundaries; for example, contract negotiation.

3.4 The Service Information Repository

This work is concerned with the creation and deployment of a repository. The repository design depends on requirements that emerge from the service definition and federation tasks. Hence it follows after other activities. There are a number of aspects to it, as listed in following paragraphs:

3.4.1 What to use as a repository

At present, the repository is based on relational database technology. However, at some time the use of object-oriented databases may be considered.

3.4.2 What to store in the repository

The repository will need to store data structures that specify services, or service groups [APM.1229].

3.4.3 Distribution of the repository within a domain

A repository cannot exist as a single co-located entity. It will be distributed. It has to be determined how the component parts co-operate. This is a specific part of a more general concern about the role of distributed databases within object-based distributed architectures.

3.4.4 Federation of repositories between domains

Repositories cannot exist outside the system; they have to be within it and provide for themselves. A problem to consider is how repositories describe, and generate gateways and interceptors for themselves.

3.4.5 Relationship to OMG (and other) repositories

The repository must be positioned with respect to other repository designs and implementations, such as from OMG, and with respect to other services, such as: Object Property services; Object Query services; Object Relationship services; Event services; Persistence services; Life Cycle and Naming services; etc. Definition of repository elements must be such that they will integrate effectively, and will operate effectively within a federation of repositories.

4 Issues

This chapter will list specific issues raised and resolved.

5 Diary

This chapter will report on progress, and changes in direction or detail.

6 References

[APM.1003.01]	The ANSA Naming Model
[APM.1005.01]	The ANSA Model for Trading and Federation
[APM.1138.01]	Remote Database Queries in Open Distributed Systems
[APM.1140.00.09]	A Designers Introduction to Trading
[APM.1142.00.02]	A Model of Interception
[APM.1162.01]	Data Management for an Enhanced Trader
[APM.1177.00.17]	The Property Repository
[APM.1193.01]	Federation Manifesto
[APM.1229.00.10]	Information Model for Federation
[APM.1275.02]	1994 - 1996 ANSA Workplan
[APM.1280.00.04]	F3 Overview (Synopsis of the second federation deliverable)
[APM.1303.00.01]	ORBIX-ANSAware Gateway Design and Implementation
[APM.1314.01]	The ANSA Binding Model
[ARM 94]	The ANSAware 4.1 manual set
[IONA 93a]	Orbix: Programmer's Guide, Iona Technologies Ltd., Dublin, Republic of Ireland
[IONA 93b]	Orbix: Advanced Programmer's Guide, Iona Technologies Ltd., Dublin, Republic of Ireland
[ISA RC.101]	Trading in the Five Projections
[ISA RC.358.03]	Types and Projections
[ISO 10746]	Basic Reference Model of Open Distributed Processing
[OMG 92]	The Common Object Request Broker: Architecture and Specification Document Number 91.12.1, Object Management Group and X/Open
[OSF 92]	OSF, DCE Application Development Guide, Open Software Foundation, 11 Cambridge Centre, Cambridge, MA 02142, USA