Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

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

ANSA Phase III

The ANSA Interception Model

Yigal Hoffner

Abstract

There are two types of boundary problems which arise when diverse systems are interconnected, or when large systems are to be partitioned. The first boundary problem concerns technical differences which may prevent interactions between objects in the different systems and must be overcome. The second where it is necessary to create administrative boundaries in order to manage interactions, i.e. enable, disable, transform or monitor them. Interception is the process which can be used to deal with such boundary problems.

This paper introduces a model of the interception process which divides the process into distinct phases. This division allows different interception strategies to be described and developed.

Special attention is given to the passing of interface references through domain boundaries as they may be used to create bindings across the same boundaries.

The implications of some of the strategies for the design of distributed application development platforms, and in particular the link to Binding and the structure of Interface References and relocation, is discussed.

This document is now superseded by APM.1514: “Federation and Interoperability”.

APM.1427.00.02 **Draft** 11th September 1995

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

Distribution:
Supersedes:
Superseded by:

The ANSA Interception Model

The ANSA Interception Model

Yigal Hoffner

APM.1427.00.02

11th September 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

1.1 Abstract

There are two types of boundary problems which arise when diverse systems are interconnected, or when large systems are to be partitioned. The first boundary problem concerns *technical* differences which may prevent interactions between objects in the different systems and must be overcome. The second where it is necessary to create *administrative* boundaries in order to manage interactions, i.e. enable, disable, transform or monitor them. *Interception* is the process which can be used to deal with such boundary problems.

This paper introduces a *model* of the *interception process* which divides the process into distinct phases. This division allows different interception strategies to be described and developed.

The different interception strategies provide the integrator of different platforms and environments with options which allow trade-off between quality of service and resource utilization concerns.

One strategy called the immediate resolution strategy provides a way of dealing with legacy systems as it does not require any changes to be made to the supporting platforms. The deferred resolution strategy, on the other hand, is more efficient with regard to resource utilization but requires changes to the supporting platforms.

Special attention is given to the passing of interface references through domain boundaries as they may be used to create bindings across the same boundaries. In such cases the appropriate gateways may have to be inserted in the binding path. One way of achieving this is to make the gateways residing at the boundaries create the appropriate gateways and insert them where necessary. This process will repeat itself if the resulting gateways intercept passing interface references as invocation parameters. The result is a cloning like cycle of gateways creating gateways which in turn may create more gateways.

The implications of some of the strategies for the design of distributed application development platforms, and in particular the link to Binding and the structure of Interface References and relocation, is discussed.

Finally, conclusions are drawn from the work and suggested future directions of work are outlined.

1.2 Motivation

The scope of distributed systems is increasing as more of the computing facilities in offices, departments, organizations and multinational companies are being connected together. In such environments it is necessary to facilitate the cooperation between different systems where this is desirable, while at the

same time prevent interaction between them where this is undesirable. In many situations where the auditing of interactions between different systems is imperative, the appropriate monitoring facilities have to be put in place. This has to be done in a dynamic fashion as the links between different service providers and consumers are set up, used and are destroyed as the need arises.

Technology to support the process of Interception is necessary. It will create and insert gateways where appropriate in order to overcome technical differences and create administrative boundaries which can be used to manage federated systems.

1.3 Interception scenarios

There are two major cases to be examined:

- where autonomous systems are being interconnected with a view to co-operation between objects from the different systems
- where an existing system is to be partitioned.

1.3.1 Interconnecting autonomous systems

A merger between two companies requires their computer systems to be connected together. Unfortunately, their systems differ in a number of respects:

- one organization supports PCs connected on a LAN whereas the other is based on an ORBIX platform. The communication protocols supported differ as well as their RPC mechanisms
- the administrative procedures supported by each organization differ:
  - in the way the information is organized
  - in the way requests for things like external purchases are made: some purchases previously done with external companies now have to be done between the two merged parts. While it is possible to change the purchasing system of one of the organizations or the other - this may be too costly

The difference in communication protocols requires a comms gateway between the two networks. This can be done by using specialized hardware.

The RPC difference can be overcome by application (independent) level gateways.

The administrative differences can be overcome by application level gateways but the details are likely to be application specific.

1.3.2 Partitioning of an existing system

A department in an organization is to be divided into two separate departments in a company restructuring. The interconnected PCs are to stay connected to the common network and communication between the two departments is necessary. However:

- for reasons of accountability, it is necessary to record some of the transactions between the departments

- it may be necessary to prohibit others. For example, the data-bases of the partitioned departments are not all accessible to people in the other department and the appropriate access control are necessary to ensure that only authorized access takes place.

The former may require gateways to be set up at the boundary between the two departments to monitor the interactions.

The latter can be done at the databases themselves if they are shared or alternatively by gateways constructed between the partitioned parts.

The partitioning of a system is an interesting example as it may occur physically in that the members of the two departments may be physically separated and hence the network may be partitioned as well. Alternatively, the partitioning may be logical, that is, each person remains to work at their station but now belongs to one department or the other.

In the first case, the partitioning may encapsulate all the members of the department requiring one gateway to separate them from the rest. In the latter, each station will have to be encapsulated by a (notional) gateway to distinguish it from others.

1.4 Background material

The reader is expected to be familiar with the ODP [ODP 93] computational and engineering models and with CORBA [OMG 92].

This document, in conjunction with future planned documents about:

- type system and type repository federation
- trader federation

will replace the “ORB Interoperability” document [APM.1021 93].

1.5 Overview and summary of current work

Interception is about recognizing when things which are bound together (such as clients and servers, for example), will require the insertion of gateways which can be subsequently be activated when the binding is used for interacting between the objects.

Interception is a process which can help:

- interactions to cross (technical) boundaries where the properties of the domains delineated by the boundaries prohibit interactions
- manage interactions by creating (administrative) boundaries which can:
  - enable the crossing
  - disable the crossing
  - transform the crossing
  - monitor the interactions crossing boundaries.

After introducing the model of interception and the different types of action that may be taken when interactions cross domain boundaries, the document concentrates on three issues:

1. the different *approaches to domain boundary crossing* which provide gateway designers with options concerning resource allocation,

performance and hence QoS issues: the conservative immediate resolution approach and the deferred resolution approach.

2. the *changes* which are required in distributed application development platforms to support the deferred resolution strategy.
3. the *minimal agreement* that has to be reached between two domains and their gateways in order to be successfully interconnected.

1.6 Aim of developing the model

The aim of developing the model of interception is to:

- investigate the management of the interception process
- describe the options available to a gateway designer
- explore the changes necessary in distributed application platforms to accommodate the different approaches to interception
- explain the link between interception and trading
- use the model to describe and explain issues such as relocation, the ANSA Binding model, proxy Trading [APM.1446 95] and others. These are areas where the interception process has been applied implicitly but this was not recognized as such at the time.

1.7 Structure of document

Chapter 2 provides an overview of the relationship between interception and trading by describing what happens when interface references are being passed in a dynamically configurable distributed system.

Chapter 3 describes the two main factors which affect the action to be taken at a gateway when an invocation passes through it:

- the type of boundary being crossed
- the type of information crossing the boundary.

Chapter 4 describes the ANSA model of interception.

Chapter 5 discusses the different types of transformations carried at different types of boundaries.

Chapter 6 introduces the different strategies which gateway designers can take when implementing interception

Chapter 7 looks at the structure of interface references required to facilitate gateway creation and the deferred resolution strategy. (The implications for the binding process are discussed in Chapter 10). The suggested structure fits with the Universal Networked Objects (UNO) proposal [BROWNELL 94].

Chapter 8 discusses the problems which arise when interface references cross domain boundaries which require gateways.

Chapter 9 looks at the role of the resolver in relation to the process of binding, migration, relocation, passivation/activation and domain crossing resolution phase. Conclusions are drawn and a recommendation for the definition of the resolver interface is given.

Chapter 10 discusses the implications of the new interface reference structure and information for the binding process.

Chapter 11 provides conclusions and future directions.

2 Problem overview

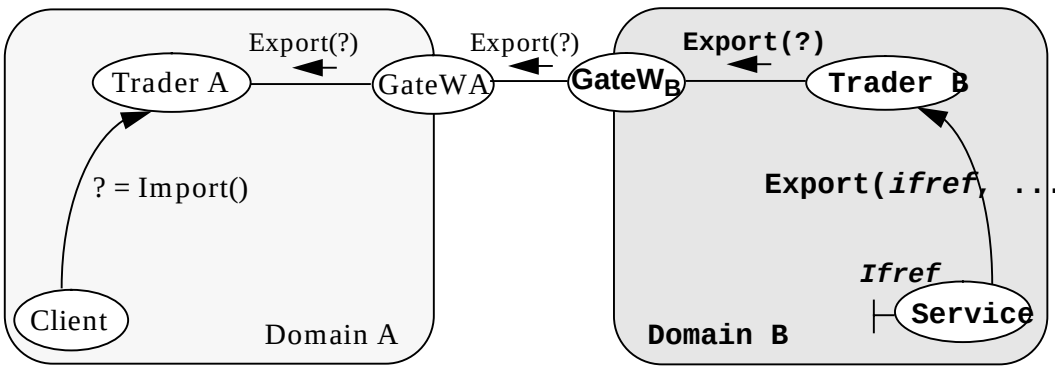
This chapter provides an overview of the relationship between interception and trading by describing what happens when interface references are being passed in a dynamically configurable distributed system.

2.1 Setting up the binding between objects in different domains

Distributed systems require the passing of information in order to set up bindings between objects at run-time in a dynamic fashion. The heterogeneous nature of such systems also indicates that different domains or pockets of homogeneity will exist, and that objects from these different domains will interact with each other. In such cases, gateways which deal with the differences are required at the domain boundaries.

Figure 2.1 shows a situation where a client and a server reside in two different domains A and B. Each domain has its own trader and these are connected together through gateways which deal with the differences between the two domains[1].

Figure 2.1: Trading across domain boundaries

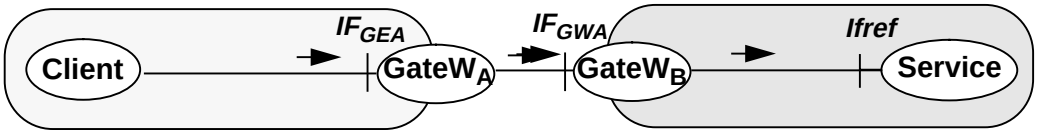


A possible scenario is one where the server in domain A exports its interface reference to its trader which in turn exports it to the trader in domain B. The same circumstances which necessitated the gateways between the traders may dictate the need for gateways between the client and server resulting in the setup shown in Figure 2.2.

Special care must be taken when an interface reference passes through a boundary which requires some transformation to be performed, or a decision to be taken, when it is being crossed. The interception process has to detect this happening and insert the necessary gateway, capable of carrying out the necessary transformations, in the invocation path of the potential link before

1. This in turn raises the question of how the gateways between the traders were established in the first place, when and by whom.

Figure 2.2: Binding across domain boundaries

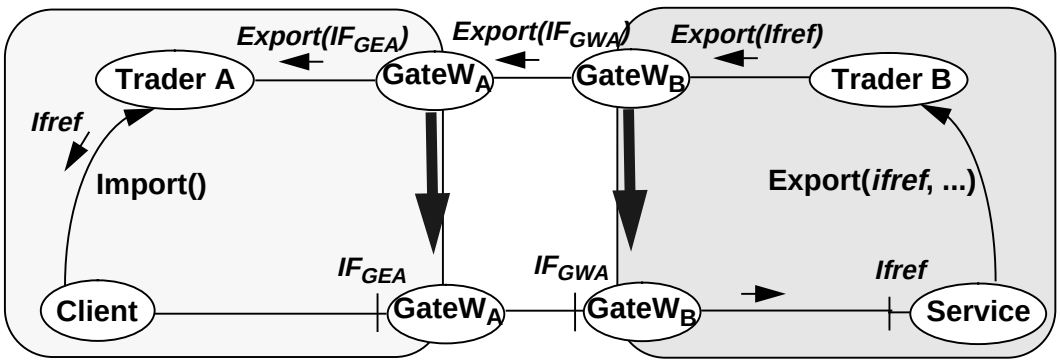


or when it is actually used. Figure 2.3 shows the relationship between the trader gateways and the gateways between the client and server.

The newly created binding between the client and server, and the respective gateways may pass interface references through them as part of client invocation or reply parameters. The new gateways may thus be required to carry out the interception process with regard to the invocation passing through it. The recursive nature of the process can be seen as cloning.

Interface reference passing is part of the trading process, either in explicit (through a trader) or in implicit form (third party trading). Hence the strong link between interception and trading.

Figure 2.3: Setting up the gateways



The questions which the rest of this paper answers are:

- in what circumstances will gateways be necessary: what are the types of boundaries which exists in distributed systems?
- what strategies for the creation, management and destruction of gateways are available?
- how can the appropriate setup of gateways be ensured?
- what management issues are involved with domains?
- what are the implications for distributed application infrastructures?

3 Types of domains and boundaries

3.1 Introduction

This chapter describes the two main factors which affect the action to be taken at a gateway when an invocation passes through it:

- the type of boundary being crossed
- the type of information crossing the boundary.

The important questions with regard to these two factors concerns who has this knowledge, when and where. The answers are pertinent to the central problem of interception and gateways: how much of the interception process and gateway code generation can be automated?

3.2 Different types of boundaries

3.2.1 Technical or "Ability/Compatibility" boundaries[1]

Boundaries are often divided into two major classes [BROWNELL 94]:

- **Technical** or **"Ability/Compatibility" boundaries** raise the question: ***can we interact safely?***

  These boundaries deal primarily with issues such as the:

  — are we talking in the same language (presentation issues)

  — are we talking about the same thing (semantic issues)

  — do we talk about and do things the same way (issues of information modelling, policy and management procedures)

A boundary in this sense delineates a scope where homogeneity with regard to technical issues can be assumed.

3.2.1.1 Examples of technical types of domains are:

- Communication (access method) domains: the variety of communication protocols create technical boundaries. For example, RPC communication protocols (e.g. DCE [OSF 92], ANSAware REX [ARM 93], CORBA [OMG 92])
- Data representation domains. Examples of such domains are:

  — language specific data representation: the way in which data is encoded in the system, for example, language bindings and problems such as little and big-endian ordering of bytes

  — Interface reference domains: different structure and representation of information necessary for a client to be able to bind and invoke server (e.g. ANSAware Ifref's[ARM 93], DCE binding handles [OSF 92] and CORBA object pointers [OMG 92])

1. Inhibition and prohibition boundaries.

- quality of service (QoS) domains: where guarantees of performance differ it may be necessary to intercept any attempts to create bindings across such boundaries depending on the requirements of the client and server
- descriptive languages: type description languages such as IDLs. There are two possible differences among them:
  - — syntactic: e.g. ANSAware versus CORBA IDL
  - — semantic: e.g. CORBA versus DCE.

3.2.2 Administrative or "Permission" boundaries

- **Administrative** or **"Permission" boundaries** raise the question**: *do we want to interact*/*are we allowed to interact?***

  These boundaries deal primarily with issues such as the:

  - — ownership, permission and control (the right to create, to provide or withhold a service, to charge)
  - — permission, liability and accountability: issues of guarantees, contractual obligations and what happens when things go wrong or one party is dissatisfied

A boundary in this sense delineates a scope where permission to co-operate is granted, while attempts to co-operate outside it have to meet with approval.

This division can be misleading: while interception usually attempts to overcome technical boundaries and create administrative boundaries, this is not the entire story. Sometimes in spite of the intention and permission to co-operate, there will be administrative differences in the manner in which things are done. In such cases it is differences of an "administrative" nature which have to be overcome, not strictly speaking of a technical nature. Hence the more appropriate use of terms like "ability/compatibility" and "permission".

3.2.2.1 Examples of Administrative domains

Examples of administrative domains are:

- authority, permission and control boundaries:
  - — where in order to interact with objects outside a domain it is necessary to receive the permission of the management of the domain
  - — where resource allocation is dictated by domain policy
- security:
  - — where authentication of the other party is necessary
  - — setting up secure communication channels
  - — defining access controls
- remuneration: where differences exist in
  - — notions of measurement of work
  - — relation between work and pay: costing of effort
  - — billing strategies
  - — payment strategies
  - — receipt exchange procedures.

3.3 Type of information crossing boundary

It is necessary to distinguish between invocation parameters which carry:

- **addresses** or **Interface References**: the passing of Interface References (in the form of explicit or implicit trading) constitutes a potential binding and therefore poses special problems. If the subsequent binding is created across the same boundary which is being crossed by the invocation carrying the Interface Reference, the creation of gateways will usually be required
- **information other than Interface References**: e.g. data such as integers, reals, strings or composite records thereof. These may or may not require translation depending on the type of the boundary crossed. No creation of gateways will be required.

3.3.1 Application specific gateways

The central questions concerning passing information are:

- **who** knows which parameters require transformations at a specific type of boundary and **when** do they know it?
- **where** is such information available?
- to what extent is this **application specific** knowledge? Can the above question be answered by looking at parameter types or is application specific information and human intervention necessary?

  In the case of an interface reference this is possible to determine because they are explicitly typed (e.g. as interface reference in ANSAware, object-pointers in CORBA and binding-handles in DCE). Special attention can be given to them where necessary without any application specific knowledge.

  Some parameters may require transformations but the type information may be insufficient to detect this automatically without additional application semantics.
- can application specific information be represented formally so that it can be used, like IDL, for example, to generate gateways?

These questions have important implications concerning the level of automation which can be achieved when generating the code for specific gateways.

3.3.2 Recursive nature of interface reference passing

Where an interface reference is passed as a parameter it will be necessary to determine the type of the interface it is pointing to. This will be necessary in order to determine whether any of the parameters of an invocation on that interface will themselves carry interface references as parameters. The situation may be recursive (Figure 3.2).

An example of the type relation between interface references is shown in Figure 3.2.

Gateways will have to intercept these interface references and construct the appropriate gateways, which in turn will have to intercept passing interface references and construct the appropriate gateways, etc.

Figure 3.1: Recursive nature of Setting up the gateways

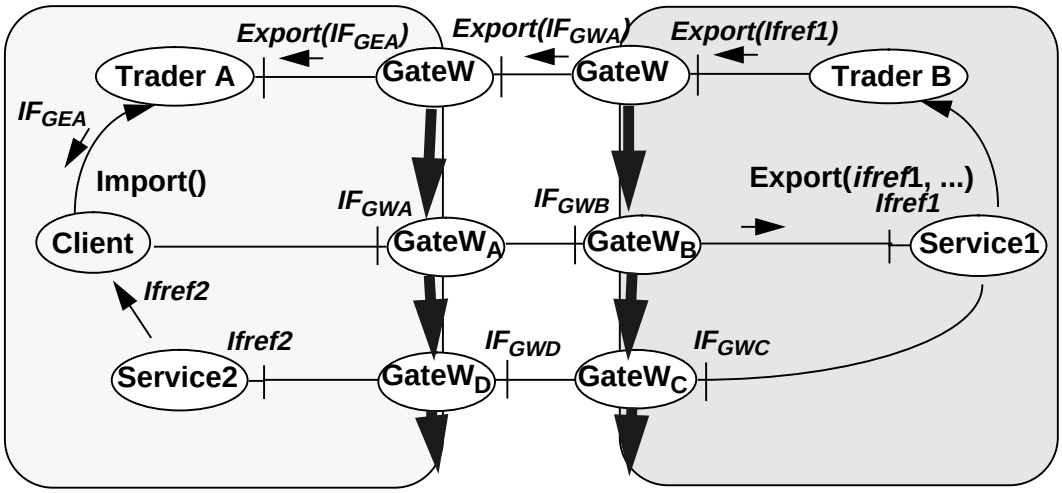
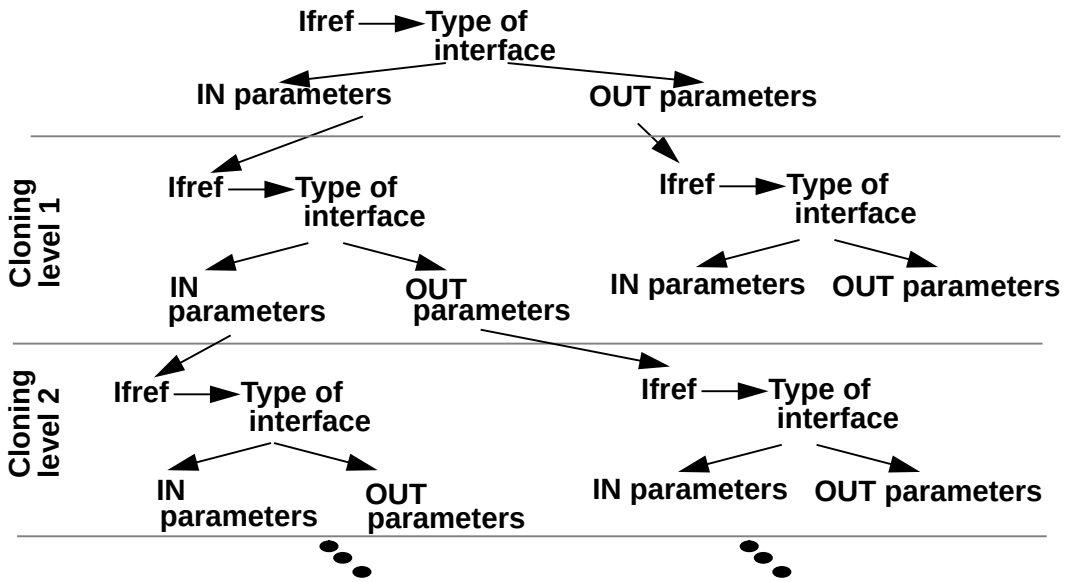


Figure 3.2: The recursive nature of interception where parameters are interface references



There is therefore a requirement to be able to determine the type of an interface to nth level in case it and its descendants pass other interface references in their invocation parameters. This indicates a link to type inference topic [APM.?]. This will have to be done either:

- **statically**: by computing all the derivative types before run-time
- **dynamically**: by using facilities such as the CORBA DSI/DII interfaces [IONA 93b] which allow dynamic marshalling and unmarshalling stubs to be created.

3.4 Types and interception

In order to determine whether the passing of an interface reference may result in further interface references being passed as invocation parameters it is

necessary to have the signature of the interface. The manner in which the signature can be obtained will determine whether gateways will be created:

- statically: all the gateways and their children gateways will have to be set-up as templates which can be instantiated at run-time
- dynamically: with the passing of an interface reference, the gateway can obtain the signature of the referred interface. It can then create the gateway by using the dynamic marshalling and unmarshalling routines such as provided in CORBA DSI and DDI [OMG 92]

The information about the type of service referenced by an interface reference can be obtained in different ways:

- Dynamic type checking:
  - **global/shared repositories**:

    Each interface reference will carry the name of the type it supports. This would be sufficient to obtain the interface signature from the global repository. The main disadvantage of this approach is that global repositories are unlikely to scale well. The management of such a repository is also difficult.

  - **reference** to the relevant type repository in the interface reference:

    This is a solution which allows run-time type checking without the need for a global repository and hence would scale well
- Static type checking:
  - **compare signatures** at compile time:

    Allows pre-computing the type of parameters to detect which subsequent binding will require gateways. Main disadvantage is that it requires creating the stubs regardless of whether they are going to be used or not.

  - **pre-computed** type graph:

    This is an optimization of a signature comparison in that it allows a faster way to determine sub-typing relationship. This however, is not enough for type inference purposes which requires obtaining the type signature.

4 The model of interception

4.1 Introduction

The ANSA model of interception has two major aspects:

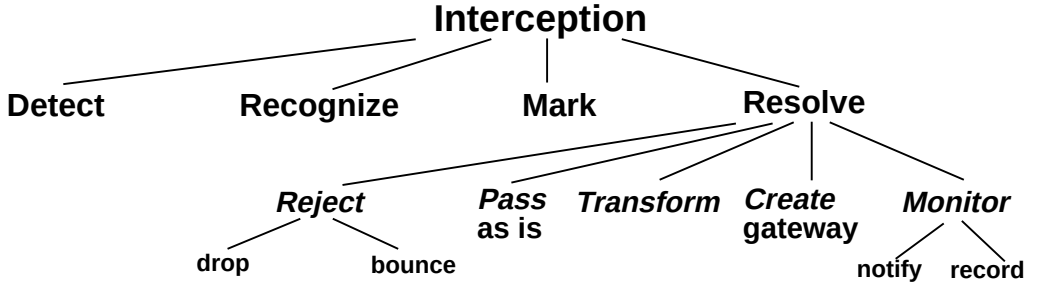
- it describes what happens when an invocation crosses a domain boundary by dividing the process into distinct phases
- it describes the recursive nature of the interception process which results in a cloning like cycle in those cases where interface references are being passed through a domain boundary.

Application of the model to special cases such as Interface References crossing boundaries will be discussed in Chapter 8.

4.2 The phases of the Interception process

The interception process can be divided into the following distinct phases (Figure 4.1). These phases are described below.

Figure 4.1: Phases of the interception process



Another view of the interception process is that of the relationship between the domains involved, the boundary being crossed and the gateway (Figure 4.2).

Figure 4.2 shows a gateway as the agent which carries out the interception process.

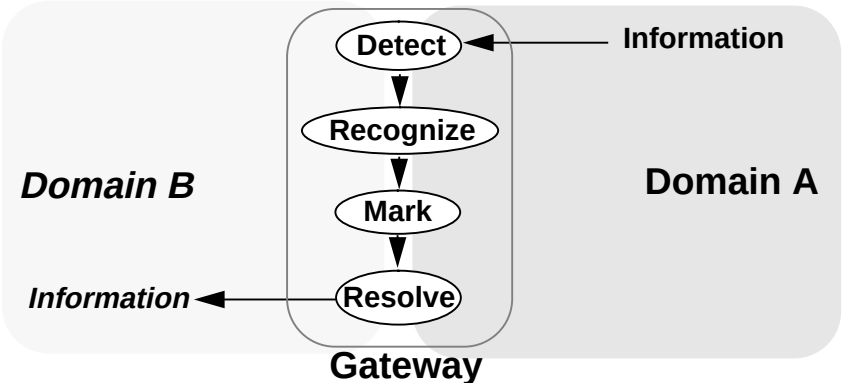
4.2.1 Detection phase

This phase is involved with the detection of an invocation crossing a boundary. It indicates that the potential for the crossing must have been detected at an earlier epoch and the appropriate mechanisms put in place to detect the actual crossing of an invocation.

4.2.1.1 Information for the Detection phase

This includes knowledge about the existence of the boundary and the ability to detect potential exit or entry from a particular domain.

Figure 4.2: Phases of the interception process



4.2.2 Recognition phase

This phase is involved with recognizing the parts of the information (invocation parameters) carried by the invocation which:

- require modification as a result of the crossing
- necessitate some action by the interception process.

Determining what parameters require action to be taken depends on the type of the boundary which is being crossed.

4.2.2.1 Information for the Recognition phase

- This includes information on
  - boundary being crossed (Type of difference between the domains, e.g. RPC type boundary[1])
  - the types of parameters which require action.

4.2.3 Marking phase

This phase is involved with the marking of the information which needs acting upon by identifying what needs to be transformed and adding information about where or how this is to be done (Figure 4.3).

The different ways of marking can determine or at least influence:

- which objects can request the resolution
- where the resolution will be carried out, and how and when

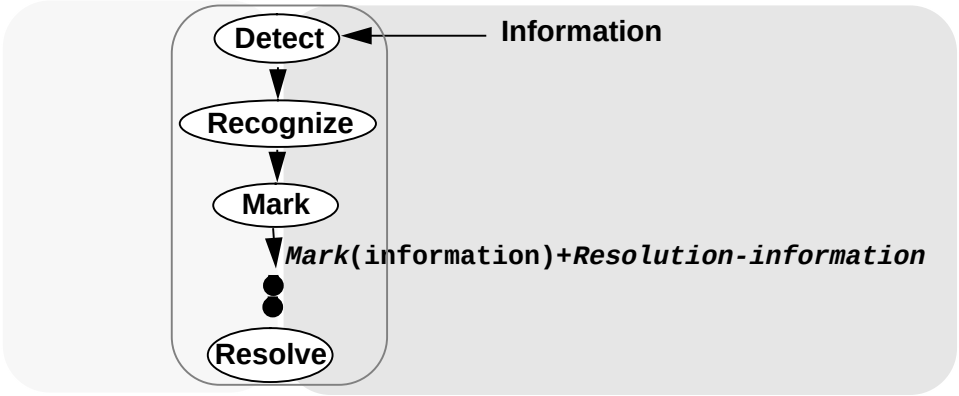
4.2.3.1 The marking

The marking of an invocation can be done at different levels:

- marking the entire invocation: ***Mark[***Op(arg_1, arg_2, ..., arg_n)***]***
- marking the Operation name: ***Mark[***Export***]*** and thereby modifying it, for example the Export() operation can become ProxyExport()
- marking the arguments: Op(***Mark[***arg_1, arg_2, ..., arg_n***]***)

1. Difference between type of boundary and the value of domain is similar to the difference between PropertyName and PropertyValue in ANSAware Trader or Orbix Match-Maker. Another example of a the type of boundary is a border between countries, the specific countries are the values e.g. Germany and France.

Figure 4.3: The marking phase



- marking a specific argument: Op(arg_1, arg_2, arg_3,...***Mark[***arg_m***]***,..., arg_n). Example of this is the marking of a property list by adding an additional property describing the gateway crossed.

4.2.3.2 *Information for the Resolution phase*

The Resolution-information may provide:

- where to carry out the resolution:
  - — name of Resolver
  - — type of Resolver
- what to do (how to carry out the resolution):
  - — script telling an agent what to do. This can be useful where for security reasons the information has been encoded and cannot be transformed at the gateway without compromising the security of the system. The end point will be given the encoded information and a script telling it what the gateway would have done had it had access to the decoded information. The transformation can be carried out at the stubs for example
  - — description of domain boundary crossed. This may provide the object requesting the resolution with information concerning where and how the resolution is to take place.

4.2.3.3 *Information for the Marking phase*

- This includes knowledge of the domain separated by the boundary (value of difference, e.g. RPC_{Dom1} = *DCE* RPC, RPC_{Dom2} = *CORBA* RPC) and the implication of the crossing for the invocation and its arguments:
  - — which objects can request the resolution
  - — where the resolution may be carried out. In some cases the marking phase will mark the information in a way which will force the resolution at a specific place. This may be necessary for security reasons for example. In other cases the marking can be regarded as a hint as to where it is possible to resolve the foreign information.

4.2.4 Resolution phase

This phase is involved with carrying out the necessary action as required by the crossing of the boundary. The resolution phase may be one or a combination of the following (some of these, but not all, are mutually exclusive):

- *rejection* of the crossing with or without notification to the source
- *passing* the invocation without changes
- *transforming* the content of the invocation
- *creating* any appropriate gateway (or gateways) in case of an interface reference being passed
- *monitoring*: notifying some interested agent or recording the event.

For example, a rejection of a crossing is unlikely be accompanied by the creation of any gateways, whilst translation and creation of gateways does not necessarily exclude each other.

4.2.4.1 Information for the Resolution phase

This includes knowledge about gateway type and parameters, factories and resources necessary for the creation of gateways.

4.3 Gateways cloning gateways

A special case which we will describe in detail in Chapter 8 concerns the passing of interface references across domain boundaries. If the new bindings are created across the same boundary, they will require the creation and insertion of gateways (i.e. the resolution in this case will be the creation of the gateway). This may happen recursively, so that if the new bindings are of a type which includes an interface reference as a parameter, the gateways will have to be able to act on those in the same manner. The resulting situation can be regarded as a cloning sequence.

An important difference between the cloning and cloned gateways may be the type of interfaces they present to their clients. **Type inference** will be required in order to help with setting up the gateways with the appropriate interface type.

4.4 Design and interception

The above description of interception cycles as a cloning process indicates that the interception process is applied at least once:

- at design time: when different entities are decomposed (and possibly distributed) e.g. when a monolithic piece of code is divided into subroutines
- at integration time: when different entities are brought together e.g. client and server.

At this point, the designer or the tool involved has to be able to determine whether later interactions across this boundary will themselves necessitate interception. If this is the case, the mechanisms which can carry out interception will have to put in. The designer is therefore carrying out the initial interception process, ensuring that the appropriate gateways are put in

place and are capable of carrying out the interception process on the invocation passing through them.

4.5 Distributing the interception process in time and space

4.5.1 Distribution in time

By splitting the interception process into phases and recognizing its potentially recursive application, it becomes apparent that the interception process can be distributed in space and time. This means that the different cycles, as well as phases of the interception process, can be carried out by different tools at different epochs. In particular:

- who carries out the initial cycle and who carries out the subsequent ones
- who carries out the detect/recognize/mark phases: designer, compiler, linker, factory, traders, gateways, binders, clients and servers
- who requests the resolution: gateway, trader, client, binder
- who performs the resolution and where:
  - gateway resolver,
  - a local resolver (e.g. Binder)
  - a proxy

The distribution of the phases of the process allow different implementation approaches and different trade-off between early and late resource allocation. This affects:

- resource usage
- length of time to bind and performance on first invocation
- predictability of response to the request to bind

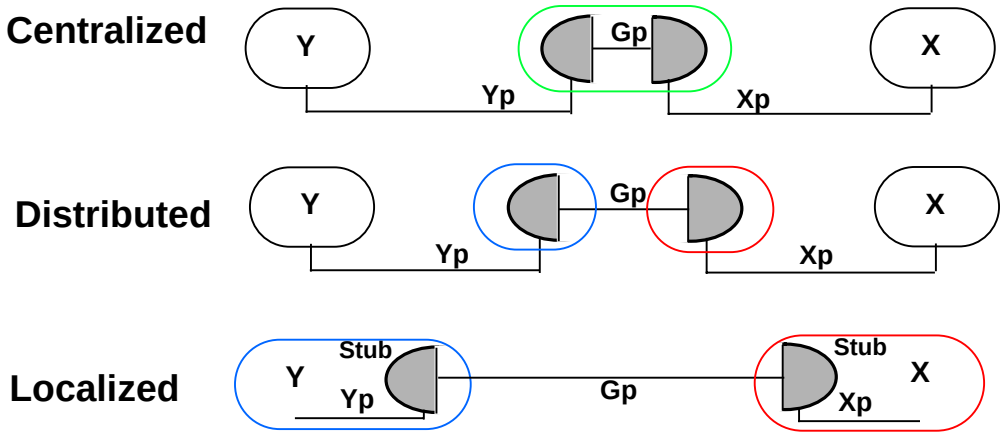
One result of the distribution of the interception process phases is different interception resolution strategies outlined in Chapter 6.

4.5.2 Distribution in space

The distribution of the interception process in space allows choices of where the resulting gateway can be inserted (Figure 4.4):

- **localized**: stubs in the client and/or server
- **distributed**: half-gateways at domain boundaries or anywhere between the client and server
- **centralized**: single (centralized gateway at client or server side, or in between in an independent domain.

Figure 4.4: Physical distribution of gateways



5 Gateway transformations

Note: This chapter requires further work!

5.1 Introduction

As mentioned in Chapter 4 section 4.2.4, the *resolution phase* of the interception process can carry out one or more of the following with invocation parameters: reject the invocation (i.e. bounce back), pass it without changes, create a gateway (i.e. a proxy in the case of an interface reference being passed), monitor the invocation (i.e. record the information passing) and/or transform the parameters.

This chapter discusses the different types of transformations which may be carried out at different boundaries.

5.2 Types of transformations carried at boundaries

There are different forms of transformation which can be carried out at gateways:

1. **representation** (encoding) **conversion**: this type of transformation is well understood. Example are:
   - stubs when marshalling and unmarshalling invocation parameters to overcome differences of concrete representations of data by different programming languages and operating systems (e.g. byte ordering)
   - RPC: this includes header information issues but also the more complex issue of RPC semantics translation. For example once only guarantees.
2. **adding parameters**, for example:
   - for security (e.g. authentication) purposes
   - for monitoring purposes
   - for transactions purposes.
3. **removing** parameters:
   - platform specific information which cannot be translated into an equivalent in another platform
4. **encryption** for protection:
   - to be decrypted at the recipient side
   - to force the recipient to go back to the gateway in order to decode it if is going to be used (e.g. encrypting interface reference information).
5. **abstraction** of types which are undefined in the recipient domain into an object. The created object can be regarded as a “type gateway” sitting on a

type boundary (the object must support the operation which the recipient is expecting from the service parameters). **The main difference between this and creating a gateway as a result of interface reference passing, is that in this case the recipient is not expecting to get an interface reference but data**. The recipient expecting a data parameter must be able to detect and deal with an interface reference instead of the expected parameter (this may be non-trivial).

6. **migration** of an object across the domain boundary in order to make local an object
   - — which cannot be accessed remotely
   - — for performance reasons

   This raises many security and portability issues. In some cases a proxy will have to be left behind in the place where the object migrated from.
7. **naming and semantic translation**: in most cases this is likely to be application specific transformation (?).

5.3 Application specific and application independent transformations

Note: Too vague!

Application specific transformations will involve any differences in application semantics and interface definition.

Application independent transformations will involve any transformation which do not involve the above differences. These will include differences in interaction language (RPC, comms) and transparencies.

I general it will be possible to automate the generation of gateways for the latter in cases where a mapping exists between the differences.

5.4 Agreement between domains for interception

Note: These are preliminary thoughts and the section requires more thinking and more work!

What is the minimal agreement required to link two domains through gateways:

- name of an agreement: still leaves the bootstrapping problem of how to reach more complicated agreements from an initial one

Practically speaking:

- language of interaction
- sets of base-types and mappings as part on initial set up
- derived types
- interface references.

6 Interception resolution strategies

6.1 Introduction

There are different ways in which the marking phases and the resolution phase of the interception process can be joined, thereby creating a spectrum of resolution strategies. The two extreme strategies are:

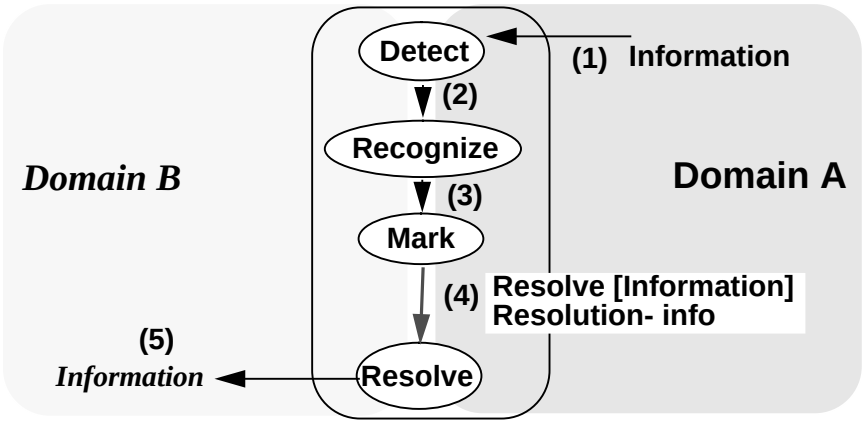
- **immediate resolution**: the resolution process is carried out regardless of whether the transformed information will be used or not
- **deferred resolution**: accomplished by passing the marked Interface References out of the domain with an indication of where the resolution should take place
- **intermediate**: flavours exist which are combinations of the extremes with optimization for special cases.

The different strategies allow the designer to make a trade-off between resource allocation constraints and performance requirements. Early resource allocation has implications for the guaranteed quality of service of those links which cross the domain boundaries.

6.2 Immediate Resolution strategy

Where the resolution process follows the Detect/Recognize/Mark phases of the interception process, the result is the **Immediate Resolution** strategy (Figure 6.1).

Figure 6.1: Immediate resolution process



The resources required for the resolution process are allocated to it regardless of whether the transformed information passing through the domain boundary will be used in Domain B or not. This will be wasteful in cases where it will not be used. In the case of interface reference passing this may pose

considerable overheads. The worst case occurs if gateways have to be created for every interface reference passed.

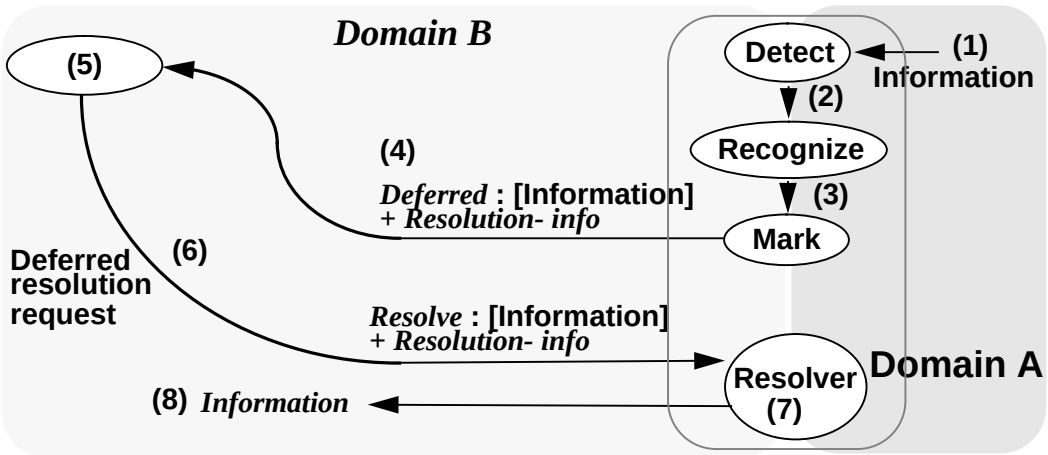
The advantage of the immediate resolution strategy is that if the already transformed information is passed into Domain B, it will require no further transformations which may be time consuming and cause delays at a later stage.

An advantage of this strategy is that Domain B infrastructure will require no changes as it will receive information in the form it is expecting. The entire transformation is carried out in the gateway prior to the information crossing the boundary. This is important in cases where it is not possible to modify an application or an environment, which is typically the case when dealing with legacy problem.

6.3 Deferred Resolution strategy

If the Detect/Recognize/Mark phases are separated from the Resolution phase, allowing an object in Domain B to decide when (and sometimes where) the resolution process will take place, the result is the **Deferred Resolution** strategy (Figure 6.2).

Figure 6.2: Deferred resolution strategy



In step (4) the gateway's marked information is forwarded to a recipient in Domain B. This object (or any object it passes this information to) can decide to use the information (5) and request the resolution (6). This strategy will require a Resolver which can accept invocations from objects outside the gateway. More of this in chapter 8.

The decision as to whether to use the immediate or deferred resolution strategy rests ultimately with the gateway and will be dictated by the policy of the domains which the gateway spans and the gateway management policy. Choosing the deferred strategy will require the recipient domain to be capable of receiving marked information.

The Resolver may not necessarily reside in the gateway itself but can be located anywhere within the recipient domain. It is also possible to have a default resolver for one or more types of boundaries crossed, so that the marking does not necessarily have to include the reference to the Resolver in the marked information.

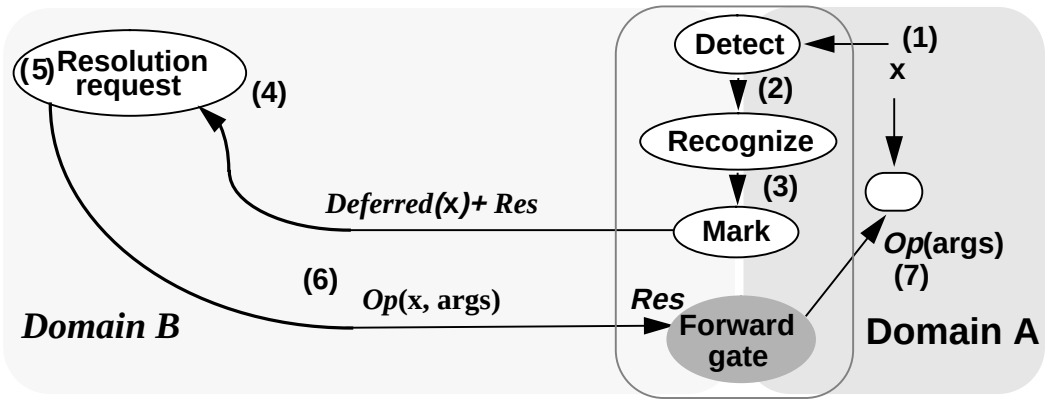
6.4 Intermediate Resolution strategies

6.4.1 Bounce-forward Resolver

Note: No resolution of Ifref to a local handle; Invocation is performed on Resolver interface.

When an interface reference is passed through a gateway, it is possible to set up the Resolver as a Bounce-forward agent. When the recipient object in Domain B wishes to use the address passed to it to invoke it, it includes the interface reference of the server with the invocation parameters sent to the Resolver.The resolver acts as a bounce-forward agent (Figure 6.3).

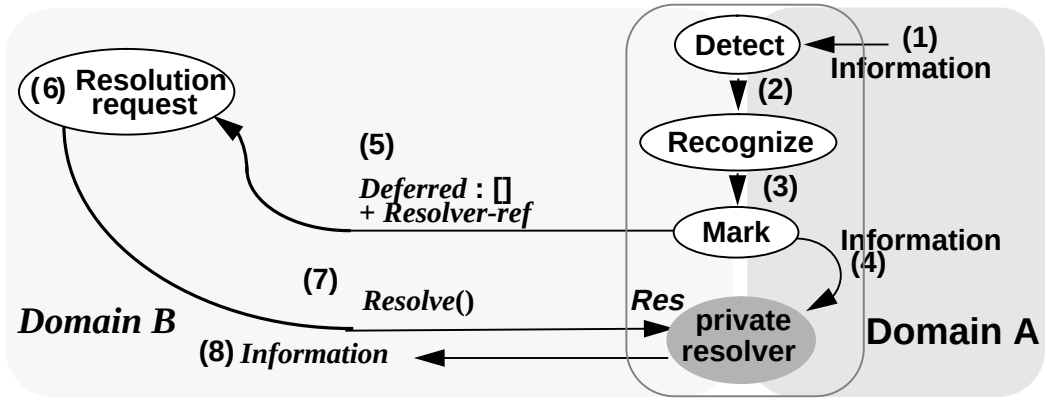
Figure 6.3: "Bounce forward" strategy (no resolution)



6.4.2 Forward but leave

In this scheme, the information passed into Domain B is a reference to a specific resolver which holds the information to be translated. Where it requires a considerable amount of space to be carried around, this would constitute a considerable saving (Figure 6.4).

Figure 6.4: "Leave but forward" strategy (deferred resolution)



7 Interface Reference structure and content

This chapter looks at the structure of interface references required to facilitate gateway creation and the deferred resolution strategy. (The implications for the binding process are discussed in Chapter 10). The suggested structure fits with the Universal Networked Objects (UNO) CORBA standard [BROWNELL 94].

7.1 Interface Reference structure

Considering the different roles a resolver may undertake: gateway resolution, relocation, migration and passivation/activation, there is a requirement for Interface References to provide for:

- **alternative** paths
- the possibility of **relocation/migration** of services
- the possibility of a cascade of **gateways** represented for the deferred resolution method.

The proposed generic structure for interface references is the following:

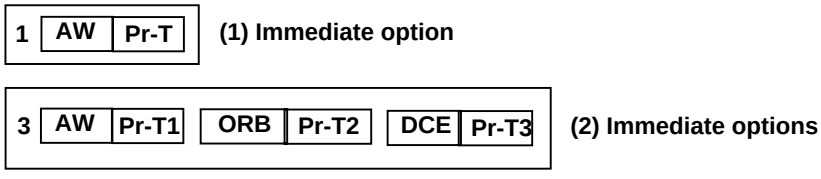
- the generic interface reference may have zero or more interface reference records
- each interface reference record is marked with the name of the platform in which the service resides
- each interface reference record has information concerning the protocols supported by the service
- interface reference records can be nested inside other platform's interface reference records.

The following sections describe how the proposed structure can be used in different circumstances.

7.1.1 Alternative paths: Invocation Reference options

An interface reference may have zero or more records providing information about possible paths between the client and server (Figure 7.1).

Figure 7.1: Interface Reference structure - one or more alternative paths



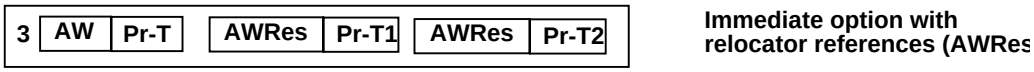
The Binder will be able to choose whichever option matches the available comms infrastructure, QoS constraints and cost in terms of resources.

Some of these paths may require a resolution process to be performed before they are used.

7.1.2 Relocation options: Invocation Reference sequenced structure

Where relocation is available in case of the server failing or migrating, this will be represented as a sequence of interface references in increasing order of scope (Figure 7.2).

Figure 7.2: Relocation information in Interface Reference

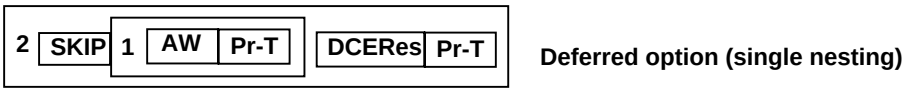


When an invocation fails, the client infrastructure can request the Binder to intervene. The Binder then searches the interface reference either for an alternative path (in case of comms failure) or for a resolver interface reference (in case the server relocated). The Binder then invokes the Resolver/Relocator giving it the interface reference. If successful, the Resolver/Relocator will return a new interface reference of the relocated service.

7.1.3 Gateway cascade: Invocation Reference nested structure

Where a cascade of gateways is present, and the deferred resolution method is used, the information about the cascade can be represented in a nested fashion (Figure 7.3).

Figure 7.3: Deferred resolution information inside an Interface Reference



There are two cases to consider:

- where the nested records are NOT options: Binder must not skip a gateway and try and bind directly to an interface nested inside the structure
- where short cuts can be made: this is particularly important where circularity of reference is to be prevented.

Whether the nesting indicates a compulsory path or not depends on the type of boundary crossed and the enterprise/management issues associated with the crossing of the boundary. Either way it should be left to the Resolver to decide.

The information marked as deferred can also be encrypted so as to prevent its resolution by unauthorized agents. This can help prevent taking any short-cuts where it is necessary to force the use of a gateway.

It is possible to have multiple stages of deferred resolution (Figure 7.4).

It is also possible to have combinations of deferred options as well as immediate options and their associated resolver/relocators (Figure 7.5).

Figure 7.4: Several stages of deferred resolution

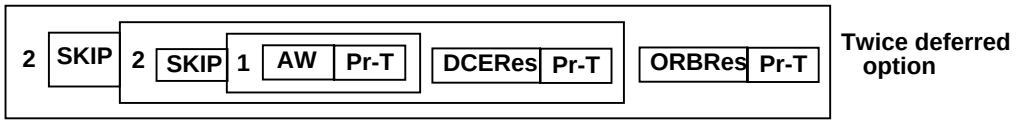
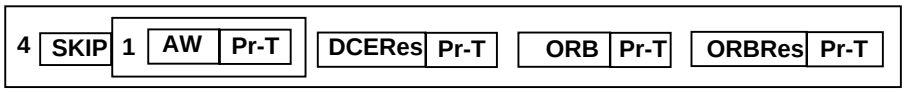


Figure 7.5: Options including deferred resolution



7.2 Requirements from the infrastructure

The above discussion indicates the need for:

- Extension to Interface References to allow:
  - — structure of interface reference to include
    - (i) sequences of interface reference records
    - (ii) nested interface reference records
    - (iii) be able to incorporate foreign Interface Reference and mark them
  - — language of interface reference to mark: Platform name, Deferred (SKIP), Resolve/Relocate. This requires domain reserved words: Platform name, Deferred (SKIP), Resolve/Relocate to be able to mark the foreign interface references.
- Extensions to Binder to deal with:
  - — relocation
  - — gateways
  - — other cases which should fit the scheme:
    - (i) passivation/activation
    - (ii) migration

The UNO (Universal Networked Objects) proposal includes the IOR (Inter-operable Object References) structure which caters for the above.

8 Interface References crossing domain boundaries

8.1 Introduction

It is important to make a distinction between communication domains and Interface Reference representation domains. The two types of domains do not necessarily have to coincide although in practice they often do[1].Thus we can have a common comms infrastructure but two domains, each with its different representation of interface references. Similarly we can have two comms domains but with the same interface references structure.

The passing of an interface reference through a comms domain boundary will ultimately require the creation of a gateway if the interface reference is to be used for binding the client to the server, unless:

- the client and server have another common protocol they can interact with directly, and
- the option of the other protocol is left open.

On the other hand, although the passing of an interface reference through an interface reference presentation domain boundary can be resolved by the creation of a gateway, this may not be necessary. The translation of the interface reference but not the creation of a gateway will be sufficient, provided it is possible to map the information in one interface reference to the other.

8.2 Immediate Resolution strategy and interface reference passing

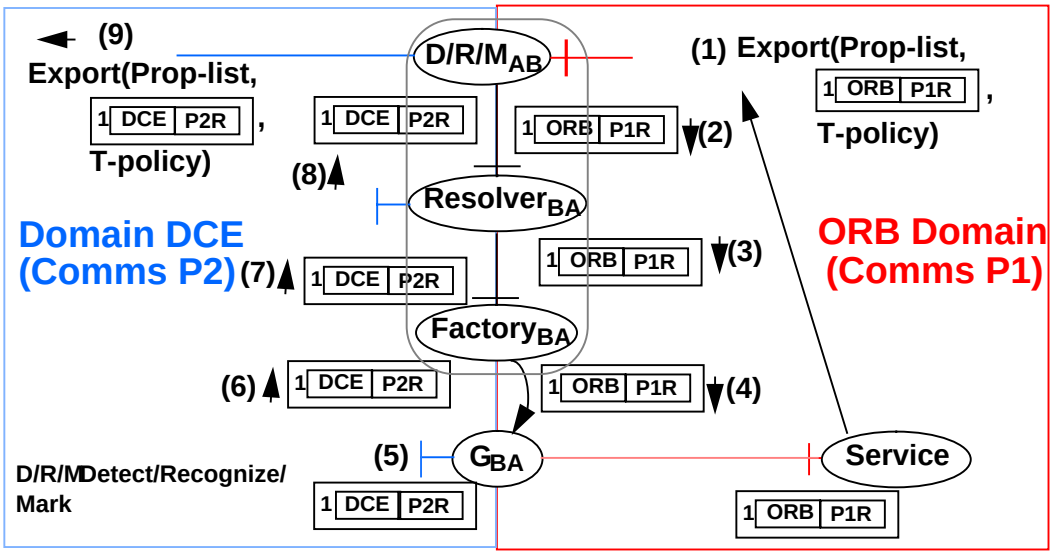
8.2.1 Comms boundary

The situation of passing an interface reference through a communications domain boundary is shown in Figure 8.1.

The interface reference passing through the gateway is detected, recognized by its type, and marked for resolution. The marked information is immediately passed to the resolver which invokes a gateway factory to create the appropriate gateway, providing it with the interface reference of the service in Domain A, and an indication of whether to set up the binding immediately or not. The gateway then decides whether to bind to the server immediately or defer doing so, according to the instructions from the resolver/ factory and its domain policy.

1. This is often because different platforms which have different RPC's also have different notions of the information and structure of interface references. See ANSAware and Orbix platforms, for example.

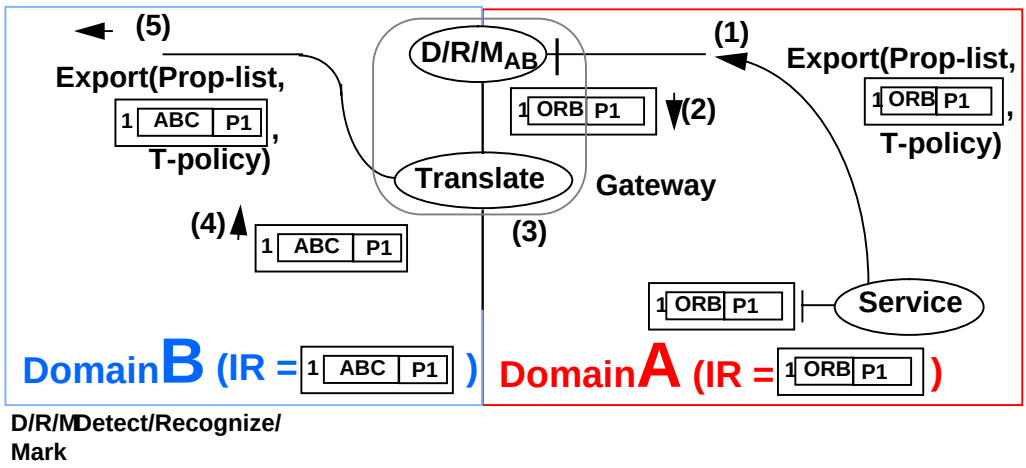
Figure 8.1: Comms protocol boundary crossing (Immediate Resolution strategy)



8.2.2 Interface reference boundary

Where the presentation interface reference information (i.e. addressing, protocol and QoS information) differs, the result is different Interface Reference domains (Figure 8.2)).

Figure 8.2: Interface Reference boundary crossing (Immediate Method)



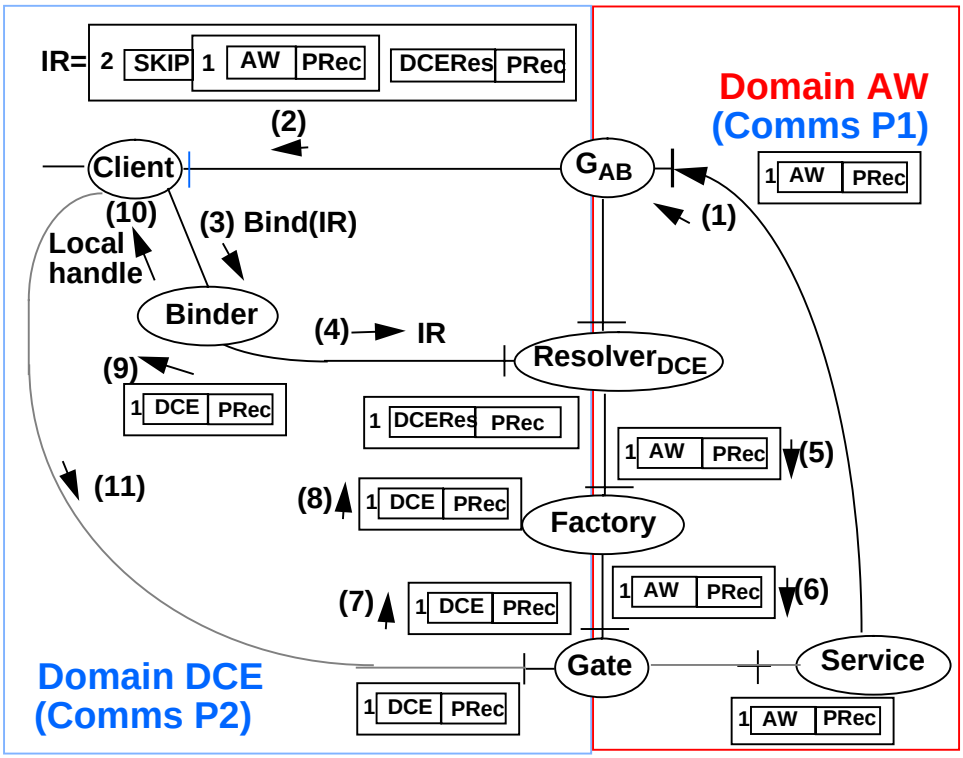
Interface reference domain crossing does not necessarily require the creation of a gateway. Where it is possible to map the information from one interface reference presentation to the other this would be sufficient.

8.3 Deferred Resolution strategy and interface reference passing

Where the resolution phase is being postponed until some object in Domain B wishes to use the marked information, the result is the Deferred Resolution strategy shown in Figure 8.3.

An example of deferred resolution with multiple stages of gateway are shown in Figure 7.7 at the end of the chapter.

Figure 8.3: Deferred Resolution strategy where an interface reference crosses a protocol domain boundary



The information in the interface reference has to be marked to enable Domain B to deal with it. There are different ways of marking the passing of an interface reference depending whether it is being passed as:

- part of an explicit Export operation: in this case it is possible to mark
  - the operation by changing it to ProxyExport for example
  - the interface reference
  - the property list
- on its own as part of third party trading.

Note: I don't like the next para. Needs more thinking!

There are two ways in which this can be done depending on whether an Interface Reference domain is being crossed or not:

- *if an Interface Reference domain is being crossed simultaneously: it will be necessary to wrap the Interface Reference from Domain A inside the interface reference of Domain B marking it as deferred information*
- *if a different domain boundary is crossed: the relevant information inside the interface reference will have to be marked. This can be done in many different ways by describing:*
  - *the type of boundary crossed and the details of the domains on either side*
  - *the type of resolver needed*

— *information on a specific instance of a resolver*

— *how to do the resolution (agent).*

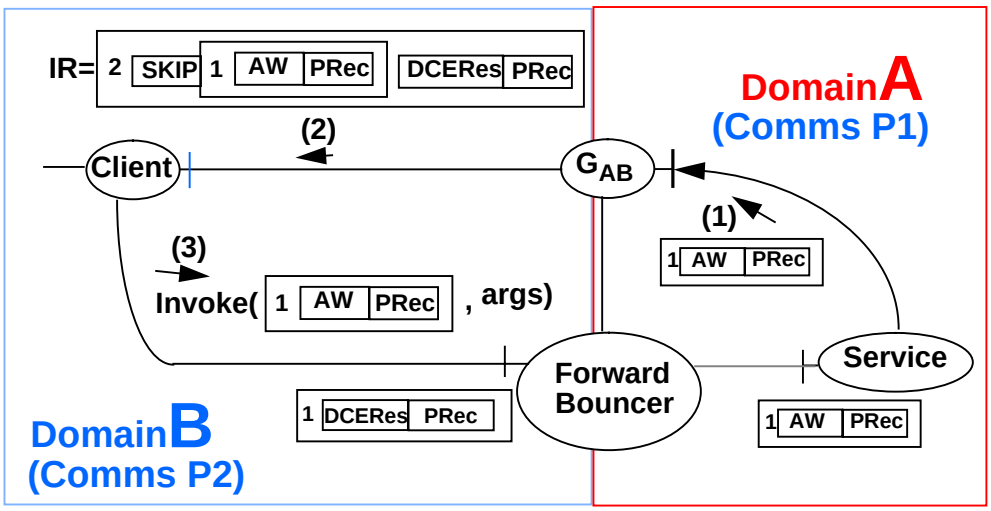
— *etc.*

8.4 Intermediate Resolution strategies

8.4.1 Bounce-Forward (through Resolver) strategy

This is an optimization of the resolution phase where the difference between the two domains requires a gateway and the information passed contains an interface reference. The resolver acts as a bounce forward agent for the invocation on that interface reference. The type of the Resolver interface has to be able to accommodate the invocation arguments and the invoker has to be able to provide the right information with the invocation arguments. Alternatively this can be done at RPC level (Figure 8.4).

Figure 8.4: Naming bounce forward strategy (no resolution)



8.4.2 Bounce-Forward RPC

This is a bounce-forward strategy implemented under the RPC mechanism. The header presented to the RPC mechanism includes the nested interface references. The Resolver/Forward-Bouncer can extract the next relevant interface reference and forward the message with the remainder of the interface reference.

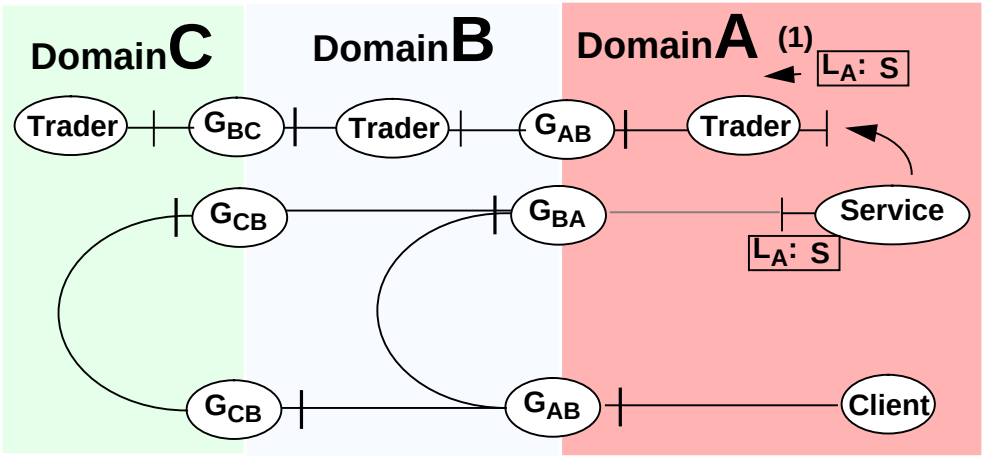
8.5 Multiple boundaries

Where more than one boundary is crossed, there is choice with regard to whether to use the intermediate or the deferred resolution strategy at each boundary. The example shown is concerned with a two stage deferred resolution strategy (Figure 7.7). Note that are several steps where there may be an alternative possible order.

8.6 Circularity of reference

When gateways are used, it is possible to have a situation such as shown in Figure 8.5.

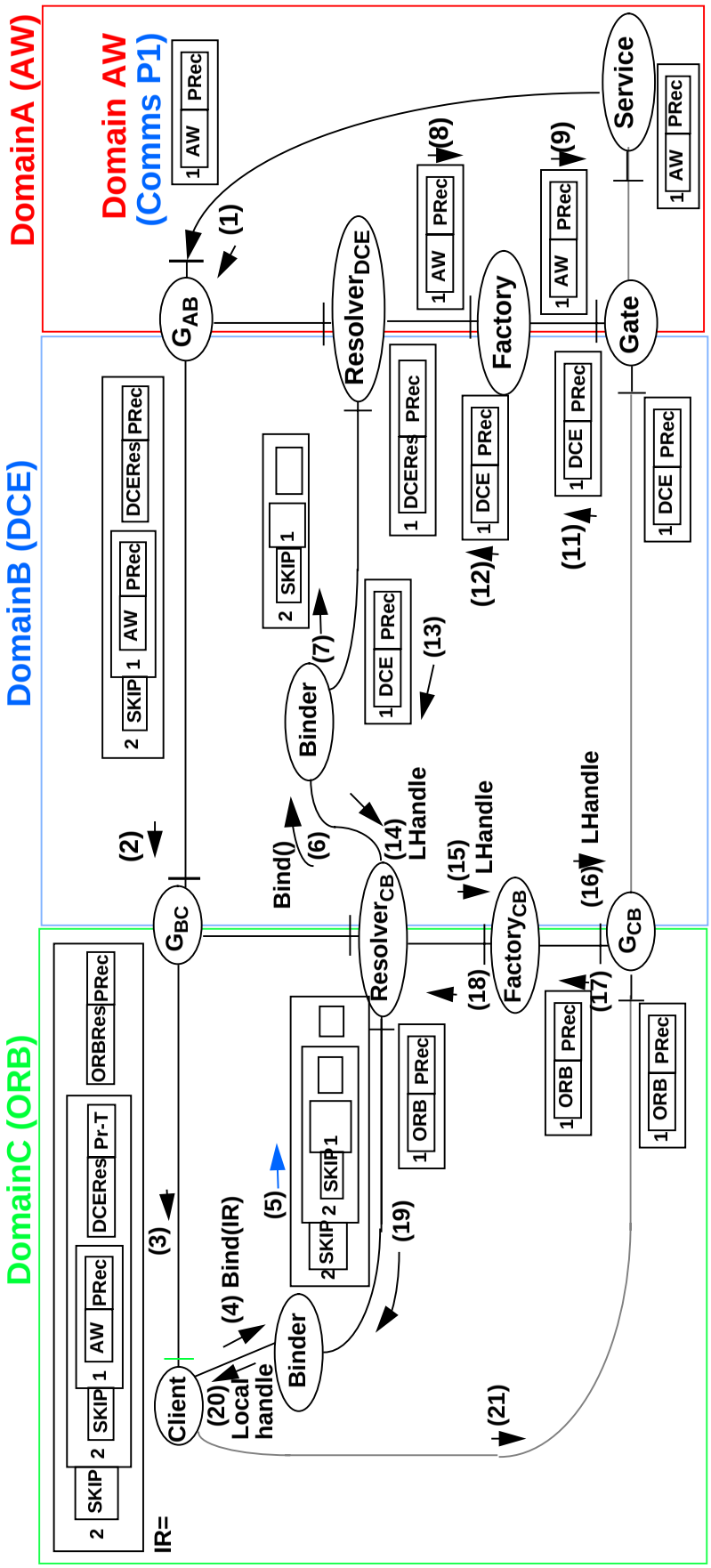
Figure 8.5: Circularity of reference



There are two approaches which can be taken with regard to circular routes:

- prevention: argument for locality of traders
- detection:
  - with deferred resolution (possible)
  - with immediate resolution (impossible)

Figure 7.7: Deferred resolution - multiple stage case



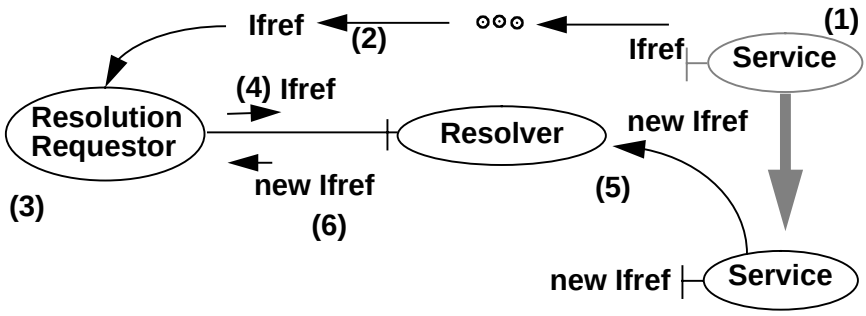
9 The Resolver

This chapter looks at the role of the resolver in relation to the process of binding, migration, relocation, passivation/activation and domain crossing resolution phase. Conclusions are drawn and a recommendation for the definition of the resolver interface is given.

9.1 The different roles of the resolver

In terms of the Interception model, a Resolver is the object which can transform the foreign information marked by the gateway when it crosses a boundary, into locally usable information. In terms of passing interface references, a Resolver is an object which exchanges unusable or out of date interface references with up-to-date ones (Figure 9.1).

Figure 9.1: The Resolver



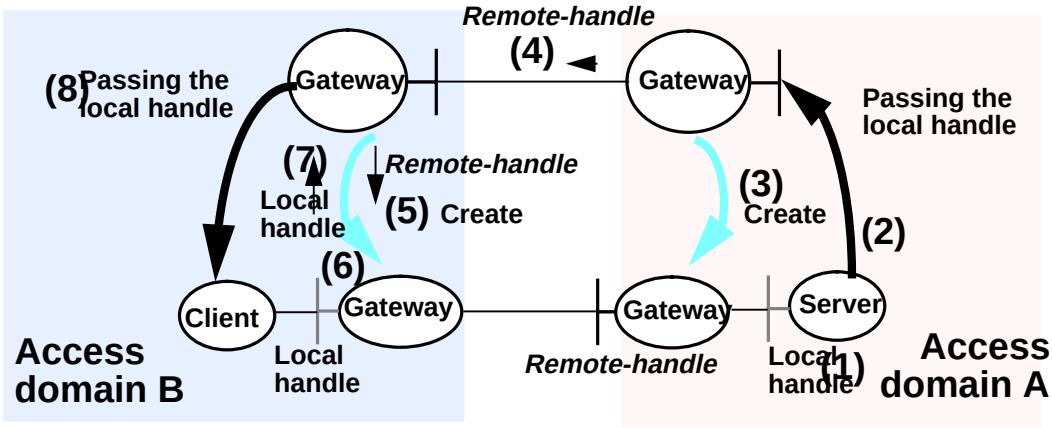
The Resolver will carry out one or more of the following roles:

- **binder**: the binder can be regarded as the resolver part of a gateway for an access method (physical separation) domain (Figure 9.2). The binder role as a resolver is to create a comms gateway to bridge the physical separation between the client and server, thereby converting an interface reference to a local handle by the client application.
- **gateway resolver**: carrying out the resolution phase by *creating* the appropriate gateway.
- **relocator-resolver**: In the case of relocation, a Relocator-Resolver will hold a mapping from *old-Ifref* to *new-Ifref*.
- **migrator-resolver**: same as the Relocator-Resolver
- **passivator/activator-resolver**: this Resolver re-activates a passivated object.

9.2 Resolution process

In general, the options a Resolver has for getting the required interface are:

Figure 9.2: Access method domains



- being notified by the relocated/re-created object
- creating the required service or gateway through a Factory
- finding it by:
  - polling other resolver services
  - enquiring Traders about other services of conformant type and matching constraints *(not sure about this one?)*
  - querying the management of domain from which the interface reference comes from, if this is possible.

9.3 The Resolver interface

The ANSAware Relocator interface provides the Relocator with the Interface Reference on which the invocation failed. In return it expects a replacement Interface Reference of the relocated service:

getIfref*(serverIfref)*
Return *(Success: SIfref; Failure : fail_code)*

9.3.1 Client side preferences

It will be expedient for the client side to provide an indication of preferences for the kind of path it requires. This may circumvent unnecessary or superfluous gateways in cases where:

- the gateway did not provide all the options in the interface reference in the first place, there is a possibility of a common protocol between the client and server and therefore no need to create any gateways
- several gateways can be created when it is possible that one of the several gateways which can be created will be enough. This may prevent a situation where the Resolver and client side have to go back and forth several times before a usable option is offered by the Resolver
- when some gateway options do not provide the appropriate QoS guarantee it will be wasteful to create these gateways in the first place.

In order to allow the object (e.g. Trader or Binder) requesting the resolution from the resolver (on behalf of the client), to provide the resolver with some

information about the client side preferences, it is suggested that the operation:

getIfref*(serverIfref)*
Return *(Success: SIfref; Failure : fail_code)*

be replaced with an operation with the following signature:

getIfref*(clientPreferences, serverIfref)*
Return *(Success: SIfref; Failure : fail_code)*

Where by *clientPreferences* we refer to information about the comms preferences of the client infrastructure: protocol and QoS information.

In case of Failure: *fail_code* should be able to denote:

- cannot provide QoS required because of resource shortage or because available comms cannot guarantee it
- insufficient resources to create gateway
- conflict with Resolver policy.

10 Binding

This chapter discusses the implications of the new interface reference structure and information for the binding process.

10.1 Binding and interface references

The binding process will have to be able to deal with the options described in sections §7.1.1, §7.1.2 and §7.1.3. In particular the Binder will have to be able to deal with foreign interface reference records within the interface reference records:

- recognizing the interface reference records where marked as deferred
- recognizing where to resolve the marked information
- distinguish between relocation resolver and a gateway resolver by taking into account the 'foreign' marking.

Note: Note: order of interface reference records is not necessarily indicative of preferences or priorities

10.2 Binder algorithm

Binder dealing with an interface reference has to:

- try each interface reference record:
  - — immediately usable record: set up comms and pass back a local handle
  - — record marked as deferred: extract Resolver interface reference and pass it the currently processed interface reference. The returned interface reference is treated in the same manner as the current one is.

10.3 Binder policy

As the order of interface reference records is not necessarily indicative of preferences or priorities from the point of view of the interface reference producer, the Binder's policy has to determine:

- whether to deal with each record at a time or search first for immediately usable references before trying to resolve deferred ones
- whether to look for short cuts in marked interface reference records

11 Conclusions and future work

Note: This chapter requires more work

11.1 Conclusions

Different approaches to interception are possible:

- conservative approach: immediate resolution does not require any changes to support platforms but can be extremely wasteful of resources
- deferred resolution provides a more efficient use of resources but requires support from the underlying platforms:
  - Binder to deal with more complex cases
  - ***getIfref****(serverIfref)* to be replaced with an operation with the following signature: ***getIfref****(clientPreferences, serverIfref)*. This has implication for the design of Relocators.

11.2 Future work

A number of related areas which have not been covered by work done to date includes:

- gateway automation issues
- gateway management: this pertains to the management of individual gateways as part of their life cycle
- domain management: this pertains to cloning issues and enforcing domain wide policy on components through interception techniques.

References

[APM.1003 93]

van der Linden, R., “The ANSA Naming Model”, APM.1003, APM Ltd, Poseidon House, Castle Park, Cambridge CB3 0RD U.K., February 1993.

[APM.1005 93]

Deschrevel, J-P., “The ANSA Model for Trading and Federation”, APM.1005, Architecture Projects Management, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1993.

[APM.1021 93]

Herbert, A.J., “ORB Interoperability”, APM.1021, Architecture Projects Management, Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1993.

[APM.1095 94]

Gomer, T. Editor, “Micro-Scenarios for Federation”, APM.1095, Architecture Projects Management, Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[APM.1314 94]

Otway, D. J., “The ANSA Binding Model”, APM.1314, Architecture Projects Management, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[APM.1299 94]

Hoffner, Y., “Gateway Visualization”, APM.1299, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[APM.1303 95]

Hoffner, Y. & Crawford, B., “ORBIX-ANSAware Gateway Design and Implementation”, APM.1303, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[APM.1310 95]

Hoffner, Y., “A Gateway Between Traders”, APM 1310, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[APM.1342 95]

“An Overview of the Description and Distribution of Services”, APM 1342, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[APM.1371 94]

Hoffner, Y., “Interception (TC Decmber 94 slides)”, APM.1371, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[APM.1387 94]

Hoffner, Y. “A Designers' Introduction to Trading”, APM.1387, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1994.

[APM.1404 95]

Hoffner, Y. & Crawford, B., “Federation Work Plan”, APM.1404, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[APM.1406 95]

Hoffner, Y., “Introduction to Application-Level Gateways”, APM.1406, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[APM.1413 94]

Hoffner, Y., “Using the Interception Model (TC March 95 slides)”, APM.1413, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[APM.1446 95]

Hoffner, Y., “Using the Interception Model to Describe Architectural Issues”, APM.1446, APM Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1995.

[ARM 93]

“The ANSAware 4.1 manual set”, Architecture Projects Management, Poseidon House, Castle Park, Cambridge CB3 0RD U.K., 1993.

[BROWNELL 94]

Brownell, D. (SunSoft]), “Universal Networked Objects”, Revised ORB 2.0 Interoperability Submission by BNR, Expersoft, IBM, ICL, IONA Technologies, and SunSoft. 1994.

[IONA 93a]

Orbix: Programmer’s Guide, Iona Technologies Ltd., Dublin, Republic of Ireland, 1993.

[IONA 93b]

Orbix: Advanced Programmer’s Guide, Iona Technologies Ltd., Dublin, Republic of Ireland, 1993.

[ODP 93]

“Basic Reference Model of Open Distributed Processing - Part 3: Prescriptive Model”, ISO/IEC CD 10746-3 (Committee Draft), ISO/IEC JTC1/SC21/WG7, June 1993.

[OMG 92]

“The Common Object Request Broker: Architecture and Specification”, Document Number 91.12.1, Object Management Group and X/Open, 1992.

[OSF 92]

OSF, DCE Application Development Guide, Open Software Foundation, 11 Cambridge Centre, Cambridge, MA 02142, USA, 1992.

[SLOMAN 87]

Sloman, M. “Distributed Systems Management”, Imperial College Research report, DOC 87/6, April 87.