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

Design for Re-Engineering the Web

ANSA ISF Group

Abstract

This document describes the proposed design for re-engineering the WWW, and the corresponding implementation plan derived from APM.1505.

For the present, the precise details are confidential to the consortium.

APM.1510.02 **Approved** 10th December 1995

Technical Report

Distribution:
Supersedes:
Superseded by:

Design for Re-Engineering the Web

Design for Re-Engineering the Web

ANSA ISF Group

APM.1510.02

10th December 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 Re-Engineering Design

This document describes the proposed design for a new, extended, WWW built over an object architecture and using IIOP as the base protocol. The intention of the design is to fulfil the requirements set out in [ISF 95]. The reader is also referred to [ISF 95] for the motivation behind undertaking this project.

This chapter describes the main parameters of the chosen design, while the following chapter discusses the choices that need to be made in order to implement the systems in terms of these components.

1.1 Evolution Path Overview

It is intended that the design should allow a clear path for development of new services in parallel with continued support for existing services and capabilities. This design consists of three implementation stages. The completion of each stage corresponds to building a new capability into the WWW.

Figure 1.1 provide an overview of the evolution path.

The first stage exploits the existing HTTP-based proxy mechanism to build an HTTP-IIOP gateway (H2I). We will also build an IIOP-HTTP gateway (I2H). This will allow existing web browsers to contact existing web servers using IIOP instead of HTTP. No modifications will be necessary to existing browsers and servers.

In the second stage we will build a web browser and server which have IIOP as a native protocol.

Figure 2.2 shows how the components from stage 1 and stage 2 fit together. Note that an IIOP web server will be able to serve exactly the same file system as an existing HTTP web server.

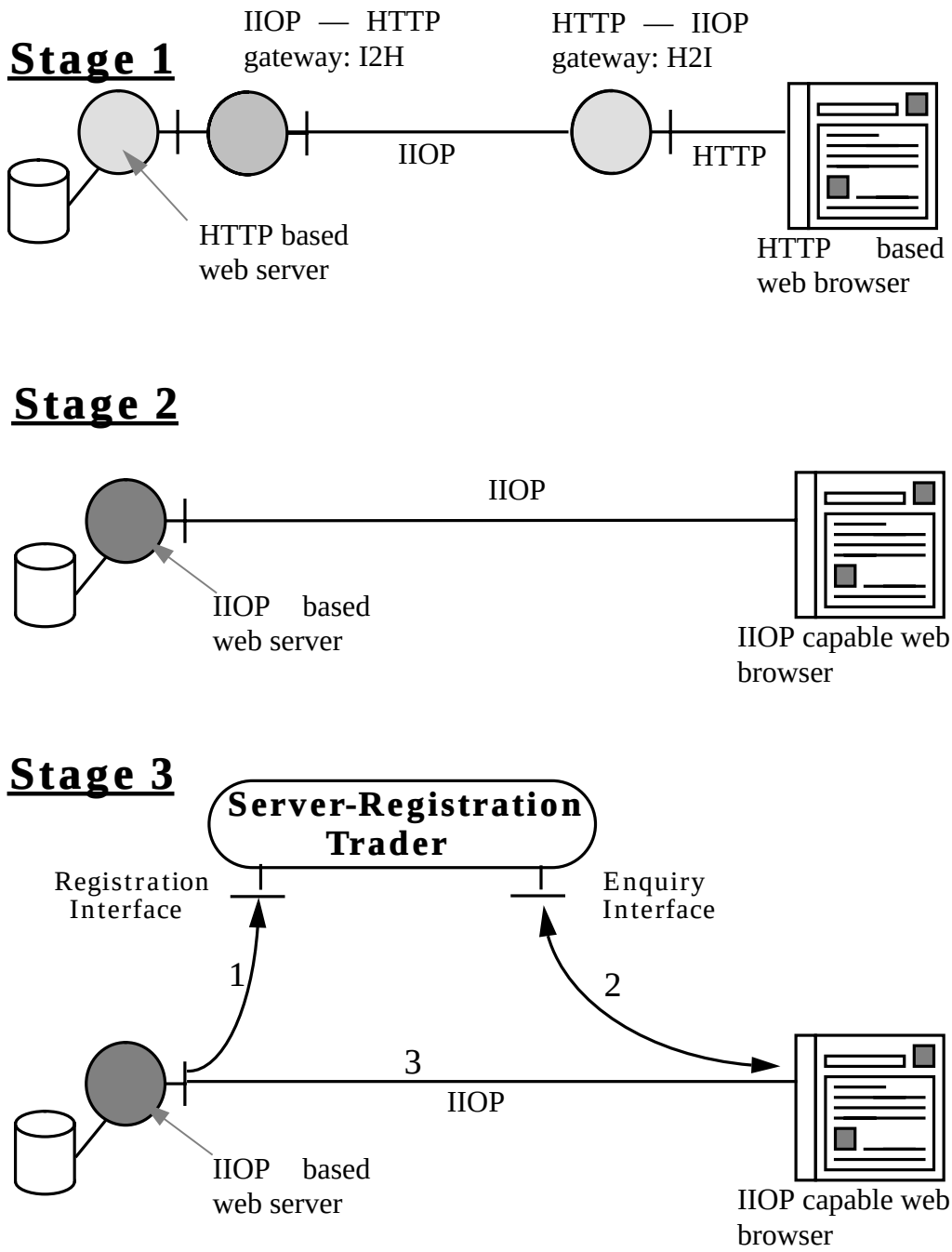
In the third stage we will incorporate trading into the native IIOP browsers and servers as well as the gateways. IIOP servers and IIOP-HTTP gateways will register themselves with a trader. Browsers and HTTP-IIOP gateways will use traders to find IIOP capable gateways and servers which can be used instead of (existing) HTTP-based servers.

It is expected that the work will also extend to a fourth stage, in which demonstration applications will be built that take advatage of the object facilities supported by the re-engineered WWW.

1.2 Gateways

The gateways will be required in the early stages of the re-engineering process - see Figure 1.1. The I2H gateway is an HTTP client and IIOP server. It accepts incoming requests via IIOP and converts them to HTTP requests.

Figure 1.1: The Overview of the Evolution Path



In Stage 1, IIOP is used as the main transport protocol, with the H2I and I2H gateways translating HTTP to IIOP and vice versa, while the clients and servers still communicate using HTTP. In Stage 2, the client and server both have a native IIOP mode and communicate directly with each other. In stage 3, IIOP-capable clients locate IIOP-capable webservers by trading for services which have registered themselves with the trader as IIOP-capable.

The H2I gateway is an IIOP client and HTTP server. It accepts incoming HTTP requests and converts them to IIOP requests.

The I2H and H2I gateways can be located near (in terms of network path) an HTTP server, providing an equivalent service via IIOP. It can also be located near (again, measured in terms of network path distance) to a native IIOP client allowing that client to access resources via HTTP. Locating the

gateways near the client and server respectively allows as much of the network distance as possible to be traversed using IIOP rather than HTTP, thus bringing the largest possible associated performance gain. This strategy also allows existing firewalls with HTTP doorways to continue to be used while still gaining the performance benefits of IIOP over HTTP.

1.3 HTTP Server

Existing standard HTTP servers must be supported throughout the evolution from today's web to an IIOP based web. This requirement can be met by use of the HCIS proxy mechanism shown in figure 2.2.

1.4 HTTP Client

Existing standard HTTP clients must be supported throughout the evolution from today's web to an IIOP based web. This requirement can be met by use of the ICHS proxy mechanism shown in figure 2.2.

1.5 IIOP Server

This is intended to be the IIOP-based equivalent of today's web servers. As described in [ISF 95], the intention of rebuilding servers with IIOP capability is ultimately to enable the encapsulation of existing services. Whereas the present WWW treats everything as a document of some form, in the future it is hoped that more active services will be available using distributed object oriented (CORBA) technology.

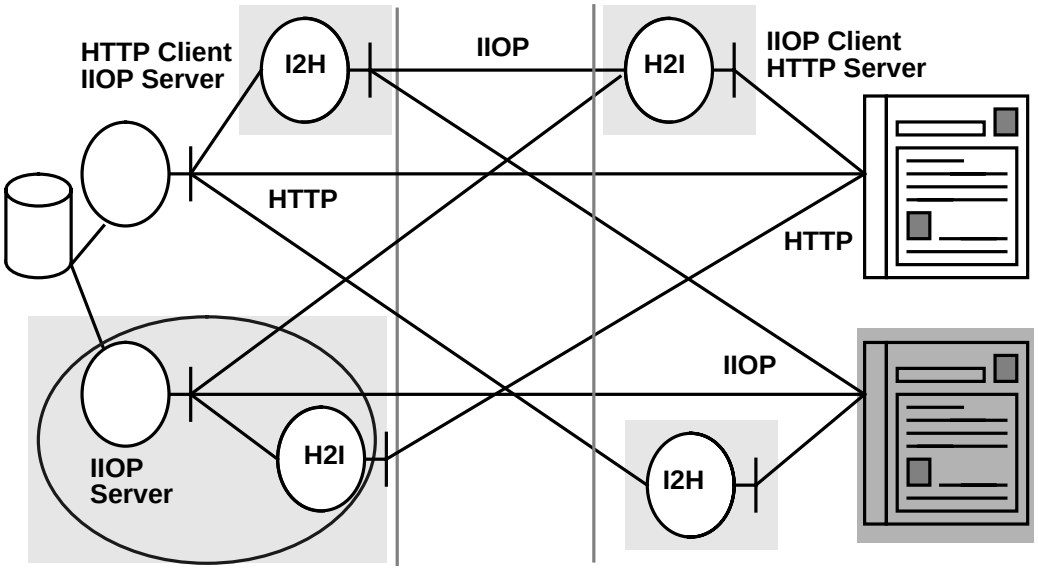
1.6 Common Filestore

This is required on the grounds that, for simplicity of integration with existing WWW services, both HTTP servers and IIOP servers will be able to serve a common file store.

1.7 IIOP Client

This is an IIOP client which can communicate with HTTP and IIOP servers directly. These will be needed in order to benefit as completely as possible from the provision of IIOP-based WWW services.

Figure 1.2: Evolution Path and Legacy Support



2 Stages of the Re-Engineering Plan

This chapter describes the main stages into which the evolution of the plan is divided.

2.1 Evolution Path

The basic division of the evolution path between the existing WWW infrastructure and the IIOP-based object infrastructure described here is in 3 stages:

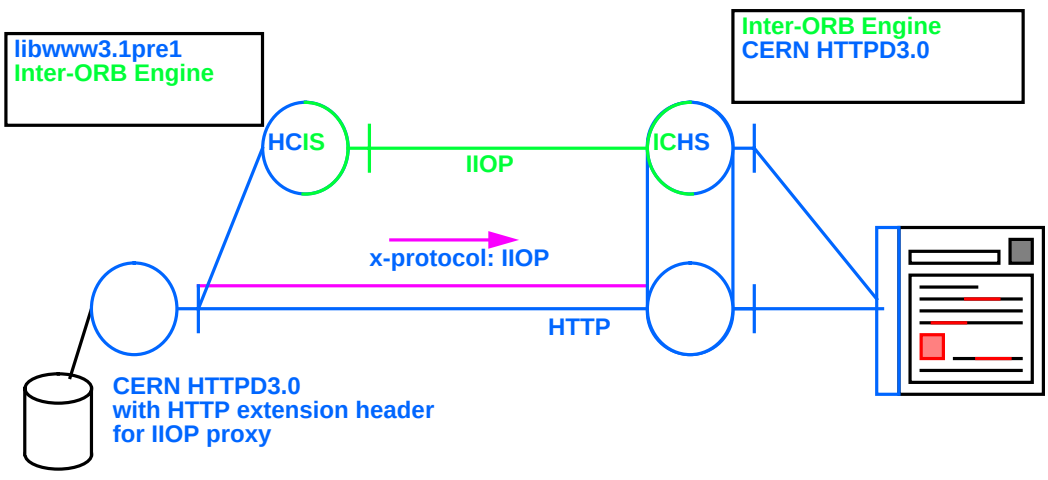
1. Stage 1: Implement gateways/proxies
2. Stage 2: Implement IIOP native webserver and client
3. Stage 3: Implement trading approach for registering IIOP servers

It is envisioned that, after completion of these 3 main stages, the development will proceed further along the avenue of testing and applications avenue. Particular tasks that will be deferred to that level of development are performance comparison of IIOP against HTTP, interfacing to non-WWW services via the Inter-Orb Engine, modelling and finetuning, and the porting of existing ANSA-based applications to use the WWW as their connection layer.

2.2 Evolution Path: Stage 1:

Stage 1 of the evolution path is to implement basic versions of the gateways that will allow HTTP requests and responses to be carried over IIOP. The two gateways are HCIS, the HTTP client IIOP server, and ICHS the IIOP client HTTP server.

Figure 2.1: Evolution Path: Stage 1



The first versions will use the simplest possible mapping to CORBA types, sequence of string pairs, for the whole request or response.

HCIS will be based upon libwww3.1pre1, the WWW Common Code library that is now available to W3C members, and the Inter-ORB engine from Sun.

ICHS will be based on CERN HTTPD version 3.0, as currently in use at APM both for live servers and various experiments. An IIOP protocol module based on the Inter-ORB Engine will be added. A basic protocol choice mechanism will also need to be added.

The ICHS protocol choice mechanism will use data delivered in extension HTTP headers that indicate the availability of an IIOP gateway. CERN HTTPD will also be modified to send the extra HTTP header.

2.3 Provisional Assignment of Stage 1 Tasks

InterORB Engine availability: Mike Beasley

HCIS: Owen Rees

CHIS and Modified HTTPD: Nigel Edwards

2.4 Evolution Path: Stage 2

Stage 2 of the evolution path is to implement the IIOP-native webserver. In stage 1, IIOP is used to transport HTTP requests as a sequence of octet. In stage 2 we will design a CORBA mapping of HTTP to IDL and use IIOP to transport HTTP requests encoded as CORBA data types using this IDL mapping. The advantage of this is that HTTP headers will no longer be sent as ASCII; rather they will be marshalled as a CORBA data type (e.g. enum). In parallel with this we will modify our CORBA stub compiler to drive our IIOP API. This will allow us to experiment with different HTTP to CORBA IDL mappings easily.

As discussed in the requirements document, HTTP responses will be presented as stream objects. Thus instead of returning a sequence of string pairs, the call invocation will return a reference to a stream object in the client's address space. The stream object will have operations to retrieve data and also an operation to abort the stream (destroy the object and stop incoming data).

Once this has been done the IIOP proxies implemented in stage 1 will be adapted to use the new CORBA encoding of HTTP.

In parallel with this the complete system is required to incorporate fragmentation and sessions. This will allow clients and servers to multiplex requests and replies over the same TCP connection. Routines to manage sessions and the degree of multiplexing will be provided. The major issue which this approach raises is whether these modifications can be made within the IIOP standard. If not, then the technical issues resulting from dealing with user-space threading are expected to be challenging, although not insurmountable. The main challenge is to maintain on-the-wire format compatibility with IIOP using the layering shown in figure 2.3.

We will also write a native IIOP webserver which will serve a web file system using either HTTP/IIOP or plain vanilla HTTP. This will be a multi-threaded server. We should be able to demonstrate big performance gains serving

Figure 2.2: Evolution Path: Stage 2

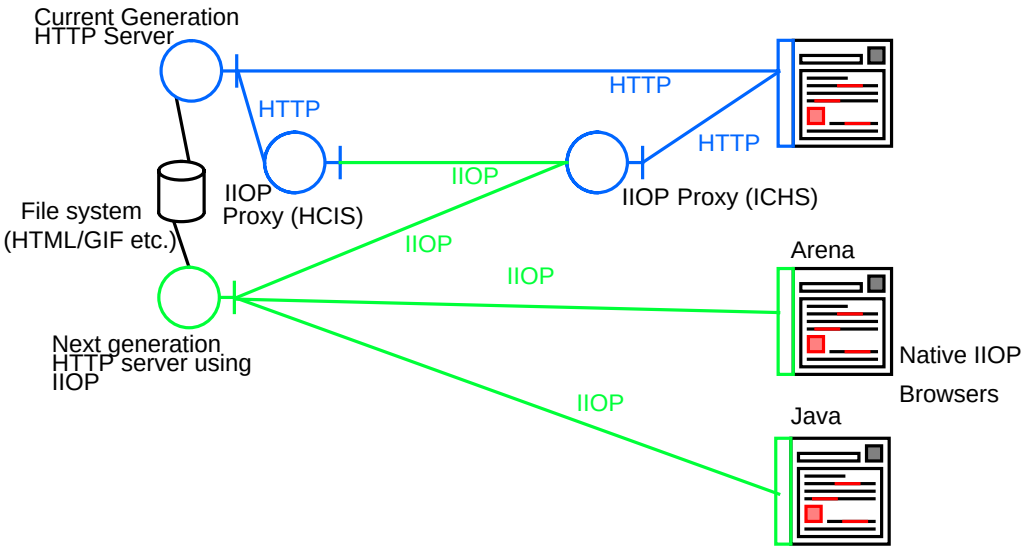
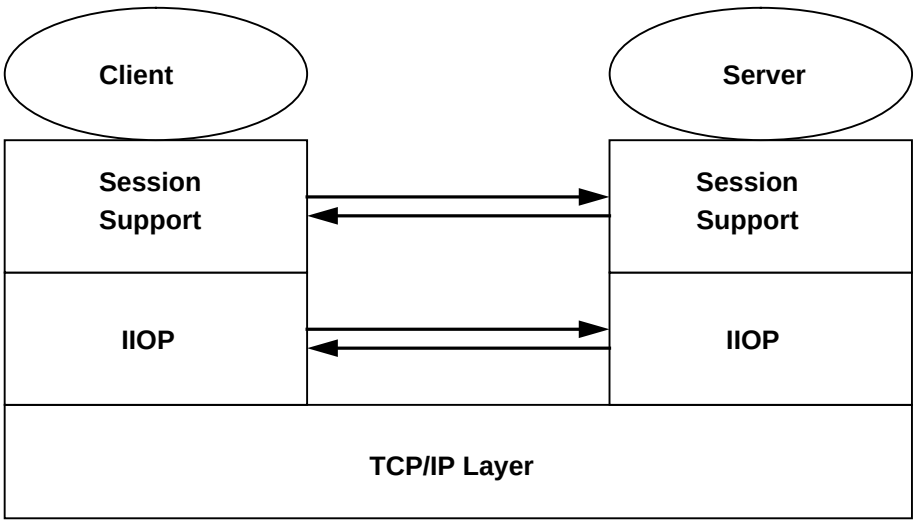


Figure 2.3: Session Management over IIOP



documents using IIOP, through efficient encoding of the HTTP headers as CORBA data types.

The webserver will allow webmasters to define new HTTP methods using IDL. The stub compiler will generate a new set of IDL skeletons for the server and template HTML forms to drive the new method. For backwards compatibility we will support CGI (perhaps incorporating the stub compiler for CGI [EDWARDS 95]). This will have the particular advantage of providing an upgrade path for WWW appliccation development path will result in WWW applications that can easily be moved onto the IIOP infrastructure. Ultimately

it is planned to incorporate other interfaces, notably a separate management interface, within the server.

Eventually a threaded browser will be needed to demonstrate the full performance gains using this technology. Ideally, this should provide comparable performance to that of Netscape Navigator. The browsers for which we are able to obtain source code are NCSA Mosaic, the CERN linemode browser, HotJava, and the W3C's Arena. Mosaic is non-threaded and depends on a heavily customised version of libwww, the linemode browser is intended only for experimental and developmental use, leaving Arena, which does support partial rendering of HTML objects, and HotJava, which does have full threads support. Arena is not genuinely threaded: instead it simulates threading by using nonblocking I/O. HotJava is presently restricted to Solaris and Windows NT platforms. We will therefore reserve the option of adding an HTTP/IIOP protocol module to HotJava and to either Arena or Mosaic. The choice between Arena and Mosaic will depend upon a study of the client source to decide whether to exploit Arena's I/O architecture for substitute threading, or to slice a new (threaded) version of libwww under the Mosaic source. [1]

Initial indications are that Arena is still too much an experimental prototype on which to base a distribution. It is also presently under a restrictive source licence while Mosaic will at least be quite simple to move onto IIOP. The latter will be accomplished by slicing in a level of indirection to the Mosaic source tree: this can be done without serious modification to the source maintenance and build tree, but will require a considerable commitment of time for the reasons outlined above.

We will implement an HTTP/IIOP protocol module for HotJava by wrapping our Web API for the SUN Inter-Orb Enginein a set of "native Java methods", and writing Java classes around these native methods to implement the URL to IIOP path. The native method code will be linked into, and the new classes loaded into, the Java run-time environment as needed (i.e. as soon as the HotJava browser wants to use the IIOP protocol).

It will be possible to deliver and install the new IIOP Java classes over the Net automatically, but it won't be possible to do the same with the native method code part. This is because Sun's Inter-Orb Engine is not written in Java so it cannot be auto delivered and installed. (This is unless somebody writes an Inter-Orb engine in Java: possible for someone with detailed knowledge of IIOP.)

2.5 Issues

The requirements document discusses two options for discovering whether an HTTP server has an IIOP facility: modifying the HTTP server to return a header containing its IIOP object reference, or trading. The advantage of the trading approach is transparency: legacy servers do not need to be modified. The disadvantage is complexity. At this stage we are still using the simpler, non-transparent approach of modifying the server to return a new header. We defer trading to stage 3.

1. Note that this is likely to be one of the harder tasks, since the interface between most clients and libwww is not well defined. However, using the amount of effort allocated, at least one of these tasks can be completed within the planned timeline.

2.6 Provisional Assignment of Stage 2 Tasks

Modifying Stage 1 proxies: Owen Rees and Nigel Edwards

Arena & Mosaic, libwww interface: Mark Madsen

Java, IIOP module: Ashley McClenaghan

IIOP fragmentation and API: Mike Beasley

IIOP Server: Owen Rees

Stub Compiler: Nigel Edwards

HTTP/IDL mapping: To be decided (a simple mapping by Owen Rees already exists)

2.7 Evolution Path: Stage 3

Implement the trading approach for registering IIOP web servers.

Stage 3 tasks are not presently assigned.

2.8 Acknowledgements

The ANSA Information Services Framework (ISF) Group consists of Owen Rees, Ashley McClenaghan, Mark Madsen, Nigel Edwards and Mike Beasley. They would like to acknowledge the contributions of all their colleagues on the ANSA Team and the ANSA Technical Committee.

Glossary

This glossary contains expansions and explanations of all the acronyms used in the body of this document.

ANSA

Advanced Networked Systems Architecture: The architecture designed by the ANSA Project at APM Ltd.

CGI

Common Gateway Interface: The de facto standard interface for accessing backend code from an HTML Form.

CORBA

Common Object Request Broker Architecture: The OMG standard architecture for any ORB.

DIMMA

Distributed Interactive Multimedia Architecture: Designed as part of Phase III of the ANSA Project.

GIOP

General Interoperability Protocol: The OMG standard for interoperability protocols.

HTML

Hypertext Markup Language: The standard WWW document architecture

HTTP

Hypertext Transfer Protocol: The standard HTML document transfer protocol used within the WWW.

HTTPng

Hypertext Transfer Protocol, the Next Generation: An enhanced version of HTTP proposed as a new standard.

IDL

Interface Definition Language.

IETF

Internet Engineering Task Force: The group responsible for development and investigation of Internet standards.

IIOP

	Internet Interoperability Protocol: Standard defined by the OMG as the Internet-based realisation of GIOP.
MIME	
	Multimedia Internet Message Extensions: The Internet standard for inclusion and description of new media types within existing message and document formats.
ODP	
	Open Dsitributed Processing: The ISO standard 7-layer model (ISO 10746).
OMG	
	Objects Management Group: A consortium with a mission to define standards for interoperability of object-based open systems.
ORB	
	Object Request Broker: An object which matches requests from client object for specified types of service capability to known services.
RPC	
	Remote Procedure Call.
SSL	
	Secure Sockets Layer: A standard defined by Netscape Communications Corporation for secure transmission of commercial information via the WWW.
TCP	
	Transport Control Protocol: The Internet standard transport protocol.
URC	
	Uniform Resource Characteristic: A proposed standard syntax and encoding for metadata about objects contained within the WWW.
URL	
	Uniform Resource Locator: The specification of the location of an object within the WWW expressed in an IETF standard syntax.
WWW	
	World Wide Web: A collection of documents, objects, and services interoperating via a collection of common protocols supported by the Internet infrastructure.

References

[ISF 95]

ANSA Information Services Framework Group, "*Requirements for Re-Engineering of the Web*", **APM.1505**, APM Ltd., Cambridge U.K., June 1995.

[EDWARDS 95]

Nigel Edwards, "A Stub Compiler for CGI and HTTP: The Programmer's Guide", **APM.1465**, APM Ltd., Cambridge U.K., June 1995.