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

ANSA and Open Distributed Processing (For IEE Intelligent Networks Workshop

Andrew Herbert

Abstract

A description of ANSA architectural principles to be presented at the IEE Workshop on “The Intelligent Network - The Next Generation”

APM.1470.01 **Approved** 30th April 1995

External Paper

Distribution:
Supersedes:
Superseded by:

Copyright © 1995 Architecture Projects Management Limited
The copyright is held on behalf of the sponsors for the time being of the ANSA Workprogramme.

IEE Workshop - The Intelligent Network
The Next Generation

ANSA and Open Distributed Processing

Andrew Herbert
ANSA Chief Architect

ANSA

ANSA - The Hidden Persuader in Open Systems

Vision

Architecture

Technology

Standards

Harvest research

Build on current technology and open standards

Intercept new requirements

The ANSA Consortium

ANSA Consortium

Computer vendors

Telecoms vendors

Telecoms operators

Systems integrators

End users

SPONSORED SHARED PROGRAMME

Core Team

Founded in 1985

Barclays Bellcore BNR
BT DRA GEC GPT FT HP
ICL Iona Telefonica

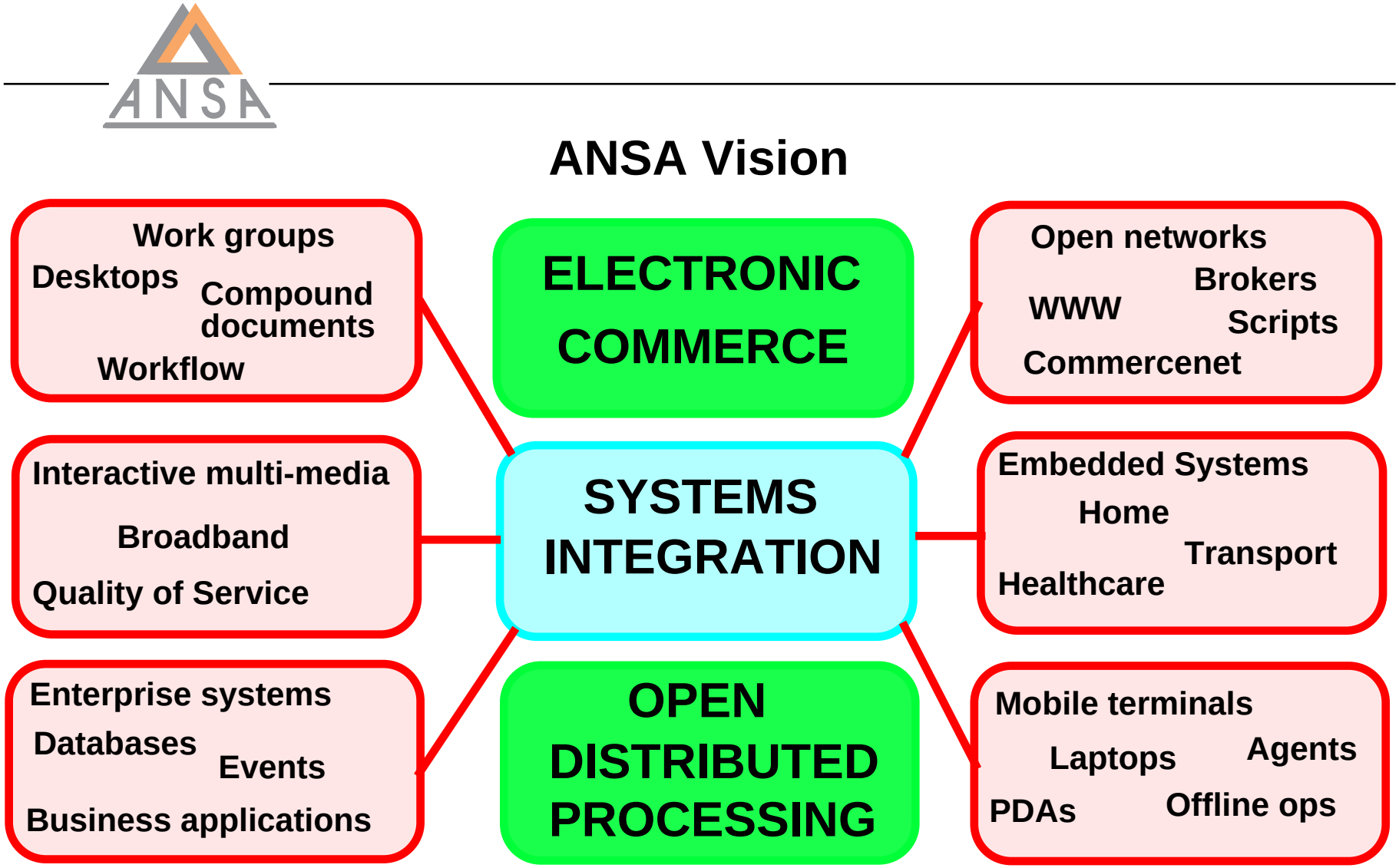
Research

prototypes
prototypes animations

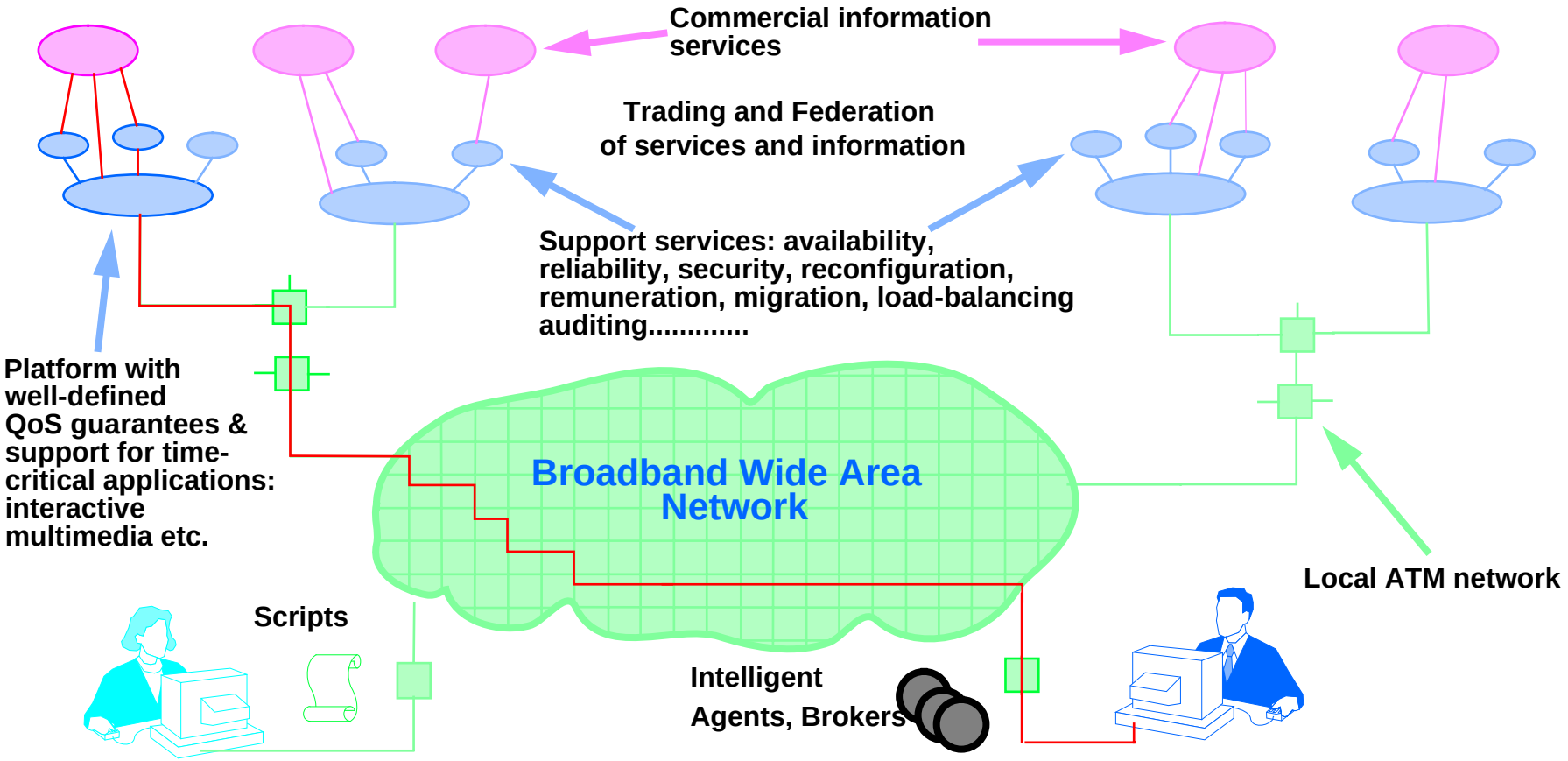
Technology transfer

projects consultancy
secondment training

Lead key standards


ANSA Vision
Work groups
Desktops
Compound documents
Workflow
ELECTRONIC COMMERCE
Open networks
Brokers
WWW
Scripts
Commercenet
Interactive multi-media
Broadband
Quality of Service
SYSTEMS INTEGRATION
Embedded Systems
Home
Transport
Healthcare
Enterprise systems
Databases
Events
Business applications
OPEN DISTRIBUTED PROCESSING
Mobile terminals
Laptops
Agents
PDAs
Offline ops

Electronic Commerce in Action



New Requirements

Interactive Multi-media

Performance

Streams

Real-time scheduling

QoS negotiation

CORBA++

Open Networks

Federated naming

Intelligent broking and trading

Cooperative, autonomous management

Security

MS NETWORK ++

Distributed Information

OLE/COM++

Intelligent information filters and agents

Information servers

Computer assisted business processes

Embedded Systems

Down scaling

Streams

Real-time scheduling

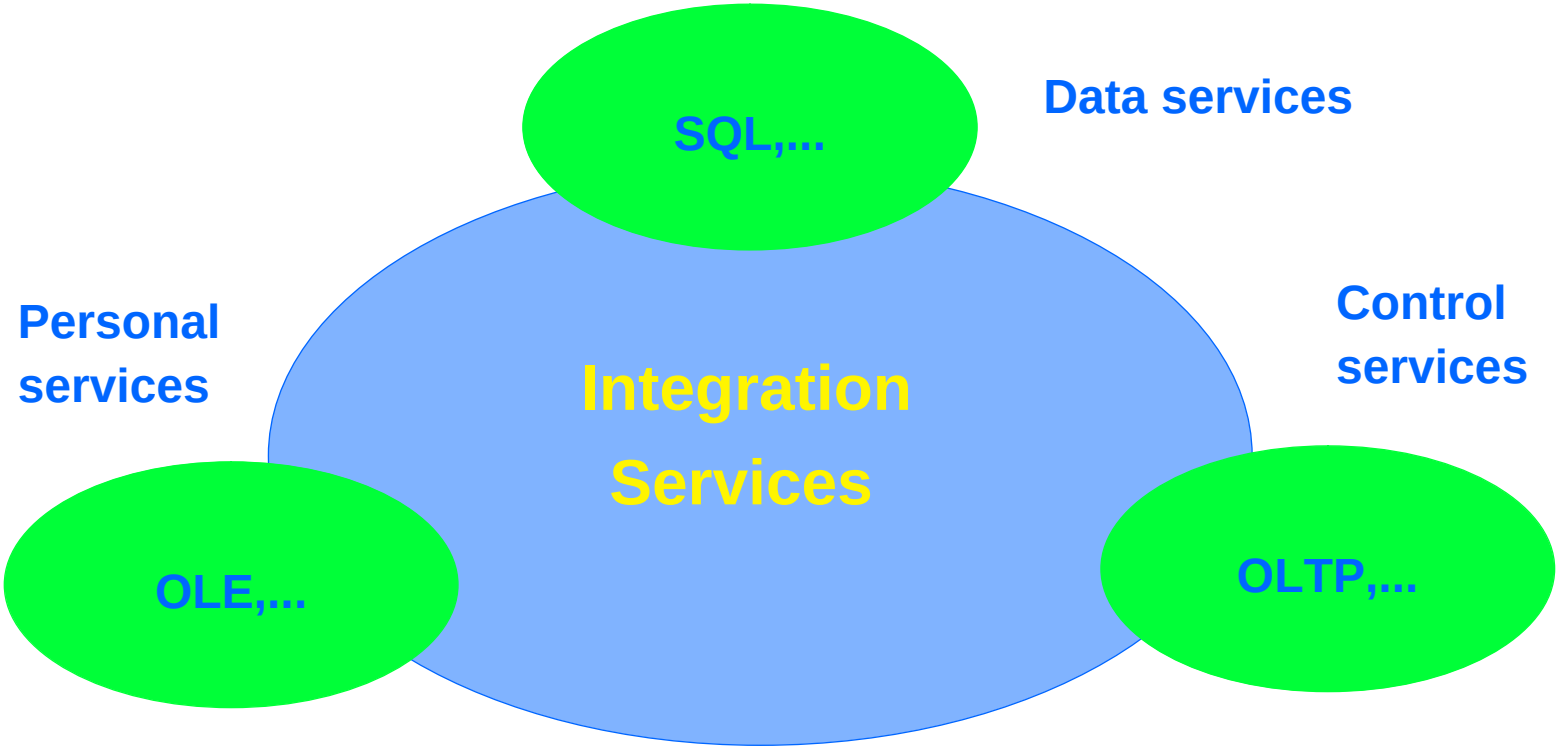
CORBA--

ANSA

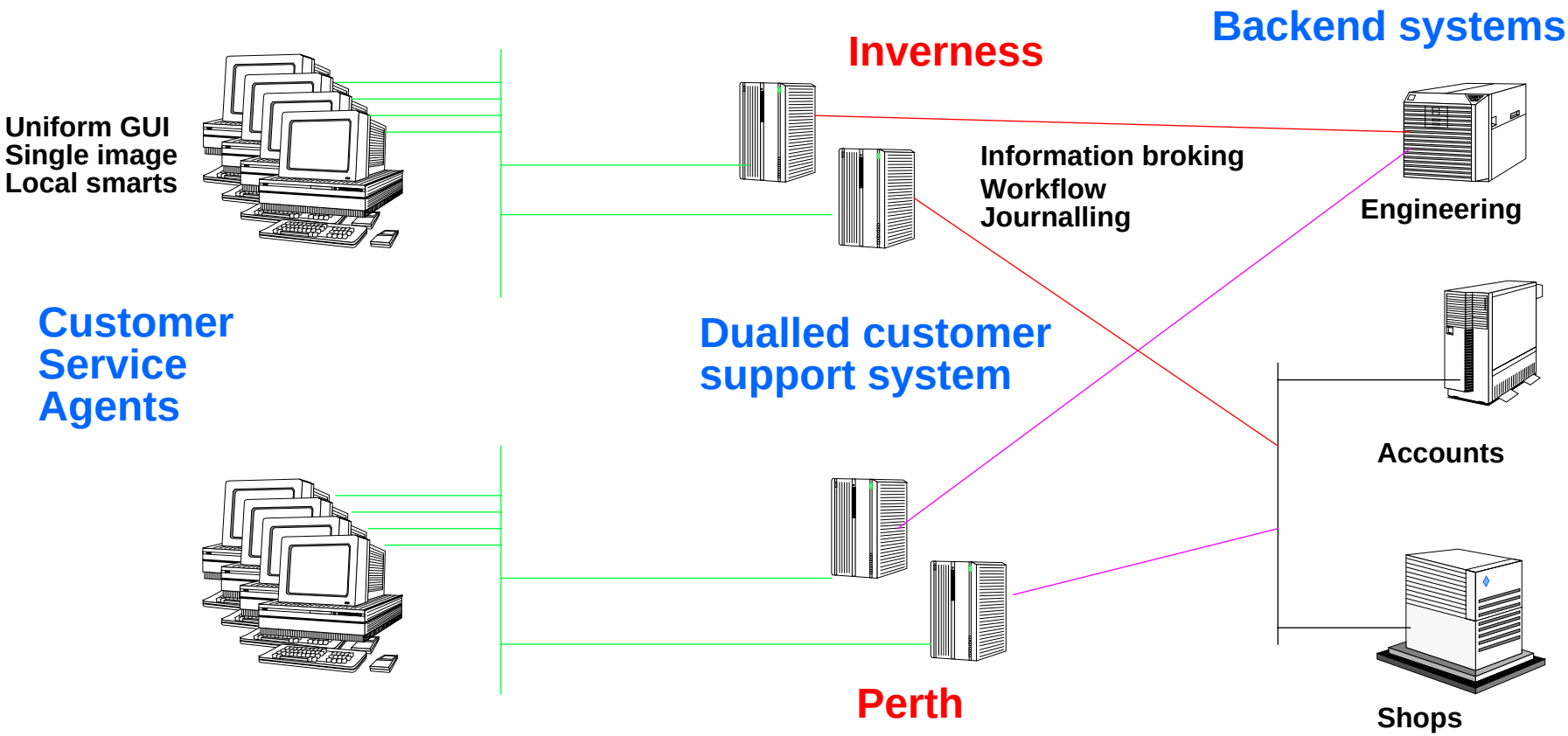
and

Open Distributed Processing

Open Distributed Processing = Integration Services = Interoperability and Management



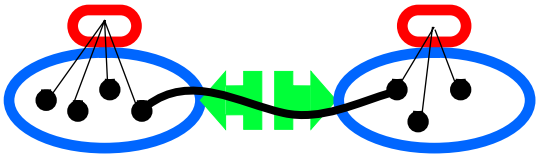
An Example - Scottish HYDRO



The ANSA Architecture

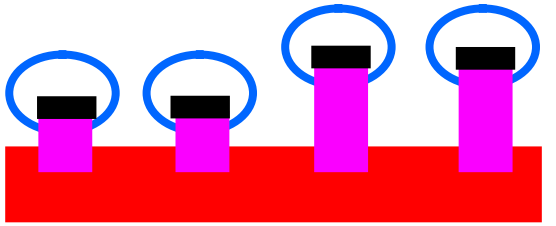
Trading and Federation

Controlled interoperability



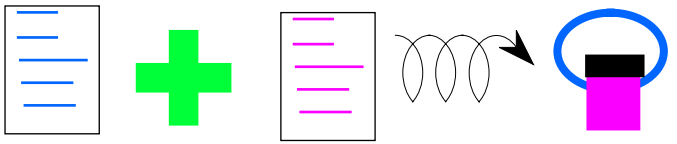
Selective Transparency

One size does not fit all



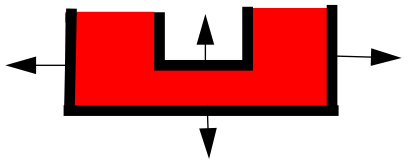
Abstract & Automate

Tools replace APIs

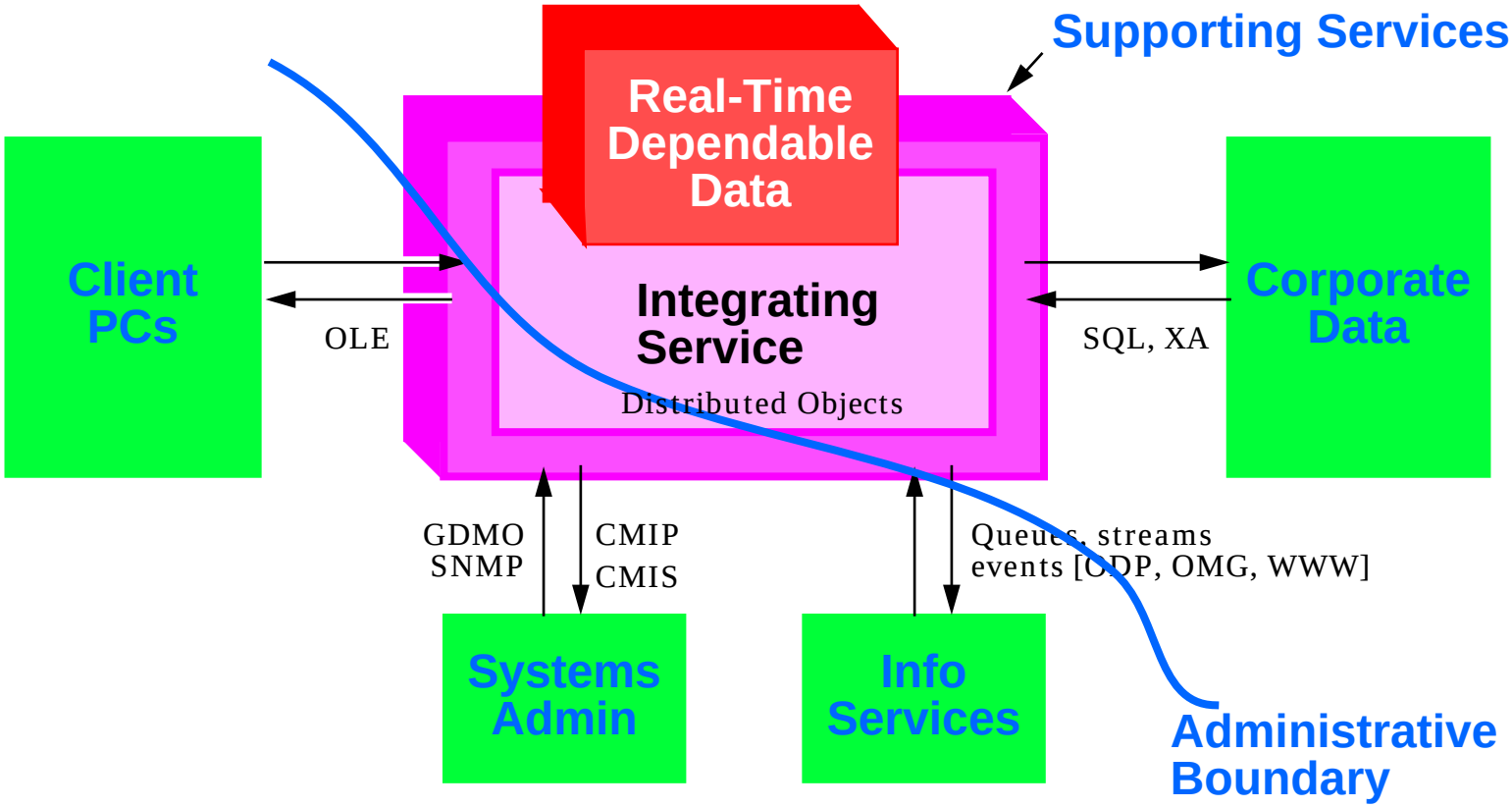


Modular Engineering

Architected internal interfaces



Template for an Integration Service



Federation

- Large systems are made up of autonomous islands interconnected incrementally
  - no central authority
  - legacy of old technology
  - conflicting choices of new technology
- Administrative boundaries
  - where checks and accounting are to occur at the boundary
- Technology boundaries
  - where protocol conversion and data translation are to occur at the boundary
- Set up interceptors (gateways / bridges) on demand, when trading

Types, Trading and Binding

- **Trading - discovering an interface that provides a service**
  - exporters make service offers
  - importers make service requests
  - trading marries imports to exports
- **Binding - knitting together a set of matching interfaces to enable interaction**
  - typically implicit for RPCs
  - necessarily explicit for streams and for RPCs where QoS is to be controlled
- **Type safety - match signatures to ensure valid interaction**
  - carry signatures into runtime system to avoid dependence on repository
  - “no surprises” rule enables federation
  - done when trading, enforced by compiler

Trade-offs in Distributed Systems

- Distributed systems engineering is all about trade-offs
  - ABSTRACTION versus SPECIALIZATION -the more you hide, the less control you have
  - CONSISTENCY versus AVAILABILITY - availability implies copies, increases risk of inconsistency
  - AUTONOMY versus UNIFORMITY -autonomy gives more freedom but leads to differences which increases complexity
  - SECURITY versus CONVENIENCE - security makes things harder to do
- Therefore we need a kit of parts and an open framework into which they slot
- Moreover the framework must accomodate coexistance of alternative parts for the same job

Selective Transparency

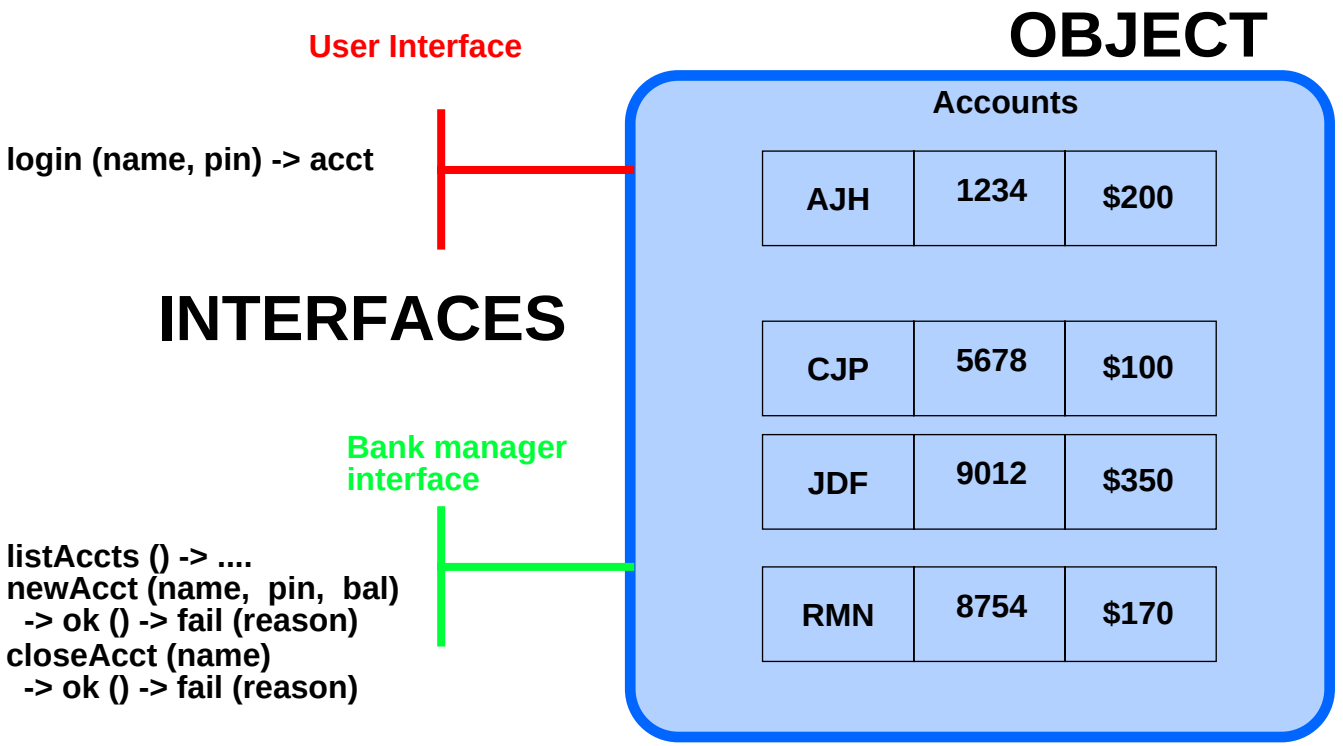
- **Transparency is about hiding irrelevant complexity**
  - **Location** — don't need to know where it is to use it
  - **Access** — don't need to know how it works to use it
  - **Migration** — it can move while you're using it to balance loads or reduce latency
  - **Replication** — there may be copies for reliability and/or availability
  - **Persistence** — it only gets resources when it needs them
  - **Partial Failure** — it always gets to a consistent state
  - **Federation** — you don't have to have the same administrator to use it
- **The transparency layer supports the functionality of the basic service distribution layer, and adds additional guarantees**
  - same API for issuing requests
  - extra management functions for controlling transparency

Distributed Programming is Different

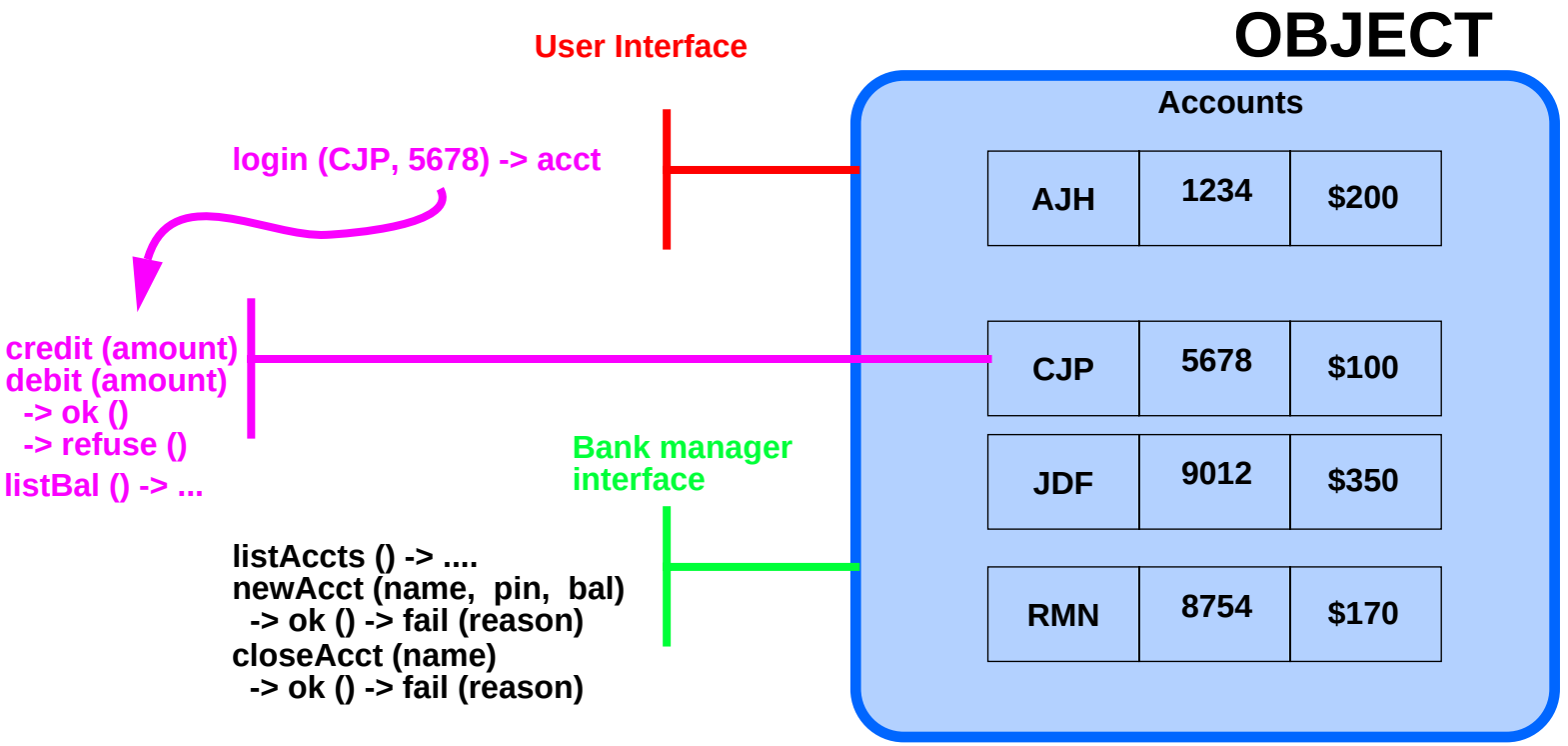
TRADITIONAL	REVERSED
Local	Remote
Sequential	Concurrent
Single Environment	Diverse Environment
Fixed Location	Mobile
Single Copy	Multiple Copies
Synchronous	Asynchronous
Direct	Indirect
Shared	Separate
Global	Context Relative
Complete failures	Partial Failures
Early Binding	Late Binding

- Trying to wedge this into a traditional view won't work

Distributed Object Model (1)



Distributed Object Model (2)



Why Do APIs Grow So Large?

- **Rich set of concepts needed for distributed programming**
  - threads
  - requests
  - replication
  - atomicity
  - .....
- **Optimized engineering for common cases**
  - e.g. forked call -> asynchronous call to save a local thread
- **Special engineering for special cases**
  - e.g. spawned atomic call -> start new top level transaction
- **Combinatorial explosion in functions overwhelms the programmer**

Let Compilers and Tools Take The Strain

- **Exploit abstraction, program in application oriented concepts**
  - **most aspects of OO really help, some hinder**
- **Simple (pre-processor) extensions go a long way**
  - **especially if leveraging an OO language**
- **orthogonality - e.g. “dot” and “bar” *vs*. threads and RPC API**
  - **languages minimize complexity without losing scope for optimization**
- **declarative - state requirements and policies not mechanisms**
  - **point already proven by IDLs and stub generators**
  - **decouple applications from engineering - ANSA PREPC experience**
- **strong type checking for safety and confidence**

Is ANSA Real?

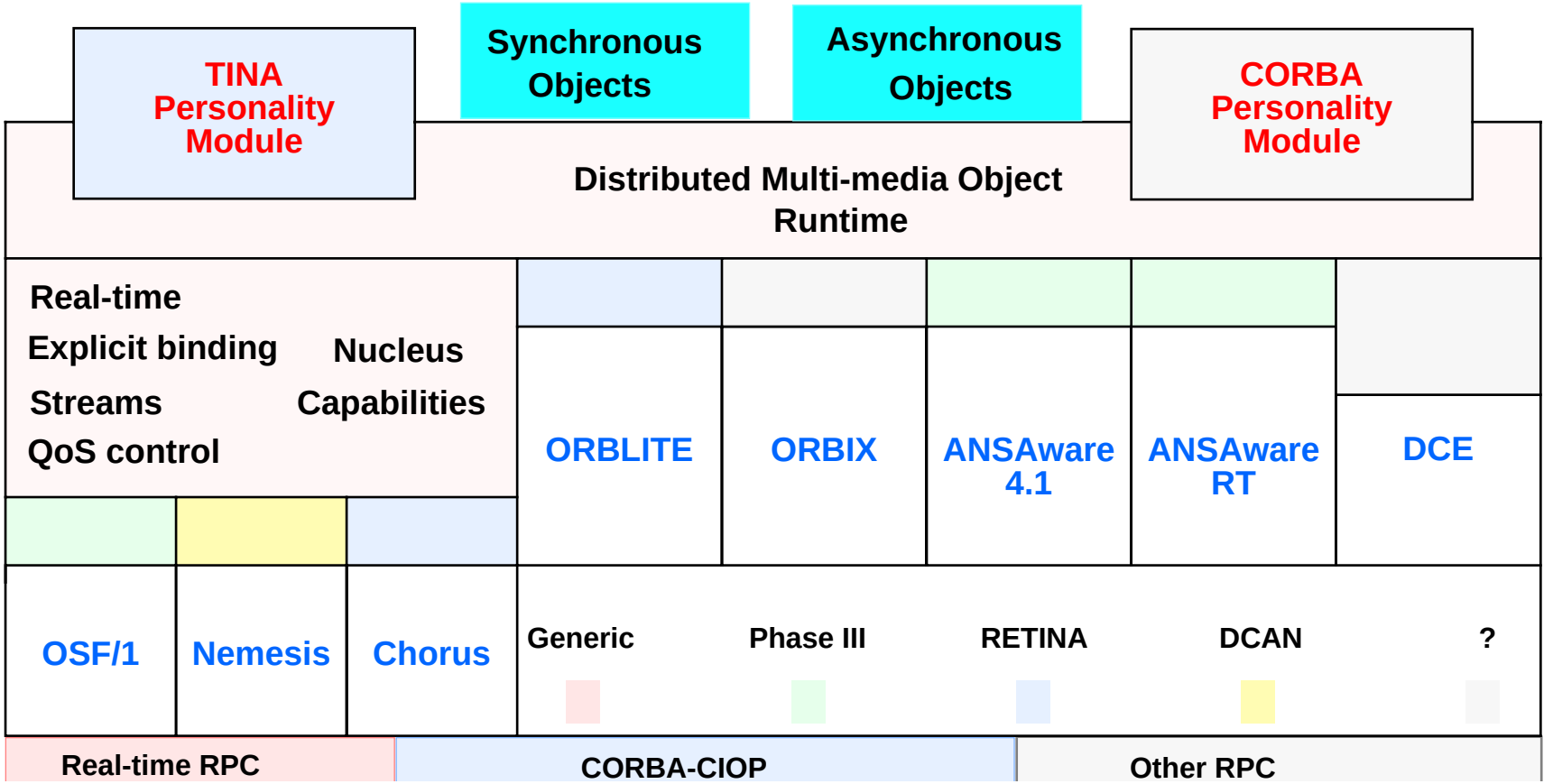
- All except security has been built for the ANSAware prototype
- ANSAware anticipated DCE RPC, IDL and threads by 2 years
- ANSAware anticipated CORBA, object life cycle and UNO by 2 years
- ANSAware has fed into products by HP, ICL, BT, Bellcore, GPT, AEG
- ANSAware ‘93 included DCE RPC and threads coexisting with previous home-grown solutions
- ANSAware ‘94 development majored on real-time and streams
- ANSAware ‘95 development will major on WWW interceptors and repository

ANSA

and

Telecoms

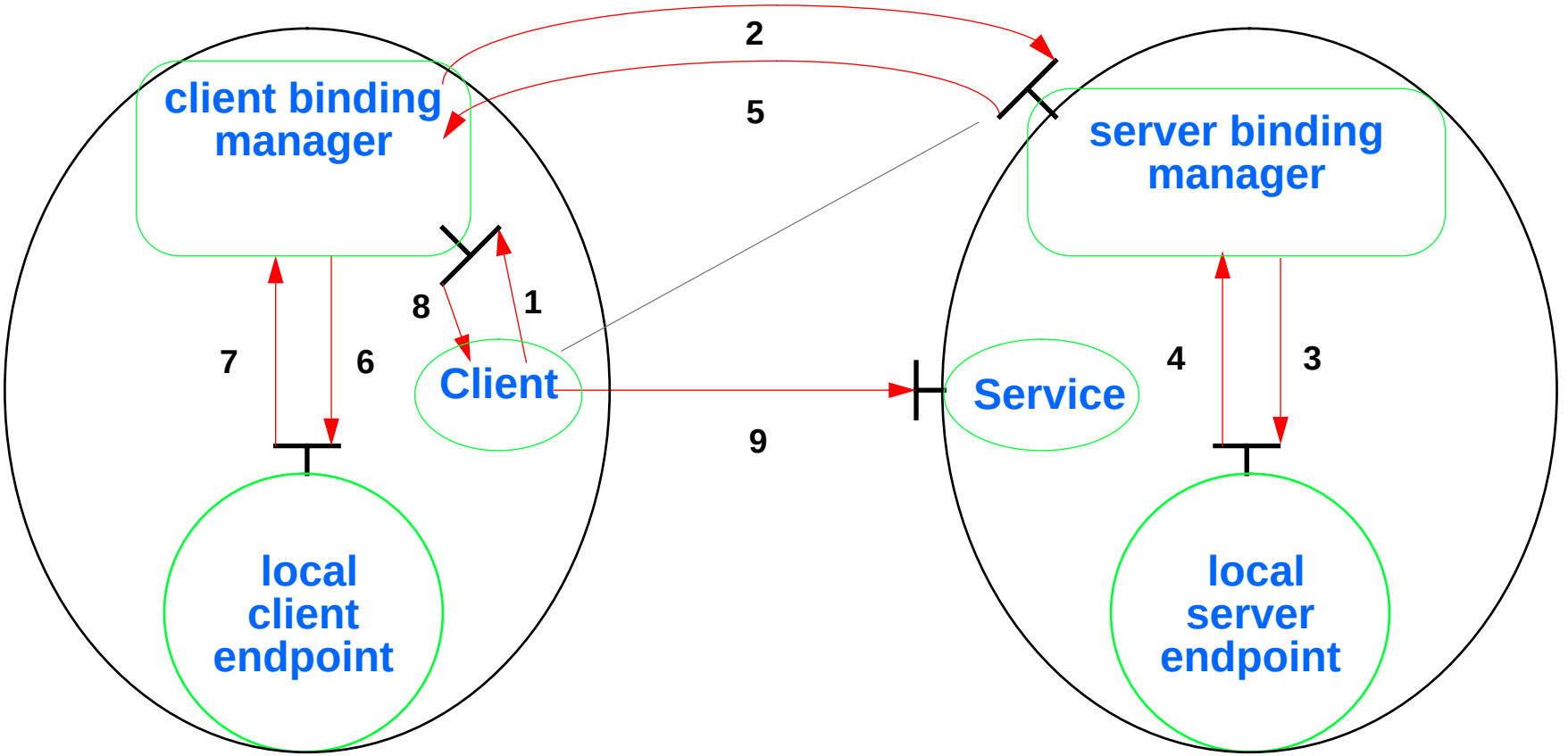
Distributed Interactive Multi-Media Architecture



Streams

- *A stream has a set of flows*
- *A flow has a set of frames (or signals) and a direction*
- *A frame has a name and a set of typed arguments*
- *Streams are typed and can be conformance type checked*
- *Frames are transmitted by non-blocking writes and read by blocking reads*

Explicit Binding



Information Services - Object Wrapping