Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Micro-scenarios for federation

Gomer Thomas, editor

Abstract

This document is a collection of *micro-scenarios*, or examples of situations which are likely to create obstacles to successful interoperation in federated systems. Each micro-scenario focuses on a single aspect of interaction. A wide spectrum of interaction aspects is included, such as accounting and billing, security, dependability, naming, IDL, type systems, transaction models, trading, protocols, transparencies, and many more. The original motivation for producing this document was to have a set of test cases for the internal use of the ANSA team in developing architectural principles for federation. As such, it is a working document. Hopefully sponsors will find these examples useful in their own right, to enhance knowledge of the kinds of problems which arise in federated systems, and hopefully sponsors will be able to contribute additional examples from their own experiences.

APM.1095.00.03 **Draft** 17 August 1995

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

Micro-scenarios for federation

Micro-scenarios for federation

Gomer Thomas, editor

APM.1095.00.03

17 August 1995

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK	(01223) 515010
INTERNATIONAL	+44 1223 515010
FAX	+44 1223 359779
E-MAIL	apm@ansa.co.uk

Contents

1 Introduction

This document contains a collection of micro-scenarios, or limited examples of the kinds of problems which can arise in federated systems. The reason they are called *micro*-scenarios is that each example focuses on a single aspect of interaction.

The original motivation for producing this document was to provide a set of test cases for the internal use of the ANSA team to assist with formulating and evaluating federation principles and a federation architecture. The philosophy is that any truly useful framework for federation must be able to deal with these kinds of situations which arise is real applications.

This current collection of micro-scenarios is by no means exhaustive or final. Hopefully it will be expanded as time goes on by the input of ANSA team members and ANSA sponsors.

The micro-scenarios in this document are grouped according to the following areas:

Management/administration/contractual considerations

- Accounting and billing
- Auditing/monitoring
- Security
  - Authorization
  - IDs and authentication
  - Encryption
- Privacy
- Performance
  - Response time
  - Priority
  - Deadline
  - Throughput
- Dependability
- Availability
- Other quality of service features

Semantic considerations

- Name assignments
- Information models
  - Representation mechanisms
  - Model content

- Operation semantics
- Transaction models
- Control of interaction
- Conversation type (i.e., what operations can be invoked in what order(s) for a meaningful conversation -- e.g., an SQL cursor cannot be “fetched” until it is opened, and an SQL positioned delete cannot be executed until a row has been fetched.)
- Trading classification/context
- Trader federation
- Exception handling

Mechanics considerations

- Interaction model (e.g. stateless server vs. stateful server)
- Interconnection technology (media, protocols, addressing)
- RPC protocol (message encodings)
- Name syntax
- Interface type systems
  - IDL
  - Conformance rules
  - Type hierarchy
- Transparencies (existence and/or mechanisms)
  - Location
  - Relocation
  - Replication
- Interface references and invocations (engineering viewpoint)
- Resource allocation policies

Transparencies

- Location
- Relocation
- Replication and Partitioning

Suggestions for additional areas are invited, as well as suggestions for additional micro-scenarios within these areas. It would be especially useful to get additional micro-scenarios in those areas where coverage is now sparse or non-existent.

Some of the micro-scenarios, in fact probably most of them, do not fit cleanly within just one area. Life is like that. However, it seems useful to have a rough grouping by areas, to assess coverage of all the major issues.

When the smoke clears away from analysis of these micro-scenarios in the context of federation principles, there are likely to be several different categories of “solution” to the situations posed in the micro-scenarios:

1. Some may be regarded as application level problems which have nothing to do with general federation principles or architecture.

2. Some may be regarded as problems which need to be solved at the application level, but for which general principles can be formulated to recommend how applications should solve them to promote maximal interoperability in a federation.
3. Some may show the need for certain generic federation services (e.g., gateways, encapsulators, integration services), and specifications can be developed for the services.
4. Some may show the need for certain general architectural principles to support federation, and these principles can be promulgated.

2 Micro-Scenarios for Management, Administrative and Contract Issues

2.1 Accounting and billing

The following are some commonly used accounting and billing schemes. What happens when these are used together in various combinations by a client accessing a server which in turn accesses other servers on behalf of the client?

- Server keeps track of resource utilization and issues bills to users at regular intervals (e.g., monthly), where each client node is automatically identified with a particular user. (This is sometimes used by computer centers with hard-wired terminals, also used by telephone companies for many types of services -- e.g., long distance calls.)
- Same situation, but user is identified and authenticated in some manner, independently of the node from which requests are made. (This is the most common scheme used by computer centers, also used by telephone companies for credit card calls.)
- User subscribes to service at fixed rate per time period and has unlimited usage during that time period -- used by some computer centers, and by many telephone companies for “local” service, also for certain services such as call waiting, busy/no-answer message recording, call forwarding. User may be identified by node or by some node-independent manner.
- User physically pays for usage at the time of usage -- e.g., coin telephones, debit card telephones.
- In some cases the above arrangements are by explicit contract between the user and service provider. For example, one orders phone service from the telephone company (and from a specific long distance carrier in the USA). In other cases the contract is implicit in the use of the service. For example, in USA many private companies provide public telephones. These companies will accept your telephone company credit card, and then they will bill you for the call through your telephone company. The bill from the telephone company will contain a separate page for each such company you have used during the month, and the amount for each company is added to your overall bill. You pay one check to your telphone company, and it disburses the appropriate amounts to these other companies. In many cases they are companies you have never heard of (and they usually charge much more than the regular phone company, taking advantage of the fact that you are a captive audience -- you need to make the call, and they are the only public phone in sight).
- In some of the above situations the services seen by the end user may be a composite of services provided by multiple service providers. When this happens, the user may be billed separately by each of the providers, or one of the providers may act as an integrator, so that the user is billed by only that one provider, and that provider is billed by other providers (with

recursion possible, of course). For example, if I have a leased line between Princeton, NJ, and Champaign, IL, I would probably lease the portion between my NJ site and the nearest long distance carrier “POP” (point of presence) from Bell Atlantic, lease the portion between my IL site and the nearest POP from Ameritech, and lease the NJ-IL portion from the long distance carrier. I’m not sure, but I think I would get bills from all three. If instead I make a single long distance call over the public network between the same two locations, the same type of path is used, but in this case I think the long distance portion is billed by usage, and the two local portions are simply part of the fixed rate monthly bills for local service at the two sites (or maybe the long distance carrier pays the local carriers for the usage), and I get a single bill from the long distance carrier.

- Billing for software on the basis of use: Instead of trying to control the distribution of software, you control its execution. Before an instance of a program will execute, it must be able to identify the user and the billing arrangements. One example of such an arrangement in common use is the FrameMaker license server. An instance of FrameMaker will not do useful work unless it can communicate with a license server and obtain a license. The FrameMaker approach is still vulnerable to software theft, since users can run a pirate copy of the license server. An alternative might be for the software vendor to run a network of license servers (e.g., on the phone network or on the Internet), and require the program instances to communicate with one of them. What mechanisms are needed to make such arrangements workable and resistant to software theft?
- One example of a combination situation: Suppose you have “message unit” local phone service, and you use call forwarding to forward your calls. If someone calls you long distance, it results in two calls being made and billed. They pay for the long distance call between their location and your normal location. You pay for the call between your normal location and the location to which you have forwarded your calls. (Moreover, there is a tricky feature interaction here. When the call forwarding service picks up the call and forwards it, it may look to the long distance carrier as if the call is completed. Then if no one is home at the location to which the call is forwarded, the call originator gets no answer, but gets billed for a completed long distance call. To prevent this, there must be some way for the call forwarding service to reroute the call, but either make it look to the long distance carrier as if it has not been answered, or provide information which allows the long distance carrier to distinguish the pick up for rerouting from a normal completed call (and of course the long distance carrier must know what to do with this information).

2.2 Auditing/monitoring

2.3 Security

This section deals with micro-scenarios for authorization, IDs/authentication, and encryption. Some of the scenarios overlap more than one of these.

2.3.1 Authorization

- A resource manager grants selected rights to individuals or groups within the administrative domain containing the resource. The manager is also willing to grant selected rights to users from other domains, but does not want the administrative overhead of individual grants. She just wants to grant certain limited privileges to all members of certain other domains.

2.3.2 IDs and authentication

- A client is accessing a service through a gateway. The client, server and gateway are all in different administrative domains. Thus, an end user must have a userid for each domain and be authenticated separately in each domain. In many cases the client software may have been written under the assumption that the server is in the same domain, or at least that there is no gateway between client and server. Also, what if there are two gateways? or three? ... Keep in mind that it is not acceptable for the gateway system administrator (or anyone else) to be able to see the user's userid or password for the server.
- A federated network has a number of authentication domains, and it is very common for clients in one domain to access services in another domain. It is very undesirable to require that every user have a userid and password in every domain. The domain managers are willing to trust one another in the sense that they are willing to use each others authentication services, whatever that means. Perhaps it means that if a user presents a userid and password, the server is happy to verify it by checking the passwd file in the user's home domain.
- A decision maker needs to make use of integrated information from:
  - Reuter Company Newsyear (business news)
  - London Stock Exchange Topic database (share movements)
  - Reuter Textline (multi-national corporations)
  - Datastream (share prices)
  - FT-Profile (business information)

  Each of the information sources has its own authentication and billing arrangements. Some use a so-called “900-number”, where usage is billed to the telephone from which the call originates. Some require that a credit card number be supplied at each call, and usage is billed to the credit card. Some require that an account be established ahead of time and that the user supply a userid and password at each call, with usage billed to the user's account. The same userid and password cannot be used across the multiple information sources, since their restrictions on the format of a userid and password are incompatible. It is undesirable that each application should have to deal individually with all the different identification and authentication arrangements.

2.3.3 Encryption

- A client is accessing a service through a gateway. Both client and server are designed to use encryption for privacy of communications. However, they are in different domains, and they do not support the same encryption/decryption algorithms.

- A client is accessing a service through a gateway. Both client and server were designed to use encryption for privacy of communications, and they may even use the same encryption/decryption algorithms. However, they were not designed with the use of a gateway in mind.

2.3.4 Privacy

- Telephone customer A has an unlisted number, which indicates A's desire to keep A's telephone number private. Telephone customer B has "calling number delivery," which displays on a device beside B's phone the originating number of any incoming call to customer B. If A calls B, what should happen, and what mechanisms are required to make it happen?
- Telephone customer A has an unlisted number, and telephone customer B has "automatic callback," which will let B initiate an automatic dial back to the last number which called B if B's phone was busy when the call came in. Calls between A and B are long distance. Long distance calls are supposed to be itemized on the phone bill, showing the time of the call, number called, duration of the call, and cost. Calls are normally billed to the party who initiates them. If A calls B when B is on the phone, and if B initiates automatic dial back, what should appear on B's phone bill. Keep in mind that ***any*** information given to B about the location of origin of the call from A (general physical location, even approximate distance from B; name of party whose phone is being used, which may not be the person calling; etc.) may be highly objectionable to A.

2.4 Performance

-

2.4.1 Response time

- Service X has a response time guarantee within a particular LAN. Client Y invokes the service from another LAN, which is connected to the first one via a less-than-LAN-speed long haul network.

2.4.2 Priority

- Transactions in an MVS/IMS system are identified by an 8-character transaction code. Each code maps to a program which is to be executed when the transaction is invoked. Each code has a priority associated with it. When a transaction is invoked, the priority at which it runs depends only on the transaction code. It is not influenced by the overall application which generated that particular invocation of the transaction, and it cannot be changed dynamically by the user. It is possible to have multiple transaction codes (with different priorities) mapped to the same program, so there is some leeway for having different applications perform the same operations at different priorities.

  A collection of IMS transactions are encapsulated as operations to allow access from an ODP environment. How are priorites propagated?

2.4.3 Deadline

-

2.4.4 Throughput

•

2.5 Dependability

- Service X has a dependability guarantee within a particular LAN. Client Y invokes the service from another LAN, which is connected to the first one via a bridge of lesser dependability.

2.6 Availability

- Service X has an availability guarantee within a particular LAN. Client Y invokes the service from another LAN, which is connected to the first one via a bridge with more limited availability.

2.7 Other Quality of Service features

•

3 Micro-Scenarios for Semantics

3.1 Name assignments

This section only deals with the semantic aspects of what names are assigned to what. Differences in the syntax of names are treated in §4.4 *Name representation*.

- In CORBA environment X there is a type hierarchy TX, and in environment Y there is a type hierarchy TY. Clients in X need to access services in Y. Both environments use the same IDL. TY is a logically equivalent to a subset of TX, but uses different names. I.e., there is a one-to-one mapping from the interfaces of TY to a subset of the interfaces of TX which preserves semantics, where the term “mapping of interfaces” means a mapping of interface names, operation names and named terminations such that signatures are preserved. How is the federation of these two type systems to be managed?
- Suppose in the example above a subset of TX is logically equivalent to a subset of TY. Then how is the federation to be managed?
- Multiple different data servers in different federation domains use different names for the named terminations corresponding to the same error conditions.

3.2 Information models

3.2.1 Representation mechanisms

This section deals with different frameworks used to represent information models, quite aside from the differences in the semantic content of the information models themselves.

•

3.2.2 Semantic content

This section deals with the semantic content of the information models. For example, two different domains may use an identical Entity-Relationship modeling methodology to represent information models, but may make slightly different choices in how to represent semantically identical or similar things within that methodology. One may choose to represent a relationship between two objects as a relationship, while the other may choose to represent it as a third object which is itself related to the two objects. (This kind of thing often shows up when using an E-R framework which allows entities to have attributes, but does not allow relationships to have attributes. One modeler may want to give the relationship attributes to meet the needs of one application, so represents it as an entity. Another modelerdoes not need the attributes, so represents it as a relationship.)

- A decision maker needs to make use of integrated information from:
  - Reuter Company Newsyear (business news)
  - London Stock Exchange Topic database (share movements)
  - Reuter Textline (multi-national corporations)
  - Datastream (share prices)
  - FT-Profile (business information)

  One can encapsulate each of these information sources with an object interface so as to make it usable by applications in an open distributed computing environment. However, it is necessary to base these encapsulations on a common information model, or common type system, so that information from multiple sources on the same company, or the same industry segment, or the same calendar period, can be tied together.

3.3 Type Hierarchy

- In ODP environment X there is a type hierarchy TX, and in environment Y there is a type hierarchy TY. There are cases where operations in the two hierarchies have essentially identical semantics, but slightly different arguments. For example, in one of the environments there is a “set” data type which includes an operation “cardinal” which returns the size of the set; in the other there is no such operation for the “set” data type. In the first environment there is a service “average” which takes as argument a set of real numbers and returns the average value of the set. In the second there is a similar operation which takes as arguments a set of real numbers and an integer which gives the cardinality of the set, and returns the average value. How can these two environments be made to interoperate?

3.4 Operation semantics

-

3.5 Transaction models

- The transaction model of the MVS/IMS system is that every message received by the system initiates a new transaction. Thus, there can be no conversational interaction within a transaction. How can this system interoperate with other systems which use a conversational model?
- A client supporting an interactive human user invokes an operation on a server with transactional semantics. After a great deal of resources have been expended on the transaction, the client notifies the server to abort the transaction because it is taking too long. The transaction is then rolled back, so effectively no useful work is done for the client. Who pays for the resources used?

3.6 Control of interaction

- The ODP and CORBA models are client-server models, where the client initiates all interactions. Suppose there is a need for interoperation between a CORBA environment and another distributed environment where the interaction model is peer-to-peer messaging; i.e., applications establish a connection and exchange messages over it, with either application able to initiate an exchange.
- It used to be true in the United States telephone network that only the party which initiated a call could terminate it. Even if the called party hung up, the line remained active. (This was a wonderful way to tie up a person's phone when you knew they were expecting a call from the love of their life. Call them up, and tuck your headset under a pillow.) Even now for 911 emergency services only the 911 operator can terminate the call. If you call 911, then hang up, the line remains active. Suppose in domain X only the requester can terminate a connection, and in domain Y only the server can terminate a connection, and a requester in domain Y initiates a connection with a server in domain X?

3.7 Conversation type

"Conversation type" means the protocol of the conversation; i.e., what ordering dependencies there are on the operations. For example, an SQL cursor cannot be "fetched" until it is opened, and an SQL positioned delete cannot be executed until a row has been fetched.

•

3.8 Trading Classification/Context

•

3.9 Trader federation

Editorial: Many of the issues surrounding trading in a federation are not unique to trading. They are the same for any service. An example of this would be handling names across contexts. This section is intended for issues which are unique to trading.

•

3.10 Exception handling

- In federation domain X exceptions are returned through named terminations. In domain Y they are returned through a "status" result. What happens when clients in X invoke services in Y, or vice versa?
- A client in domain X accesses a server in domain Y through a gateway. Both domains use the same named terminations for the same types of errors. How does the client distinguish a network failure between client and gateway from a network failure between gateway and server, since both result in the "network failure" named termination?

4 Micro-Scenarios for Mechanics

4.1 Interaction model

- Domain X is based on a connect-oriented model, where clients establish a connection to servers, then can make multiple invocations of operations sequentially over the connection. User authentication and other service parameters are established at the time the connection is set up and do not need to be re-established at each invocation. Domain Y is based on a connectionless model, where each operation invocation is made in isolation, and where authentication and other relevant parameters are established at each invocation. What happens when a client in one domain needs to invoke a service in the other?

4.2 Interconnection technology (media, protocols, addressing)

- An application running on a PC under Windows needs to be able to access information on:
  - — Dialup data sources, via modem
  - — Corporate databases, on an MVS mainframe
  - — Department databases, on minicomputers

  The MVS mainframe only supports SNA communications. Some of the minicomputers support only TCP/IP, some only DECnet. The PC is on a Novell network, where the underlying transport protocol is SPX/IPX. This diversity of communications protocols all needs to be transparent to applications.
- Two implementations of CORBA use the same RPC protocols and provide all the same invocation infrastructure and services, except that one uses TCP/IP as the underlying transport protocol, and the other uses ISO TR4. It becomes necessary for objects in the two environments to interoperate.

4.3 RPC protocol (message encodings)

- A client in a CORBA-compliant environment needs access to a server in a DCE environment, or vice versa.
- In ISO, RPC is defined as an application protocol, in level 7 of the OSI Reference Model. In DCE, OMG CORBA, and ANSAware, the RPC protocol is implemented directly over the message passing layer (the transport or session layer). How does an ISO RPC client interact with a CORBA server?

4.4 Name representation

- When a client in name context A invokes a service in name context B, both the arguments and the results involve names. (Yes, in the computational model they involve interface references, but the recursion ends with named terminations. In the engineering model they probably come down to names well before that.) What has to be done to allow names to be correctly interpreted?

4.5 Interface type systems

4.5.1 IDL

- A client in a CORBA-compliant environment needs access to a server in a DCE environment, or vice versa. The two environments use different IDLs to describe interfaces.

4.5.2 Conformance rules

- Type checking in domain X is based on an exact match between interface types for arguments and results. Type checking in domain Y is based on the more flexible ANSA comformance rules, where interface type A conforms to interface type B if (essentially) the set of operations in A is a superset of the set of operations in B (but each operation in B must have exactly the same signature as the corresponding operation in A). Type checking in domain Z is based on a still more flexible comformance test, in which the set of operations in A is essentially a superset of the set of operations in B, but corresponding operations do not need to be identical. The arguments of a invocation made by a client must be a superset of the arguments expected by the server, and the results returned by the server must be a superset of the results expected by the client. (An example would be a statistical operation in which the server interface expects a set of numbers as input and returns the mean, median and standard deviation, but the client is designed to provide the set and the cardinality of the set as input and only get the mean and standard deviation as results.) How is type conformance checking handled in the federated system of X, Y, Z?

4.6 Transparencies (existence and/or mechanisms)

See chapter 5.

4.7 Interface references and invocations (engineering viewpoint)

- Given the great diversity of communications protocols, with their associated diverse addressing formats, and given the amount of other information which must be part of an interface reference, in the engineering model the interface reference will typically be a pointer to some infrastructure data structure where all of the associated information can be found. In a federated system, the different domains will each have their own data structure, and in fact the data structures may be totally

incompatible in format. How does one in fact pass interface references across domain boundaries?

4.8 Resource allocation policies

•

5 Micro-Scenarios for Transparencies

The micro-scenarios in this section do not necessarily arise just in federation.

5.1 Replication and Partitioning Transparancies

- A set of data objects is replicated across multiple servers. Each server has a copy of the set, and each server supports the same operations on the set. It is desirable for the replication to be transparent to client applications. They should not need to be aware that the objects are replicated, and the replicated set should appears to them as if it were a single, non-replicated set.
- A set of data objects is *horizontally* partitioned across multiple servers. Each server has a subset of the objects, and all servers support the same operations on the subsets. I.e., the subsets have identical types. The subsets are disjoint. It may be possible to determine from the attribute values of an object which subset it belongs to. It is desirable for the partitioning to be transparent to client applications. It should appear to them as if there is a single, non-partitioned set. One must keep in mind that there are operations on the set, as well as operations on the individual objects of the set -- for example, find the average value of some attribute of the objects in the set, or select a subset consisting of all objects in the set satisfying some predicate on their attribute values. An important special case is selecting a single object which has a specified value for some “key” attribute, where the key attribute is also used as the basis for partitioning -- i.e., the value of the key attribute determines which subset the object belongs to.
- A set of data objects is *vertically* partitioned across multiple servers. There is a set of logical objects. Each logical object corresponds to a group of physical sub-objects, one sub-object on each of the servers. Each physical sub-object maintains some, but not all, of the state information of the corresponding logical object. For example, part of the information about a customer may be maintained in the accounting department, part in the marketing department, and part in the customer service department. It is desirable for the partitioning to be transparent to client applications. It should appear to them as if there is simply a set of logical objects. As usual, there may be set operations to support, as well as operations on the individual objects. Moreover, there may be operations on an individual object which require state information from multiple sub-objects.
- Combinations of the above. For example, a set of data objects may be distributed across multiple servers, but the subsets may be neither identical nor disjoint. Or a set of data objects may be vertically partitioned, then some of the vertical partitions may be further horizontally partitioned. (In the customer information example, the

customer service information may be partitioned by geographical service area, while the marketing information may be partitioned by market segment, and the accounting information may be centralized.

5.2 Location Transparencies

•

5.3 Relocation Transparencies

- In domain X relocation transparency is handled by a relocation information service. When a service is relocated, the old interface reference is guaranteed to be made invalid. Whenever an interface reference turns out to be invalid, the client infrastructure can invoke the relocation information service designated for that interface and get a new reference. In domain Y relocation transparency is handled by a forwarding mechanism. When a service is relocated, a forwarding agent will respond to any service requests and reroute them to the new location of the service. How do these domains interoperate?
- Client A in domain X has access to server B in domain Y, using an interceptor. Server B relocates to domain Z, leaving a “forwarding address” with a relocation service in domain Y. Then A attempts to access server B. Any of the following problems may arise:
  - — There is no suitable interceptor between domains X and Z.
  - — Domain Z refuses to accept invocations from domain X.
  - — The relocation service in domain Y cannot deal with interface references from domain Z (even though domain X can).