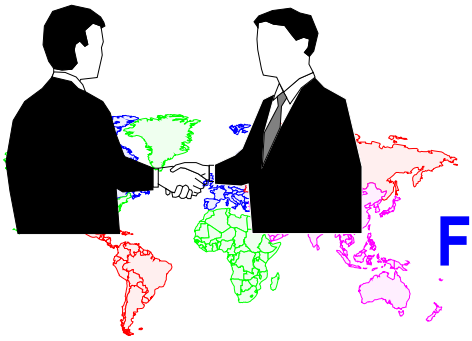
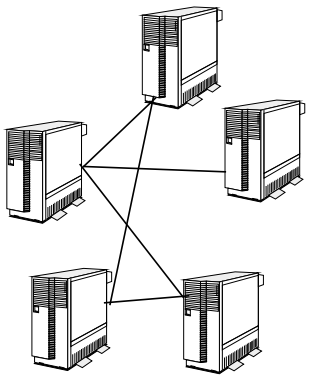


TRADING

FOR SERVICES

ON THE WWW

Andrew Herbert
ajh@ansa.co.uk
http://www.ansa.co.uk



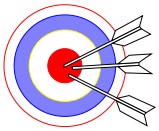
Name Services Aren't Enough

- using a name to identify a service implies
  - I know which object delivers the service
  - the object exists
  - the name is still bound to object it denoted when I learnt the name (which makes system evolution a bit tricky....)
- *trading* is about *discovering* objects
  - "Yellow Pages" for finding a suitable object in terms of *properties*
  - guarantees found object is of suitable *type*
  - discovery might include *instantiation* or *resource management*.

Using names: trading vs binding vs re-location

Trading finds a suitable object instance.
Trading fails if there is no such object.
Trading yields an *object pointer*

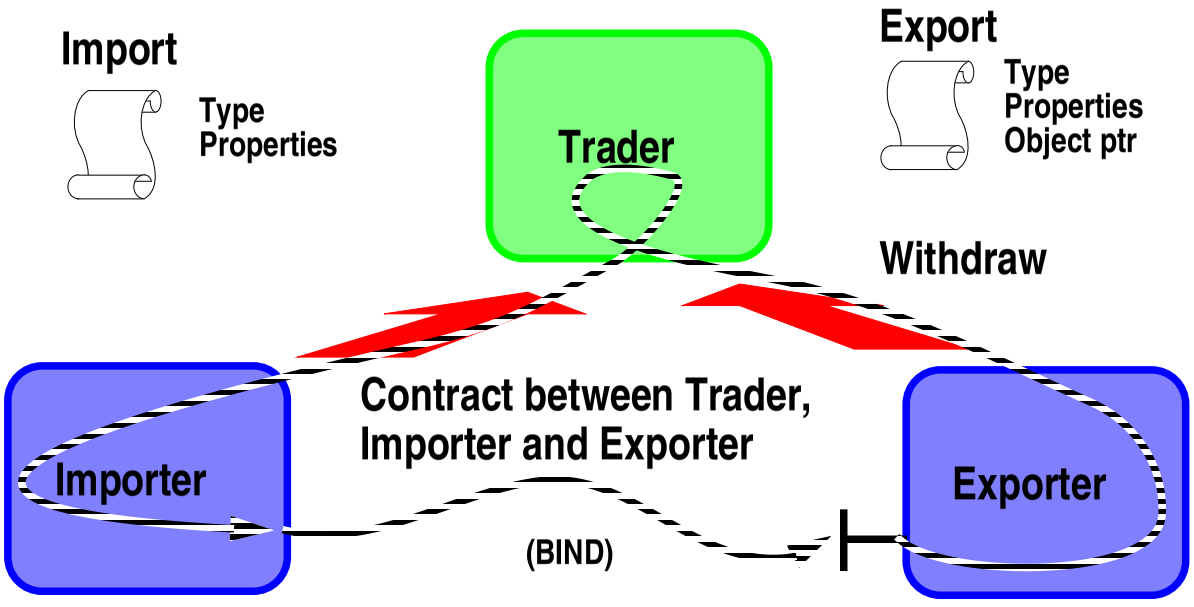
ORBs *bind* object pointers (long term names) to addresses (short term names); a binding fails if the target object fails or if the network fails



Rebinding gets you to the *same* object instance;
re-trading might get you to another object instance

Re-location tracks an object to preserve a binding
e.g. across migration or recovery!

Key Trading Operations



Type Matching

Type safety is a good thing
Compilers type check *programs* (local bindings)
Traders type check *systems* (remote bindings)

Minimum useful check is interface signature conformance

Beware of adding extra checks that might prevent servers and client from evolving separately

Trading and Properties

Signatures are not sufficient to choose a service

Who owns the service?

What will it cost to use?

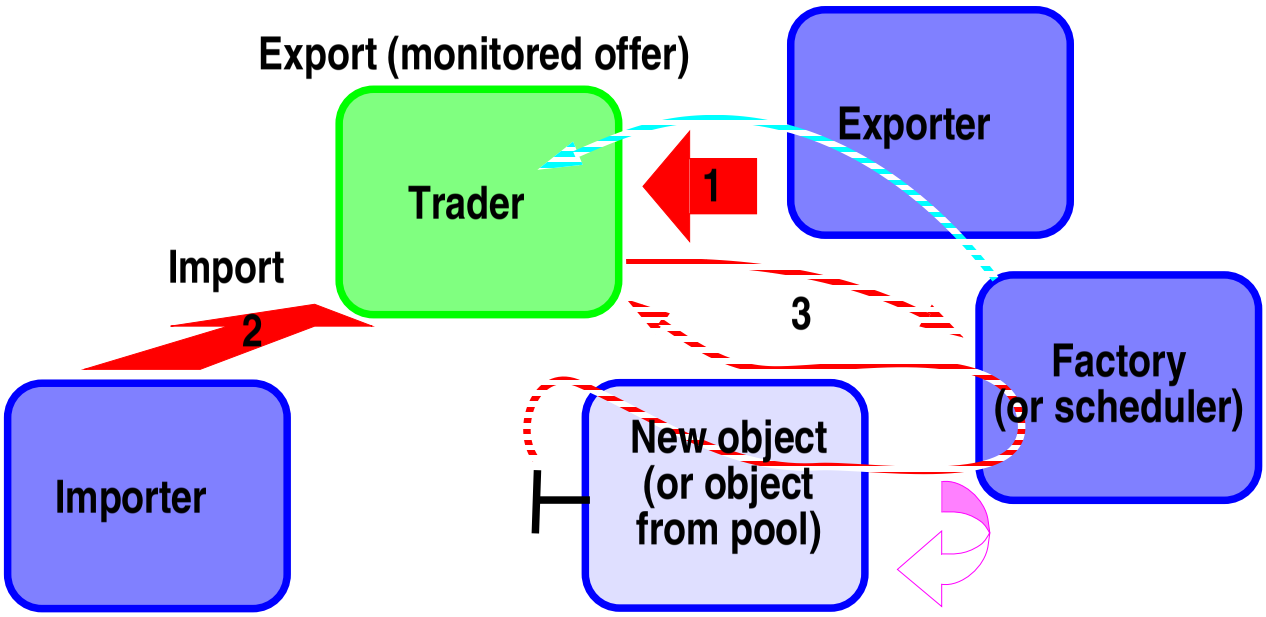
Where is the service?

What does it do?

Can I bind to it?

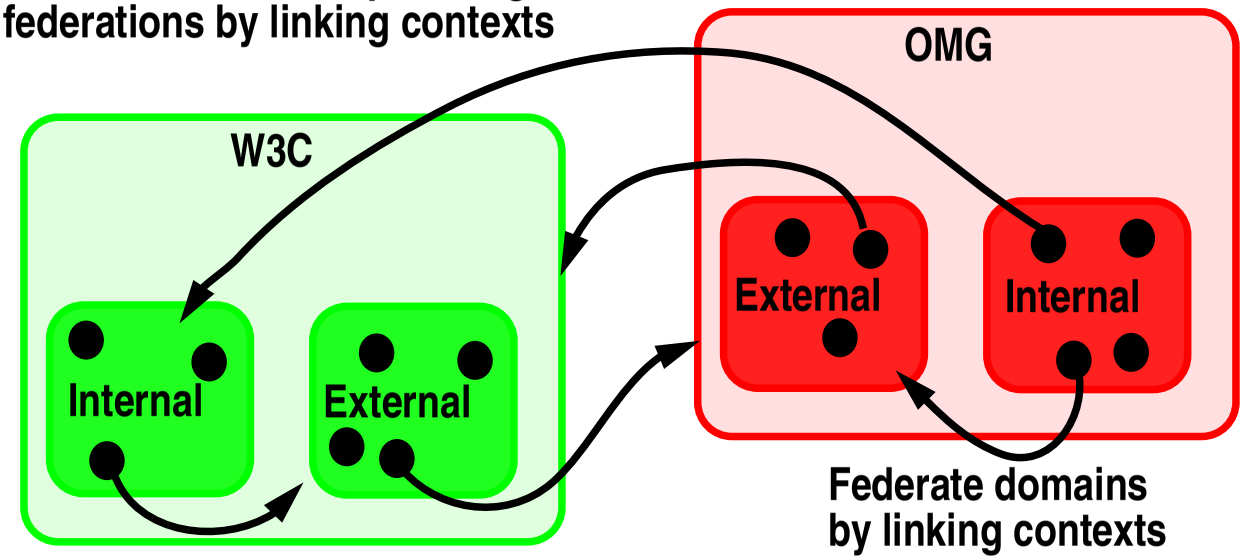
Qualify import with desired properties.

Trading and Life Cycle



Trader Federation

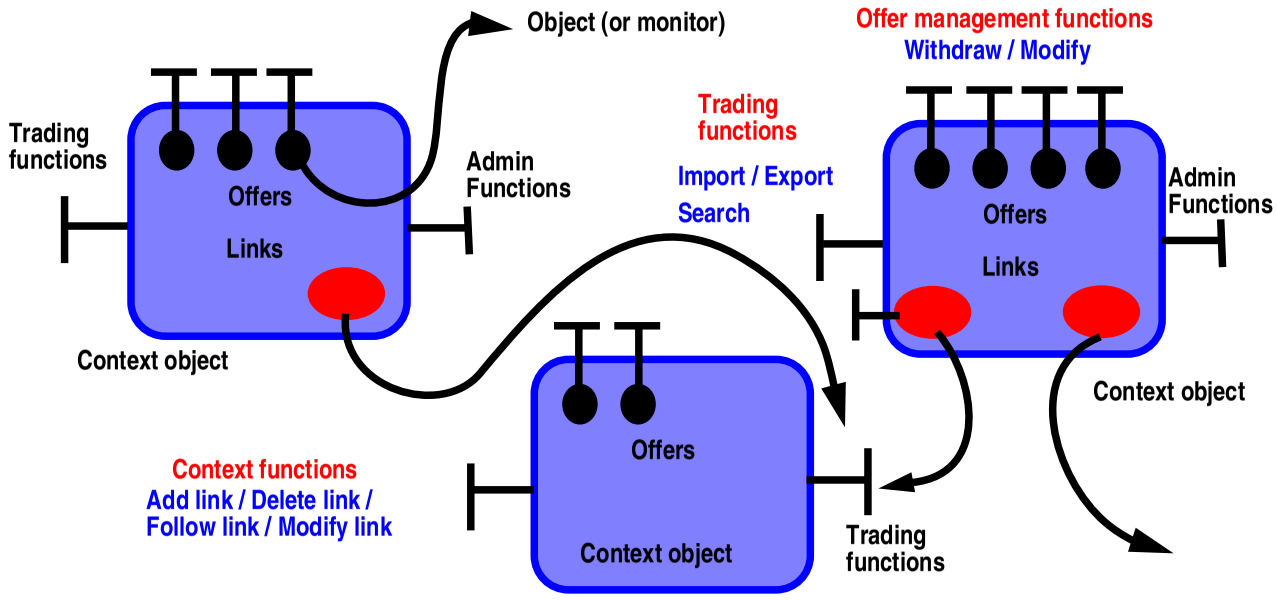
Administrators set up trading federations by linking contexts



Benefits of Federation

- Context relative naming
  - Controlled visibility of your name space to others
  - Controlled aliasing of foreign name spaces for your users
  - Your trader gets to vote on what services your users can access
  - Conventions to aid usability (e.g. ubiquitous names)
  - “join in” model fits Internet culture better than global schemes
- Choice of implementation strategies
  - customized implementation (built-in resource policy, locality, ...)
  - leveraged name server (distributed, fault tolerant)
  - data base (volume, searching)

System Model



Trading and the WWW
Possible Topics

- Benefits of *trading* when downloading objects across the WWW
- What type system do we want for Web objects?
- What life cycle support do we want for Web objects?
- Benefits of the WWW when building traders

Trading on Download

- Trade to find the right server for the object (and the right version of the object)
  - more powerful that class -> file name mappings
  - Web server becomes a Web object repository (with integrated development and lifecycle tools)
  - Server life cycle made transparent to clients
- Context relative vs “global” URLs
  - controlled visibility
  - substitution / filtering of dangerous services
  - agreed conventions for naming “standard” functions

Web Object Type System

- Assertion: signatures + context + properties gives practical disambiguity without need to manage a global name space
  - property matching is evolutionary and flexible
  - name spaces are too rigid and authoritarian
- Assertion: global name spaces are fragile and difficult to repair
  - q.v. DNS security and performance problems
  - transparency getting in the way - context relative naming can side step this
- Important properties: creator, location, version, quality of service characteristics
  - what is the information model for Web objects
  - where should the management information be stored (see later)
  - are properties “guaranteed” - what is the trust model for federated traders?

Benefits of the WWW when building Traders

- Briscoe, BT: “a Web page is a trader”
  - set of object pointers (alternative locations) plus description
  - references have open structure enabling arbitrary processing
  - context relative naming
  - description is plain text, but authoring tools could structure the meta data
- Browser facilities for client side processing of URL, driven by policy
- Analyse web page idioms and define higher level concepts (e.g. alternative URLs) to simplify user model
- Conventions for structuring URLs to permit wrapping of alternatives
  - c.f. CORBA IORs
- Investigate means to distill system management information from content