IMS General Web Services Base Profile

Version 1.0 Final Specification

Document Name: IMS General Web Services Base Profile
Date: 19 December 2005

Executive Summary

The General Web Service Base Profile provides a basic structure for the definition of Web Services. It consists of a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications that promote interoperability. The General Web Services Base Profile addresses the most common problems experienced when implementing web service specifications. The General Web Services Base Profile defines the selection of mechanisms within referenced specifications that are well understood, widely implemented and useful.

The General Web Services Base Profile promotes interoperability for web service based specification implementations on different software and vendor platforms. The Base Profile focuses on a core set of web service specifications and the most common problems experienced implementing the identified web service specifications. It is not a goal of the General Web Services Base Profile to create a plug-and-play architecture for web services or to guarantee complete interoperability. The General Web Services Base Profile addresses interoperability in the application layer, in particular, the description of behaviors exposed via Web Services.

The General Web Service Base Profile is derived from the Web Services Interoperability Basic Profile v1.1 and the Web Services Interoperability Simple SOAP Binding Profile v1.0. The IMS Global Learning Consortium (IMS/GLC) recommendations for the General Web Service Base Profile are to adopt:

- XML Schema V1.0 – all data models in IMS specifications will be defined in terms of XML Schema (XSD);
- HTTP V1.1 – the Hypertext Transfer Protocol (HTTP) is the mandated protocol binding for the SOAP messages;
- SOAP V1.1 – SOAP is the mandated messaging protocol;
- WSDL V1.1 – an instance of the service is defined using Web Services Description Language (WSDL) v1.1.

The General Web Service Base Profile can be extended by the adoption of one or more support General Web Service profiles. Other IMS General Web Service documents describe profiles for Addressing (transport-neutral web service addressing), Attachments (sending non-XML documents with the SOAP messages) and Security (secure data exchange).

In principle, the SOAP-based binding for the web services supports many communications messaging models (the Information Model for an IMS/GLC specification is defined independently of the messaging nature, i.e., this is determined by the form of the Web Services Description Language binding). At the current time only one messaging model is supported:

- Synchronous – this is a request/response message exchange in which the service initiator is blocked until the response message is received

Further messaging models will be added as and when required, i.e., asynchronous, polled, and publish and subscribe. There are three methods by which the functional capability of the base profile can be extended:

- Addition of new SOAP messages – the addition of new business transactions and the usage of new messaging models requires the creation of new SOAP messages;
- Extensions to the SOAP header – the current IMS General Web Service Base Profile makes use of the SOAP header to contain the application-to-application transaction status information. It is recommended that proprietary extensions to the SOAP header maintain the current usage patterns;
- Extensions in the data contained within the SOAP body – the SOAP body contains the XML instance that is used to represent the parameters defined for the transaction operations in the specification. There may be a need to add new parameters or to extend the XML structures of the current parameters.

The IMS General Web Service documentation is augmented with a substantial set of support tools that enables organizations to create their own Web Services Description Language bindings based upon the IMS General Web Service. Other materials enable .NET and J2EE implementations to be built based upon reference implementations of the IMS General Web Service.

Table of Contents

1. Introduction

1.1 Scope and Context

The objective of the General Web Services activity is to provide a framework for guiding project teams looking to use web services as part of IMS Global Learning Consortium (IMS/GLC) specification development. A General Web Services service binding will provide a methodology and an application profile that meets the following criteria:

- Interoperability – Artefacts produced under the General Web Services activity will seek to identify mechanisms and standards that promote interoperability between web service specification implementations across different software and operating system platform;
- Efficiency – Artefacts produced under the General Web Services activity will be designed to help other IMS project teams efficiently and effectively evaluate web services protocols as they pertain to the functional requirements of the project team;
- Consistency – Artefacts produced under the General Web Services activity will be designed to facilitate the implementation of consistent approach to the implementation of web service protocols across IMS project teams and specifications;
- Flexibility – Artefacts produced under the General Web Services activity will be flexible enough to adapt to evolving web service protocols such as 'SOAP'/'SOAP with message attachments' and to work with a variety of binding methods for web services such as Web Services Description Language (WSDL);
- Practicality – Artefacts produced under the General Web Services activity will seek to facilitate vendor's ability to implement IMS/GLC based Web Service solutions and interoperability across platforms and vendor implementations of web service protocols.

The General Web Services Base Profile (GWSBP) provides a basic structure for the definition of Web Services. It consists of a set of non-proprietary Web Services specifications, along with clarifications and amendments to those specifications that promote interoperability. The General Web Services Base Profile addresses the most common problems experienced implementing web service specifications. The General Web Services Base Profile defines the selection of mechanisms within referenced specifications that are well understood, widely implemented and useful. The General Web Services Base Profiles will contain syntax conventions consistent with IMS Specification Development Methodology [SpecDev, 03] and Abstract Framework [AbsASCs, 03], [AbsGloss, 03], [AbsWhite, 03].

The General Web Services Base Profile promotes interoperability across web service based specification implementations on different software and vendor platforms. The Base Profile is focused on a core set of web service specifications and the most common problems experienced implementing the identified Web Service specifications. It is not a goal of the General Web Services Base Profile to create a plug-and-play architecture for web services or to guarantee complete interoperability. The General Web Services Base Profile addresses interoperability in the application layer, in particular, the description of behaviors exposed via Web Services. The General Web Services Base Profile assumes the interoperability of the lower layer protocols is sufficient.

Included in this document is a set of recommendations and best practices including how to extend the General Web Services Base Profile. The extensions will have guidance for creating Web Services binding for the Abstract Framework's common services layer such as error handling, security, manifest, context and routing. IMS/GLC has defined a complementary set of profiles that build upon the Base Profile to incorporate more of the Web Services standards and specification developed by the World Wide Web Consortium (W3C). Therefore, this document should be read in conjunction with the set of IMS General Web Services extension profiles [GWS, 05a], [GWS, 05b], [GWS, 05c] and the IMS General Web Services WSDL Binding Guidelines [GWS, 05d]. The terms of reference for the creation of both documents are defined in the original project charter [GWS, 03a].

1.2 Structure of this Document

The structure of this document is:

2. Context for GWS Profiles	The context of the set of profiles, base plus others, that are used to realize Web Service bindings of IMS specifications;
3. Base Profile Rules	The mapping of the WS-I Basic Profile v1.1 rules to the requirements of the IMS GWS, i.e., there are some differences between the WS-I Basic Profile and the IMS GWS Base Profile;
4. Relationship to the Abstract Framework	A description of how the Profiles are related to the Abstract Framework. This explains how the application and common services features are supported through a web service binding;
5. Implementation Profiles	A discussion of web service implementation details, including issues surrounding the type of messaging model and inter and intra organization communication;
6. Best Practice Recommendations	A discussion on the best practices that should be adopted when supporting exception handling, synchronous messaging, secure architectures and other architectural-based features;
7. Extending the Base Profile	A brief discussion of the ways in which the Base Profile can be extended to support proprietary requirements and an indication of future developments in these profiles;
8. Conformance to the Base Profile	An explanation of how conformance for systems that use the Base Profile can be demonstrated. This explains the usage of the IMS Application Profiling Guidelines;
Appendix A – Glossary of Terms	Definition of the concepts, terms and technologies used within this document. This material complements the Abstract Framework Glossary;
Appendix B – Messaging Questionnaire Template	The messaging questionnaire template that is used to define the binding state space for a particular specification.

1.3 Nomenclature

a-API	Abstract Application Programming Interface
API	Application Programming Interface
BPEL4WS	Business Process Execution Language for Web Services
CORBA	Common Object Request Broker Architecture
CRUD	Create, Read, Update and Delete
DCOM	Distributed Component Object Model
ebXML	Electronic Business XML
FTP	File Transfer Protocol
GWSBP	General Web Services Base Profile
HTTP	Hypertext Transport Protocol
HTTPS	Secure Hypertext Transport Protocol
IAF	IMS Abstract Framework
IMS/GLC	IMS Global Learning Consortium
IPSec	IP Security
LAN	Local Area Network

IIOP	Internet Inter-ORB Protocol
MIME	Multipurpose Internet Mail Extensions
MOM	Middleware Oriented Messaging
MTOM	Message Transmission Optimization Mechanism
POS	Point of Service
REST	Representational State Transfer
RPC	Remote Procedure Call
SAML	Security Assertion Mark-up Language
SAP	Service Access Point
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SOAPwA	SOAP with Attachment
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UDDI	Universal Description Discovery & Integration
UML	Unified Modelling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
VLE	Virtual Learning Environment
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
WS-CDL	Web Services Choreography Description Language
WSA	Web Service Architecture
WSDL	Web Services Description Language
WS-I	Web Services Interoperability Organization
XMI	XML Metadata Interface
XML	Extensible Mark-up Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

1.4 References

[AbsASCs, 03] *IMS Abstract Framework: Applications, Services & Components v1.0*, Ed. C.Smythe, IMS/GLC, July 2003.

[AbsGloss, 03] *IMS Abstract Framework: Glossary v1.0*, Ed. C.Smythe, IMS/GLC, July 2003.

[AbsWhite, 03] *IMS Abstract Framework: White Paper v1.0*, Ed. C.Smythe, IMS/GLC, July 2003.

[APG, 05a] *IMS Application Profile Guidelines Whitepaper: Part 1 Management Overview*, S.Wilson and K.Riley, Version 1.0, IMS/GLC, October 2005.

[APG, 05b]	*IMS Application Profile Guidelines Whitepaper: Part 2 Technical Manual*, S.Wilson and K.Riley, Version 1.0, IMS/GLC, October 2005.
[BPEL, 03]	*Business Process Execution Language for Web Services*, T.Andrews, F.Curbera, H.Dholakia, Y.Goland, J.Klein, F.Leymann.K.Liu, D.Roller, D.Smith, S.Thatte, I.Trickovic and S. Weerawarana, V1.1, BEA Systems, IBM, Microsoft, SAP and Siebel Systems, May 2003.
[Cockburn, 01]	*Writing Effective Use-case*, A.Cockburn, Addison-Wesley, 2001, ISBN 0-201-70225-8.
[ebXML, 01]	*Message Service Specification ebXML Transport, Routing & Packaging*, ebXML, Version 1.0, May 2001.
[GWS, 03a]	*General Web Services Project Team Charter*, C.Schroeder, R.Kleinman and S.Griffin, IMS/GLC, June 2003.
[GWS, 05a]	*IMS General Web Services Addressing Profile Final Release*, C.Schroeder, J.Simon and C.Smythe, V1.0 IMS/GLC, December 2005.
[GWS, 05b]	*IMS General Web Services Attachments Profile Final Release*, C.Schroeder, J.Simon and C.Smythe, V1.0 IMS/GLC, December 2005.
[GWS, 05c]	*IMS General Web Services Security Profile Final Release*, C.Schroeder, J.Simon and C.Smythe, V1.0 IMS/GLC, December 2005.
[GWS, 05d]	*IMS General Web Services WSDL Binding Guidelines Final Release*, C.Schroeder, J.Simon and C.Smythe, V1.0 IMS/GLC, December 2005.
[GWS, 05e]	*IMS Binding Auto-generation Toolkit Manual*, C.Smythe, V1.0 IMS/GLC, December 2005.
[GWS, 05f]	*IMS General Web Services UML to WSDL Binding Auto-generation Guidelines Public Draft*, C.Schroeder, S.Raju and C.Smythe, V1.0 IMS/GLC, January 2005
[MTOM, 05]	*SOAP Message Transmission Optimization Mechanism*, http://www.w3.org/TR/2004/CR-soap12-mtom-20040826/, August 2004.
[SOAP, 01a]	*SOAP Messages with Attachments*, W3C, W3C Note 11, December 2000.
[SOAP, 03a]	*SOAP Version 1.2 Part 1: Messaging Framework*, W3C, W3C Final Specification, June 2003.
[SOAP, 03b]	*SOAP Version 1.2 Part 2: Adjuncts*, W3C, W3C Final Specification, June 2003.
[SpecDev, 03]	*IMS Specification Development Methods & Best Practices Draft v5.0*, C.Smythe, IMS/GLC, Sept 2003.
[WSA, 03]	*Web Services Architecture*, D.Booth, H.Haas, F.McCabe, E.Newcomer, M.Champion, C.Ferris and D.Orchard, http://www.w3.org/TR/ws-arch/ W3C, W3C Working Draft, August 2003.
[WSAddress, 04]	*W3C WS-Addressing Submission*, http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810, W3C, August 2004.
[WSDL, 01]	*Web Services Description Language*, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, Version 1.1, W3C, W3C Note, March 2001.
[WSDL, 04]	*Web Services Description Language*, Version 2.0, W3C, W3C Working Draft 3, August 2004.
[WSI, 03]	*Web Services Interoperability Basic Profile Version 1.0a*, Eds K.Ballinger, D.Ehnebuske, M.Gudgin, M.Nottingham and P.Yendluri, Web Services-Interoperability Organization, August 2003.
[WSI, 04a]	*Web Services Interoperability Basic Profile Version 1.1*, Eds K.Ballinger, D.Ehnebuske, C.Ferris, M.Gudgin, C.K.Liu, M.Nottingham and P.Yendluri, Web Services-Interoperability Organization, August 2004.
[WSI, 04b]	*WS-I Attachments Profile Version 1.0*, Eds C.Ferris, A.Karmarkar and C.K.Liu, Web Services-Interoperability Organization, August 2004.
[WSI, 04c]	*WS-I Conformance Claim Attachment Mechanisms Version 1.0*, Eds M.Nottingham and C. von Riegen, Web Services-Interoperability Organization, November 2004.
[WSI, 04d]	*WS-I Simple SOAP Binding Profile Version 1.0*, Ed M.Nottingham, Web Services-Interoperability Organization, August 2004.

[WSR, 03]	*Web Services Reliability (WS-Reliability)*, Version 1, OASIS Specification, OASIS, January 2003.
[XML, 04]	*Extensible Markup Language (XML) 1.0 (Third Edition)*, T.Bray, J.Paoli, C.M.Sperberg-McQueen, E.Maler and F.Yergeau, W3C, W3C Recommendation, February 2004.
[XSD, 01]	*XML Schema Part 0: Primer*, Ed. D.C.Fallside, W3C, W3C Recommendation, May 2001.

2. Context for the GWS Profiles

2.1 Syntax Conventions

2.1.1 Recommendations from the IMS Abstract Framework

All IMS service-oriented specifications will be defined in the context of the IMS Abstract Framework [AbsWhite, 03], [AbsASC, 03], [AbsGloss, 03]. Web services are one possible binding of an IMS service-oriented specification. Section 4 of this document describes how an IMS specification as defined by an Information Model is related to web services through the appropriate 'Infrastructure Binding'.

2.1.2 Recommendations from the International Conformance Program

The IMS International Conformance Program is responsible for defining how a particular information model and binding is constrained to support a particular domain in the form of an Application Profile Guidelines [APG, 05a], [APG, 05b]. Support for Conformance Testing will be added where appropriate, e.g., the WS-I enables conformance statements to be added to a WSDL file.

2.1.3 Recommendations from the IMS Binding Auto-generation Toolkit

The IMS Binding Auto-generation Toolkit document describes how the Unified Modelling Language (UML) is to be used to document an IMS information model [GWS, 05e]. The UML description is designed to facilitate the corresponding bindings, i.e., the UML description is not designed from the perspective of implementation as an Application Programming Interface (API). The UML description is also made available as an XML Metadata Interface (XMI) file; this enables tools to import the UML description and generate the corresponding code stubs. IMS has developed XML Stylesheet Language Transformations (XSLT) files to support automated UML conversion to WSDL, etc. The usage of these XSLTs and the corresponding tool is explained in the IMS Binding Auto-generation Toolkit Manual [GWS, 05e].

2.1.4 Recommendations from W3C Web Service Architecture

The Web Service Architecture (WSA) is a normative document that seeks to provide a context and model for understanding Web services and to facilitate placing Web services specifications and technologies within a larger Web Services framework and with other technologies outside of WSA. The WSA's goal is to promote interoperability through the common definition of a web service and its core concepts and relationships. The purpose of WSA is not to specify how Web services are implemented or to impose restrictions on combination or coordination of Web services. WSA provides discussion of the WSA's core concepts and relationships from various perspectives. Where appropriate, IMS will use the WSA in the IAF.

For more information on the Web Services Architecture, go to W3C Web Services Architecture (http://www.w3.org/TR/ws-arch/).

2.1.5 Recommendations from Web Services Interoperability Basic Profile

The IMS GWS Base Profile is based upon the WS-I Basic Profile v1.1 [WSI, 04a] and the WS-I Simple SOAP Binding Profile v1.0 [WSI, 04d]. The WS-I Basic Profile is shown in Table 2.1.

Table 2.1 WS-I basic profile v1.1.

Core Specification	Description
XML Schema V1.0	All data models will be defined in terms of XML Schema and will require the definition of the corresponding control documents (XSD).
HTTP V1.1 (RFC2616)	HTTP is the mandated protocol binding for the SOAP messages.
SOAP V1.1	SOAP is the mandated messaging protocol.

Core Specification	Description
WSDL V1.1	An instance of the service is defined using WSDL v1.1. WSDL is used to enable the description of services as sets of endpoints operating on messages.
UDDI V2.0	An instance of the service may be defined using a UDDI v2.0 binding.

It is recognized that SOAP v1.2 and WSDL v2.0 are available as later versions of the SOAP and WSDL specifications respectively. SOAP v1.2 and WSDL v2.0 will be considered for future revisions.. It should be noted, that from the perspective of adopted specifications, the combination of the WS-I Basic Profile v1.1 and Simple SOAP Binding Profile v1.0 is identical to the WS-I Basic Profile v1.0a [WSI, 03].

2.2 Core Technologies

2.2.1 XML

XML 1.0 (Third Edition) is the core data representation technology adopted for the binding of IMS specifications [XML, 04]. An IMS data-model oriented information model can be defined as a hierarchy. Hierarchical models are convenient for representing data consisting of many elements and sub-elements. XML is perfectly suited for representing hierarchical models.

2.2.2 XML Schemas

XML Schema Definition (XSD) is the primary XML binding control document format of IMS (at present these bindings are working to the May 2001 version of XML Schema) [XSD, 01]. The XSD defines elements, their content models, and attributes. It also defines the standard IMS vocabularies. The XSD defines the element types and attribute groups separately from the elements. The key recommendations for XML Schema with respect to the Base Profile are:

a) When used in the context of SOAP messages all data constructs should be defined as elements. Attributes should not be used. This follows the recommendation of the WS-I;

b) All data types should be defined as ‘Global’. The usage of ‘Local’ data types causes problems for some of the WSDL import tools, e.g., Axis;

c) All data types should have the character string “.Type” at the end of their name. This avoids naming conflicts between instances and their types.

2.2.3 SOAP

SOAP is a messaging protocol for XML documents which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment [SOAP, 03a], [SOAP, 03b]. It is a stateless, one-way message exchange mechanism, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying transport protocol and/or application-specific information. SOAP provides the framework by which application-specific information may be conveyed in an extensible manner. Also, SOAP provides a full description of the expected actions taken by a SOAP processor on receiving a SOAP message.

A SOAP message contains two SOAP-specific sub-elements within the overall Envelope: the ‘Header’ and ‘Body’. The contents of these elements are application defined and not a part of the SOAP specifications, although the latter does have something to say about how such elements must be handled. The ‘Header’ is optional. Headers have been designed in anticipation of various uses for SOAP, many of which will involve the participation of other SOAP processing nodes along a message’s path from a sender to an ultimate receiver, to allow SOAP processors to exchange information to provide value-added services. These form the mechanism by which SOAP messages may be extended in an application-specific manner. The ‘Body’ is the mandatory element within an Envelope and this is where the main information conveyed in a SOAP message must be carried. The immediate child elements of a ‘Header’ are called header blocks, and represent some logical grouping of data that can be targeted at SOAP nodes encountered in the path of a message from a sender to a receiver.

The SOAP envelope is the container for the messages to be exchanged between the systems. These messages need to be physically transmitted across the communications system using an appropriate transport mechanism. In many cases the Hypertext Transport Protocol (HTTP) is used as this transport mechanism. The key recommendations for XML Schema with respect to the Base Profiles are:

a) Only bindings based upon SOAPv1.1 across HTTPv1.1 are supported;

b) The SOAP message body contains all of the parameters defined in the corresponding operation;

c) The status information is to be placed in the SOAP message header.

2.2.4 WS-Addressing

Web Services Addressing (WS-Addressing) defines two interoperable constructs that convey information that is typically provided by transport protocols and messaging systems. These constructs normalize this underlying information into a uniform format that can be processed independently of transport or application. The two constructs are ‘endpoint references’ and ‘message information’ headers. Both of these constructs are designed to be extensible and re-usable so that other specifications can build on and leverage endpoint references and message information headers.

A Web service endpoint is a (referenceable) entity, processor, or resource where Web service messages can be targeted. Endpoint references convey the information needed to identify/reference a Web service endpoint, and may be used in several different ways: endpoint references are suitable for conveying the information needed to access a Web service endpoint, but are also used to provide addresses for individual messages sent to and from Web services. To deal with this last usage case this specification defines a family of message information headers that allows uniform addressing of messages independent of underlying transport. These message information headers convey end-to-end message characteristics including addressing for source and destination endpoints as well as message identity.

2.2.5 Attachments for SOAP Messages

Attachments for SOAP messages are available in several forms:

a) SOAP with Attachments (SOAPwA) – SOAP With Attachments (SOAPwA) extends SOAPv1.1 by providing a MIME binding to support multiple message payloads, while ignoring the convention by which Remote Procedure Call (RPC) arguments may be marshalled and unmarshalled in XML. SOAPwA is especially suited for use cases where the two communicating parties are NOT located within the same organization, and the exchange paradigm is therefore more one of asynchronous Document Exchange across the Internet, than synchronous Remote Procedure Call within a single business (or University) enterprise;

b) WS-Attachments – this specification defines an abstract model for SOAP attachments and based on this model defines a mechanism for encapsulating a SOAP message and zero or more attachments in a DIME message. SOAP attachments are described using the notion of a compound document structure consisting of a primary SOAP message and zero or more related documents known as attachments;

c) Message Transmission Optimization Mechanism (MTOM) – MTOM is one of the W3C message attachment approaches to enable SOAP messages to contain non-XML objects. MTOM is a development of SOAPwA and is proposed as a replacement for the original SOAPwA specification.

The equivalent attachments recommendations for IMS GWS are defined in the IMS GWS Attachments Profile document [GWS, 05b].

2.2.6 WSDL

A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** that are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitute a reusable **binding**. A port is defined by associating a network address with a reusable binding and a collection of ports defines a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types –** a container for data type definitions using some type system (such as XSD);

- **Message –**an abstract, typed definition of the data being communicated;
- **Operation –** an abstract description of an action supported by the service;
- **Port Type –**an abstract set of operations supported by one or more endpoints;
- **Binding –** a concrete protocol and data format specification for a particular port type;
- **Port –** a single endpoint defined as a combination of a binding and a network address;
- **Service –** a collection of related endpoints.

The key recommendations for WSDL with respect to the Base Profile are:

a) A separate set of WSDL files are defined for each type of communication mode being supported by the binding and there will be one form of the WSDL binding will be contained within a single physical file;

b) Alternatively there will be a split file binding in which the XSD information is defined in a separate file from that which contains the rest of the WSDL definition. The WSDL file imports the XSD definitions using the <xsd:import> construct;

c) The second alternative is to have the WSDL definition split into the service specific and abstract files plus the XSD, i.e., to create three separate but linked files. The service-specific file imports the abstract definitions using the <wsdl:import> construct and the abstract definitions file imports the XSD definitions using the <xsd:import> construct;

d) The final alternative is to have the two separate WSDL files plus several XSD files. The data schema and the message structure schema will be defined in separate files. The message structure XSD will contain all of the messages required for the different communication modes to be supported.

2.2.7 WS-Security

WS-Security defines a standard way to incorporate security information into a SOAP message using existing security standards for confidentiality, integrity, non-repudiation, authentication and authorization. WS-Security provides a method for representing security information in a SOAP message. WS-Security defines a way to pass security tokens, such as a simple username, SAML, X.509 certificates and Kerberos tickets, a mechanism using XML Signature to digitally sign all or part of a SOAP message, a mechanism using XML Encryption to encrypt part of a SOAP message and a method for attaching signature and encryption headers to a SOAP message. The equivalent security profile for IMS GWS is defined in the IMS GWS Security Profile [GWS, 05c].

2.2.8 Choreography

IMS plans to adopt the appropriate message and service choreography specifications, as opposed to creating its own. The underlying message choreography support in the GWS bindings are detailed in Section 5 of this document. Reliable messaging is only supported as a feature of the underlying communications infrastructure, e.g., through the usage of the Transmission Control Protocol (TCP). When stable, the following specifications will be adopted:

a) Web Services Reliable Messaging Framework (WS-Reliability) [WSR, 03] – the purpose of WS-Reliability is to address reliable messaging requirements, which become critical, for example, when using Web Services in B2B applications. SOAP [SOAP1.1] over HTTP [RFC2616] is not sufficient when an application-level messaging protocol must also address reliability and security. While security is getting traction in the development of Web Services standards, reliability is not. This specification is intended as an initial proposal for defining reliability in the context of current Web Services standards. The specification borrows from previous work in messaging an transport protocols, e.g., SOAP, and the ebXML Message Service. It proposes appropriate modifications to apply this work to Web Services;

b) Web Services Choreography Description Language Version 1.0 (W3C Working Draft 27 April 2004) – Web Services Choreography Description Language (WS-CDL) is an XML-based language that describes peer-to-peer collaborations of Web Services participants by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal. The Web Services specifications offer a communication bridge between the heterogeneous computational environments used to develop and host applications. The Web Services

Choreography specification is targeted for composing interoperable peer-to-peer collaborations between any type of Web Service participant regardless of the supporting platform or programming model used by the implementation of the hosting environment;

c) Business Process Execution Language for Web Services (BPEL4WS – BPEL4WS (or BPEL for short) is an XML-based standard for defining Web services can be combined to implement business processes [BPEL, 03]. It builds WSDL and XSD. BPEL provides an XML notation and semantics for specifying business process behavior based on Web Services. A BPEL4WS process is defined in terms of its interactions with partners. A partner may provide services to the process, require services from the process, or participate in a two-way interaction with the process. Thus BPEL orchestrates Web Services by specifying the order in which it is meaningful to call a collection of services, and assigns responsibilities for each of the services to partners. It can be used to specify both the public interfaces for the partners and the description of the executable process.

3. Base Profile Rules

3.1 IMS Base Profile

The IMS GWS Base Profile is defined in Table 3.1. The only difference between this profile and that of the WS-I Basic Profile/Simple SOAP Binding Profile is that the UDDI specification is not included in the profile.

Table 3.1 IMS GWS base profile.

Core Specification	Description
XML Schema V1.0	All data models in IMS specifications will be defined in terms of XML Schema and will require the definition of the corresponding control documents (XSD).
HTTPv1.1	HTTP is the mandated protocol binding for the SOAP messages.
SOAP V1.1	SOAP is the mandated messaging protocol.
WSDL V1.1	An instance of the service is defined using WSDL v1.1.

3.2 Base Profile Rules Derived from WS-I Basic Profile

Table 3.2 summarizes the set of rules used in the WS-I Basic Profile v1.1 and their equivalent adoption status for the IMS GWS Base Profile. Within Table 3.2 the following conventions are used:

- The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119;
- Normative statements in the Profile, i.e., those impacting conformance, as outlined in “Profile Conformance”, are presented in the following manner: Rnnnn*Statement text here*, where “nnnn” is replaced by the statement number. Each statement contains exactly one requirement level keyword, e.g., “MUST”, and one conformance target keyword, e.g., “MESSAGE”.

Table 3.2 Summary of the WS-I basic profile rules.

Rule	Description	IMS GWS Relevance
Profile Conformance		
R0002	A DESCRIPTION MAY contain conformance claims regarding instances, as specified in the conformance claim schema.	The mechanism for conformance claims will be addressed in GWS 2.0.
R0003	A DESCRIPTION’s conformance claims MUST be children of the <wsdl:documentation> element of each of the elements: <wsdl:port>, <wsdl:binding, wsdl:portType>, <wsdl:operation> (as a child element of <wsdl:portType> but not of <wsdl:binding) and <wsdl:message>.	The mechanism for conformance claims will be addressed in GWS 2.0.
Messaging		
R9980	An ENVELOPE MUST conform to the structure specified in SOAP 1.1 Section 4, “SOAP Envelope” (subject to amendment by the Profile).	Adopted.
R1015	A RECEIVER MUST generate a fault if they encounter a message whose document element has a local name of Envelope but a namespace name that is not <soap:Envelop>	Adopted.

Rule	Description	IMS GWS Relevance
R1014	The children of the <soap:Body> element in a MESSAGE MUST be namespace qualified.	Adopted.
R1008	A MESSAGE MUST NOT contain a Document Type Declaration.	Adopted.
R1009	A MESSAGE MUST NOT contain Processing Instructions.	Adopted.
R1033	An ENVELOPE SHOULD NOT contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".	Adopted.
R1034	A DESCRIPTION SHOULD NOT contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".	Adopted.
R1011	MESSAGE MUST NOT have any element children of <soap:Envelope> following the <soap:Body element.	Adopted.
R1005	A MESSAGE MUST NOT contain <soap:encodingStyle> attributes on any of the elements whose namespace name is “http://schemas.xmlsoap.org/soap/envelope/”.	Adopted.
R1006	A MESSAGE MUST NOT contain <soap:encodingStyle> attributes on any element that is a child of <soap:Body>.	Adopted.
R1007	A MESSAGE described in an rpc-literal binding MUST NOT contain <soap:encodingStyle> attribute on any elements are grandchildren of <soap:Body>.	Rejected. ‘rpc-literal’ bindings are not supported.
R1013	<soap:mustUnderstand> attribute MUST only use the lexical forms “0” and “1”.	Adopted.
R1017	A RECEIVER MUST NOT mandate the use of the ‘xsi:type’ attribute in messages except as required in order to indicate a derived type (see XML Schema Part 1: Structures, Section 2.6.1).	Adopted.
R1032	The soap:Envelope, soap:Header, and soap:Body elements in an ENVELOPE MUST NOT have attributes in the namespace “http://schemas.xmlsoap.org/soap/envelope/”.	Adopted.
R1025	A RECEIVER MUST handle messages in such a way that it appears that all checking of mandatory header blocks is performed before any actual processing.	Adopted.
R1027	A RECEIVER MUST generate a “soap:MustUnderstand” fault when a message contains a mandatory header block (i.e., one that has a <soap:mustUnderstand> attribute with the value “1”) targeted at the receiver (via <soap:actor>) that the receiver does not understand.	Adopted.
R1028	When a Fault is generated by a RECEIVER, further processing SHOULD NOT be performed on the SOAP message aside from that which is necessary to rollback, or compensate for, any effects of processing the message prior to the generation of the Fault.	Adopted.
R1029	Where the normal outcome of processing a SOAP message would have resulted in the transmission of a SOAP response, but rather a SOAP Fault is generated instead, a RECEIVER MUST transmit a SOAP Fault message in place of the response.	Adopted.
R1030	RECEIVER that generates a fault SHOULD notify the end user that a fault has been generated when practical, by whatever means is deemed appropriate to the circumstance.	Adopted.

Rule	Description	IMS GWS Relevance
R1107	A RECEIVER MUST interpret SOAP messages containing only a <soap:Fault> element as a Fault.	Adopted.
R1000	When a MESSAGE contains a <soap:Fault> element, that element MUST NOT have element children other than ‘faultcode’, ‘faultstring’, ‘faultactor’ and ‘detail’.	Adopted.
R1001	When a MESSAGE contains a <soap:Fault> element its element children MUST be unqualified.	Adopted.
R1002	A RECEIVER MUST accept fault messages that have any number of elements, including zero, appearing as children of the <detail> element. Such children can be qualified or unqualified.	Adopted.
R1003	A RECEIVER MUST accept fault messages that have any number of qualified or unqualified attributes, including zero, appearing on the detail element. The namespace of qualified attributes can be anything other than “http://schemas.xmlsoap.org/soap/envelope/”.	Adopted.
R1016	A RECEIVER MUST accept fault messages that carry an ‘xml:lang’ attribute on the <faultstring> element.	Adopted.
R1004	When a MESSAGE contains a <faultcode> element the content of that element SHOULD be one of the fault codes defined in SOAP 1.1 or a namespace qualified fault code.	Adopted.
R1031	When a MESSAGE contains a <faultcode> element the content of that element SHOULD NOT use of the SOAP 1.1 “dot” notation to refine the meaning of the Fault.	Adopted.
R1140	MESSAGE SHOULD be sent using HTTP/1.1.	Adopted.
R1141	MESSAGE MUST be sent using either HTTP/1.1 or HTTP/1.0.	Adopted. HTTP/1.0 support is implied by supporting HTTP/1.1.
R1132	A HTTP request MESSAGE MUST use the HTTP POST method.	Adopted.
R1108	A MESSAGE MUST NOT use the HTTP Extension Framework (RFC2774).	Adopted.
R1109	The value of the ‘SOAPAction’ HTTP header field in a HTTP request MESSAGE MUST be a quoted string.	Adopted.
R1119	A RECEIVER MAY respond with a Fault if the value of the ‘SOAPAction’ HTTP header field is not quoted.	Adopted.
R1127	A RECEIVER MUST NOT rely on the value of the SOAPAction HTTP header to correctly process the message.	Adopted.
R1124	An INSTANCE MUST use a 2xx HTTP status code for responses that indicate a successful outcome of a request.	Adopted.
R1111	An INSTANCE SHOULD use a “200 OK” HTTP status code for responses that contain a SOAP message that is not a SOAP fault.	Adopted.
R1112	An INSTANCE SHOULD use either a “200 OK” or “202 Accepted” R1130 HTTP status code for a response that does do not contain a SOAP message but indicates successful HTTP outcome of a request.	Adopted.
R1130	An INSTANCE MUST use HTTP status code “307 Temporary Redirect” when redirecting a request to a different endpoint.	Adopted.

Rule	Description	IMS GWS Relevance
R1131	A CONSUMER MAY automatically redirect a request when it encounters a “307 Temporary Redirect” HTTP status code in a response.	Adopted.
R1125	An INSTANCE MUST use a 4xx HTTP status code for responses that indicate a problem with the format of the request.	Adopted.
R1113	An INSTANCE SHOULD use a “400 Bad Request” HTTP status code, if the request message is a malformed HTTP request, or not well-formed XML.	Adopted.
R1114	An INSTANCE SHOULD use a “405 Method not Allowed” HTTP status code if the request method was not “POST”.	Adopted.
R1115	An INSTANCE SHOULD use a “415 Unsupported Media Type” HTTP status code if the Content-Type HTTP request header did not have a value consistent with the value specified for the corresponding binding of the input message.	Adopted.
R1126	An INSTANCE MUST use a “500 Internal Server Error” HTTP status code if the response message is a SOAP Fault.	Adopted.
R1120	An INSTANCE MAY use the HTTP state mechanism (“Cookies”).	Rejected. “Cookies” are not to be used.
R1122	An INSTANCE using Cookies SHOULD conform to RFC2965.	Rejected. “Cookies” are not to be used.
R1121	An INSTANCE SHOULD NOT require consumer support for Cookies in order to function correctly.	Rejected. “Cookies” are not to be used.
R1123	The value of the cookie MUST be considered to be opaque by the CONSUMER.	Rejected. “Cookies” are not to be used.
Service Description		
R0001	An INSTANCE MUST be described by a WSDL 1.1 service description, by a UDDI binding template, or both.	An instance will be described by a WSDL 1.1 service description.
R2028	A DESCRIPTION using the WSDL namespace MUST be valid according to the XML Schema found at “http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd”.	Adopted. The prefix used in the WS-I profile may be changed.
R2029	A DESCRIPTION using the WSDL SOAP binding namespace MUST be valid according to the XML Schema found at “http://schemas.xmlsoap.org/wsdl/soap/2003-02-11.xsd”	Adopted. The prefix used in the WS-I profile may be changed.
R2001	A DESCRIPTION MUST only use the WSDL “import” statement to import another WSDL description.	Adopted. This method is used to import the abstract definitions file into the service specific file.
R2803	In a DESCRIPTION, the namespace attribute of the wsdl:import MUST NOT be a relative URI.	Adopted.
R2002	To import XML Schema Definitions, a DESCRIPTION MUST use the XML Schema “import” statement.	Adopted. This method is used to import the XSD descriptions.
R2003	A DESCRIPTION MUST use the XML Schema “import” statement only within the <xs:schema> element of the types section.	Adopted.
R2004	A DESCRIPTION MUST NOT use the XML Schema “import” statement to import a Schema from any document whose root element is not “schema” from the namespace “http://www.w3.org/2001/XMLSchema”.	Adopted.

Rule	Description	IMS GWS Relevance
R2009	An XML Schema directly or indirectly imported by a DESCRIPTION MAY include the Unicode Byte Order Mark (BOM).	Adopted.
R2010	An XML Schema directly or indirectly imported by a DESCRIPTION MUST use either UTF-8 or UTF-16 encoding.	Adopted.
R2011	An XML Schema directly or indirectly imported by a DESCRIPTION MUST use version 1.0 of the Extensible Markup Language W3C Recommendation.	Adopted.
R2007	A DESCRIPTION MUST specify a non-empty location attribute on the <wsdl:import> element.	Adopted. The location will be based upon the URL root of: http://www.imsglobal.org/services/..
R2008	In a DESCRIPTION the value of the location attribute of a <wsdl:import> element SHOULD be treated as a hint.	Adopted.
R2022	When they appear in a DESCRIPTION, <wsdl:import> elements MUST precede all other elements from the WSDL namespace except <wsdl:documentation>.	Adopted.
R2023	When they appear in a DESCRIPTION, <wsdl:types> elements MUST precede all other elements from the WSDL namespace except <wsdl:documentation> and <wsdl:import>.	Adopted.
R4004	A DESCRIPTION MUST use version 1.0 of the Extensible Markup Language W3C Recommendation.	Adopted.
R4005	A DESCRIPTION SHOULD NOT contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".	Adopted.
R4002	A DESCRIPTION MAY include the Unicode Byte Order Mark (BOM).	Adopted.
R4003	DESCRIPTION MUST use either UTF-8 or UTF-16 encoding.	Adopted.
R2005	The ‘targetNamespace’ attribute on the <wsdl:definitions> element of a description that is being imported MUST have the same value as the namespace attribute on the <wsdl:import> element in the importing DESCRIPTION.	Adopted.
R2030	In a DESCRIPTION the <wsdl:documentation> element MAY be present as the first child element of <wsdl:import>, <wsdl:part> and <wsdl:definitions> in addition to the elements cited in the WSDL1.1 specification.	Adopted.
R2025	A DESCRIPTION containing WSDL extensions MUST NOT use them to contradict other requirements of the Profile.	Adopted.
R2026	A DESCRIPTION SHOULD NOT include extension elements with a ‘wsdl:required’ attribute value of “true” on any WSDL construct (<wsdl:binding>, <wsdl:portType>, <wsdl:message>, <wsdl:types> or <wsdl:import>) that claims conformance to the Profile.	Adopted.
R2027	If during the processing of an element in the WSDL namespace in a description, a consumer encounters a WSDL extension element amongst its element children, that has a <wsdl:required> attribute with a boolean value of “true” that the consumer does not understand or cannot process, the CONSUMER MUST fail processing of that element in the WSDL namespace.	Adopted.

Rule	Description	IMS GWS Relevance
R2101	A DESCRIPTION MUST NOT use QName references to elements in namespaces that have been neither imported, nor defined in the referring WSDL document.	Adopted.
R2102	A QName reference to a Schema component in a DESCRIPTION MUST use the namespace defined in the 'targetNamespace' attribute on the <xs:schema> element, or to a namespace defined in the namespace attribute on an <xs:import> element within the <xs:schema> element.	Adopted.
R2105	All <xs:schema> elements contained in a <wsdl:types> element of a DESCRIPTION MUST have a 'targetNamespace' attribute with a valid and non-null value, UNLESS the <xs:schema> element has <xs:import> and/or <xs:annotation> as its only child element(s).	Adopted.
R2110	In a DESCRIPTION, array declarations MUST NOT extend or restrict the 'soapenc:Array' type.	Adopted.
R2111	In a DESCRIPTION, array declarations MUST NOT use 'wsdl:arrayType' attribute in the type declaration.	Adopted.
R2112	In a DESCRIPTION, array declaration wrapper elements SHOULD NOT be named using the convention ArrayOfXXX.	Adopted.
R2113	A MESSAGE containing serialized arrays MUST NOT include the 'soapenc:arrayType' attribute.	Adopted.
R2114	The target namespace for WSDL definitions and the target namespace for schema definitions in a DESCRIPTION MAY be the same.	Adopted.
R2201	A document-literal binding in a DESCRIPTION MUST, in each of its <soapbind:body> element(s), have at most one part listed in the parts attribute, if the parts attribute is specified.	Adopted.
R2209	A <wsdl:binding> in a DESCRIPTION SHOULD bind every <wsdl:part> of a <wsdl:message> in the <wsdl:portType> to which it refers to one of <soapbind:body>, <soapbind:header>, <soapbind:fault> or <soapbind:headerfault>.	Adopted.
R2210	If a document-literal binding in a DESCRIPTION does not specify the parts attribute on a <soapbind:body> element, the corresponding abstract <wsdl:message> MUST define zero or one <wsdl:parts>.	Adopted. Every message will have one <wsdl:part> for the SOAP message body.
R2202	A <wsdl:binding> in a DESCRIPTION MAY contain <soapbind:body> element(s) that specify that zero parts form the <soap:Body>.	Every message will have one part for the <soap:Body>.
R2203	An rpc-literal binding in a DESCRIPTION MUST refer, in its <soapbind:body> element(s), only to <wsdl:part> element(s) that have been defined using the type attribute.	Rejected. 'rpc-literal' bindings are not supported.
R2211	A MESSAGE described with an rpc-literal binding MUST NOT have the 'xsi:nil' attribute with a value of "1" or "true" on the part accessors.	Rejected. 'rpc-literal' bindings are not supported.
R2207	A <wsdl:message> in a DESCRIPTION MAY contain <wsdl:parts> that use the elements attribute provided those <wsdl:parts> are not referred to by a <soapbind:body> in an rpc-literal binding.	Rejected. 'rpc-literal' bindings are not supported.

Rule	Description	IMS GWS Relevance
R2204	A document-literal binding in a DESCRIPTION MUST refer, in each of its <soapbind:body> element(s), only to <wsdl:part> element(s) that have been defined using the element attribute.	Adopted.
R2208	A binding in a DESCRIPTION MAY contain <soapbind:header> element(s) that refer to <wsdl:parts> in the same <wsdl:message> that are referred to by its <soapbind:body> element(s).	Adopted.
R2212	An ENVELOPE MUST contain exactly one part accessor element for each of the <wsdl:part> elements bound to the envelope's corresponding <soapbind:body> element.	Adopted.
R2213	In a doc-literal description where the value of the parts attribute of soapbind:body is an empty string, the corresponding ENVELOPE MUST have no element content in the <soap:Body> element.	The value of the parts attribute of 'soapbind:body' will never be an empty string,
R2214	In a rpc-literal description where the value of the parts attribute of 'soapbind:body' is an empty string, the corresponding ENVELOPE MUST have no part accessor elements.	Rejected. 'rpc-literal' bindings are not supported.
R2205	A <wsdl:binding> in a DESCRIPTION MUST refer, in each of its <soapbind:header>, <soapbind:headerfault> and <soapbind:fault> elements, only to <wsdl:part> element(s) that have been defined using the element attribute.	Adopted.
R2206	A <wsdl:message> in a DESCRIPTION containing a <wsdl:part> that uses the element attribute MUST refer, in that attribute, to a global element declaration.	Adopted.
R2301	The order of the elements in the <soap:body> of a MESSAGE MUST be the same as that of the <wsdl:parts> in the <wsdl:message> that describes it.	Adopted.
R2302	A DESCRIPTION MAY use the 'parameterOrder' attribute of an <wsdl:operation> element to indicate the return value and method signatures as a hint to code generators.	Adopted.
R2303	A DESCRIPTION MUST NOT use Solicit-Response and Notification type operations in a <wsdl:portType> definition.	Adopted.
R2304	A <wsdl:portType> in a DESCRIPTION MUST have operations with distinct values for their name attributes.	Adopted.
R2305	A <wsdl:portType> in a DESCRIPTION MUST be constructed so that the 'parameterOrder' attribute, if present, omits at most one <wsdl:part> from the output message.	Adopted.
R2306	A <wsdl:message> in a DESCRIPTION MUST NOT specify both type and element attributes on the same <wsdl:part>.	Adopted.
R2401	A <wsdl:binding> element in a DESCRIPTION MUST use WSDL SOAP Binding as defined in WSDL 1.1 Section 3.	Adopted.
R2701	The <wsdl:binding> element in a DESCRIPTION MUST be constructed so that its <soapbind:binding> child element specifies the transport attribute.	Adopted.
R2702	A <wsdl:binding> element in a DESCRIPTION MUST specify the HTTP transport protocol with SOAP binding. Specifically, the transport attribute of its <soapbind:binding> child MUST have the value "http://schemas.xmlsoap.org/soap/http".	Adopted.

Rule	Description	IMS GWS Relevance
R2705	A <wsdl:binding> in a DESCRIPTION MUST use either a rpc-literal binding or a document-literal binding.	Adopted. The binding will always be based upon ‘document-literal’.
R2706	A <wsdl:binding> in a DESCRIPTION MUST use the value of “literal” for the use attribute in all <soapbind:body>, <soapbind:fault>, <soapbind:header> and <soapbind:headerfault> elements.	Adopted.
R2709	A <wsdl:portType> in a DESCRIPTION MAY have zero or more <wsdl:binding> that refer to it, defined in the same or other WSDL documents.	A <wsdl:portType> definition will always result in a <wsdl:binding> definition.
R2710	The operations in a <wsdl:binding> in a DESCRIPTION MUST result in wire signatures that are different from one another.	Adopted.
R2711	A DESCRIPTION SHOULD NOT have more than one <wsdl:port> with the same value for the location attribute of the <soapbind:address> element.	Adopted.
R2712	A document-literal binding MUST be represented on the wire as a MESSAGE with a <soap:Body> whose child element is an instance of the global element declaration referenced by the corresponding <wsdl:message> part.	Adopted.
R2714	For one-way operations, an INSTANCE MUST NOT return a HTTP response that contains a SOAP envelope. Specifically, the HTTP response entity-body must be empty.	Adopted. At the present time all services use Request/Response.
R2750	A CONSUMER MUST ignore a SOAP response carried in a response from a one-way operation.	Adopted. At the present time all services use Request/Response.
R2727	For one-way operations, a CONSUMER MUST NOT interpret a successful HTTP response status code (i.e., 2xx) to mean the message is valid or that the receiver would process it.	Adopted. At the present time all services use Request/Response.
R2716	A document-literal binding in a DESCRIPTION MUST NOT have the namespace attribute specified on contained <soapbind:body>, <soapbind:header>, <soapbind:headerfault> and <soapbind:fault> elements.	Adopted.
R2717	An rpc-literal binding in a DESCRIPTION MUST have the namespace attribute specified, the value of which MUST be an absolute URI, on contained <soapbind:body> elements.	Rejected. ‘rpc-literal’ bindings are not supported.
R2726	An rpc-literal binding in a DESCRIPTION MUST NOT have the namespace attribute specified on contained <soapbind:header>, <soapbind:headerfault> and <soapbind:fault> elements.	Rejected. ‘rpc-literal’ bindings are not supported.
R2718	A <wsdl:binding> in a DESCRIPTION MUST have the same set of <wsdl:operations> as the <wsdl:portType> to which it refers.	Adopted.
R2719	A <wsdl:binding> in a DESCRIPTION MAY contain no <soapbind:headerfault> elements if there are no known header faults.	Adopted. Currently IMS does not define header faults.
R2740	A <wsdl:binding> in a DESCRIPTION SHOULD contain a <soapbind:fault> describing each known fault.	Known business transaction faults are reported in the IMS StatusInfo object carried in the SOAP header.
R2741	A <wsdl:binding> in a DESCRIPTION SHOULD contain a soapbind:headerfault describing each known header fault.	Adopted. Currently IMS does not define header faults.

Rule	Description	IMS GWS Relevance
R2742	A MESSAGE MAY contain a fault detail entry in a SOAP fault that is not described by a <wsdl:fault> element in the corresponding WSDL description.	Known business transaction faults are reported in the IMS StatusInfo object carried in the SOAP header.
R2743	A MESSAGE MAY contain the details of a header processing related fault in a SOAP header block that is not described by a <wsdl:headerfault> element in the corresponding WSDL description.	Adopted. Currently IMS does not define header faults.
R2720	A <wsdl:binding> in a DESCRIPTION MUST use the attribute named part with a schema type of “NMTOKEN” on all contained <soapbind:header> and <soapbind:headerfault> elements.	Adopted.
R2749	A <wsdl:binding> in a DESCRIPTION MUST NOT use the attribute named parts on contained <soapbind:header> and <soapbind:headerfault> elements.	Adopted.
R2721	A <wsdl:binding> in a DESCRIPTION MUST have the name attribute specified on all contained <soapbind:fault> elements.	Known business transaction faults are reported in the IMS StatusInfo object carried in the SOAP header.
R2754	In a DESCRIPTION, the value of the name attribute on a <soapbind:fault> element MUST match the value of the name attribute on its parent <wsdl:fault> element.	Known business transaction faults are reported in the IMS StatusInfo object carried in the SOAP header.
R2722	A <wsdl:binding> in a DESCRIPTION MAY specify the use attribute on contained <soapbind:fault> elements.	Known business transaction faults are reported in the IMS StatusInfo object carried in the SOAP header.
R2723	If in a <wsdl:binding> in a DESCRIPTION the use attribute on a contained <soapbind:fault> element is present, its value MUST be “literal”.	Known business transaction faults are reported in the IMS StatusInfo object carried in the SOAP header.
R2707	A <wsdl:binding> in a DESCRIPTION that contains one or more <soapbind:body>, <soapbind:fault>, <soapbind:header> or <soapbind:headerfault> elements that do not specify the use attribute MUST be interpreted as though the value “literal” had been specified in each case.	Adopted.
R2724	If an INSTANCE receives a message that is inconsistent with its WSDL description, it SHOULD generate a <soap:Fault> with a faultcode of “Client”, unless a “MustUnderstand” or “VersionMismatch” fault is generated.	Adopted.
R2725	If an INSTANCE receives a message that is inconsistent with its WSDL description, it MUST check for “VersionMismatch”, “MustUnderstand” and “Client” fault conditions in that order.	Adopted.
R2729	A MESSAGE described with an rpc-literal binding that is a response message MUST have a wrapper element whose name is the corresponding <wsdl:operation> name suffixed with the string “Response”.	Rejected. ‘rpc-literal’ bindings are not supported.
R2735	A MESSAGE described with an rpc-literal binding MUST place the part accessor elements for parameters and return value in no namespace.	Rejected. ‘rpc-literal’ bindings are not supported.
R2755	The part accessor elements in a MESSAGE described with an rpc-literal binding MUST have a local name of the same value as the name attribute of the corresponding wsdl:part element.	Rejected. ‘rpc-literal’ bindings are not supported.

Rule	Description	IMS GWS Relevance
R2737	A MESSAGE described with an rpc-literal binding MUST namespace qualify the children of part accessor elements for the parameters and the return value with the ‘targetNamespace’ in which their types are defined.	Rejected. ‘rpc-literal’ bindings are not supported.
R2738	A MESSAGE MUST include all <soapbind:headers> specified on a <wsdl:input> or <wsdl:output> of a <wsdl:operation> of a <wsdl:binding> that describes it.	Adopted.
R2739	A MESSAGE MAY contain SOAP header blocks that are not described in the <wsdl:binding> that describes it.	Adopted.
R2753	A MESSAGE containing SOAP header blocks that are not described in the appropriate <wsdl:binding> MAY have the ‘mustUnderstand’ attribute on such SOAP header blocks set to ‘1’.	Adopted.
R2751	The order of <soapbind:header> elements in <soapbind:binding> sections of a DESCRIPTION MUST be considered independent of the order of SOAP header blocks in the message.	Adopted.
R2752	A MESSAGE MAY contain more than one instance of each SOAP header block for each <soapbind:header> element in the appropriate child of <soapbind:binding> in the corresponding description.	Adopted.
R2744	A HTTP request MESSAGE MUST contain a ‘SOAPAction’ HTTP header field with a quoted value equal to the value of the ‘soapAction’ attribute of <soapbind:operation>, if present in the corresponding WSDL description.	Adopted.
R2745	A HTTP request MESSAGE MUST contain a ‘SOAPAction’ HTTP header field with a quoted empty string value, if in the corresponding WSDL description, the ‘soapAction of <soapbind:operation> is either not present, or present with an empty string as its value.	Adopted.
R2747	A CONSUMER MUST understand and process all WSDL 1.1 SOAP Binding extension elements, irrespective of the presence or absence of the <wsdl:required> attribute on an extension element; and irrespective of the value of the <wsdl:required> attribute, when present.	Adopted.
R2748	A CONSUMER MUST NOT interpret the presence of the ‘wsdl:required’ attribute on a ‘soapbind’ extension element with a value of “false” to mean the extension element is optional in the messages generated from the WSDL description.	Adopted.
R2800	A DESCRIPTION MAY use any construct from XML Schema 1.0.	Adopted. The element form for abstract types must be used. Global definitions for elements must always be used.
R2801	A DESCRIPTION MUST use XML Schema 1.0 Recommendation as the basis of user defined data-types and structures.	Adopted.
Service Publication & Discovery		
R3100	REGDATA of type ‘uddi:bindingTemplate’ representing a conformant INSTANCE MUST contain the <uddi:accessPoint> element.	UDDI is not a part of the GWS base profiles.

Rule	Description	IMS GWS Relevance
R3002	REGDATA of type ‘uddi:tModel’ representing a conformant Web service type MUST use WSDL as the description language.	UDDI is not a part of the GWS base profiles.
R3003	REGDATA of type ‘uddi:tModel’ representing a conformant Web service type MUST be categorized using the uddi:types taxonomy and a categorization of “wsdlSpec”.	UDDI is not a part of the GWS base profiles.
R3010	REGDATA of type uddi:tModel representing a conformant Web service type MUST follow V1.08 of the UDDI Best Practice for Using WSDL in a UDDI Registry.	UDDI is not a part of the GWS base profiles.
R3011	The ‘wsdl:binding’ that is referenced by REGDATA of type ‘uddi:tModel’ MUST itself conform to the Profile.	UDDI is not a part of the GWS base profiles.
Security[a]		
R5000	An INSTANCE MAY require the use of HTTPS.	Adopted for the secure forms of the base profile.
R5001	If an INSTANCE requires the use of HTTPS, the location attribute of the <soapbind:address> element in its <wsdl:port> description MUST be a URI whose scheme is “https”; otherwise it MUST be a URI whose scheme is “http”.	Adopted for the secure forms of the base profile.
R5010	An INSTANCE MAY require the use of HTTPS with mutual authentication.	Adopted for the secure forms of the base profile.

IMS GWS base profile plus security

a. The IMS GWS Security Profile [GWS, 05c] should be used for a more complete statement on security aspects of the Base Profile.

Table 3.3 summarizes the set of extension points available in the WS-I Basic Profile v1.1 and thus within the IMS GWS. Extensibility points in underlying specifications are presented as:

- Ennnn*Extensibility Point Name – Description*, where “nnnn” is replaced by a number that is unique among the extensibility points in the Profile. As with requirement statements, extensibility statements can be considered namespace-qualified.

Table 3.3 Summary of the WS-I basic profile extension points.

Rule	Description	IMS GWS Relevance
Simple Object Access Protocol (SOAP) 1.1		
E0001	**Header blocks** – Header blocks are the fundamental extensibility mechanism in SOAP.	Adopted.
E0002	**Processing order** – the order of processing of a SOAP envelope’s components (e.g., headers) is unspecified, and therefore may need to be negotiated out-of-band.	Adopted.
E0003	**Use of intermediaries –** SOAP Intermediaries is an underspecified mechanism in SOAP 1.1, and their use may require out-of-band negotiation. Their use may also necessitate careful consideration of where Profile conformance is measured.	Adopted.
E0004	**soap:actor values** – values of the ‘soap:actor’ attribute, other than the special uri “http://schemas.xmlsoap.org/soap/actor/next” , represent a private agreement between parties of the web service.	Adopted.
E0005	**Fault details** – the contents of a Fault’s <detail> element are not prescribed by SOAP 1.1.	Adopted.

Rule	Description	IMS GWS Relevance
E0006	**Envelope serialization** – the Profile does not constrain some aspects of how the envelope is serialized into the message.	Adopted.
Hypertext Transfer Protocol – HTTP/1.1		
E0007	**HTTP Authentication** – HTTP authentication allows for extension schemes, arbitrary digest hash algorithms and parameters.	Adopted.
E0008	**Unspecified Header Fields** – HTTP allows arbitrary headers to occur in messages.	Adopted.
E0009	**Expect-extensions** – the Expect/Continue mechanism in HTTP allows for expect-extensions	Adopted.
E0010	**Content-Encoding** – the set of content-codings allowed by HTTP is open-ended and any besides ‘gzip’, ‘compress’, or ‘deflate’ are an extensibility point.	Adopted.
E0011	**Transfer-Encoding** – the set of transfer-encodings allowed by HTTP is open-ended.	Adopted.
E0012	**Upgrade** – HTTP allows a connection to change to an arbitrary protocol using the Upgrade header.	Adopted.
E0024	**Namespace Attributes** – namespace attributes on soap:Envelope and soap:Header elements	Adopted.
E0025	**Attributes on soap:Body elements** – neither namespaced nor local attributes are constrained by SOAP 1.1.	Adopted.
XML Schema Structures		
E0017	**Schema annotations** – XML Schema allows for annotations, which may be used to convey additional information about data structures.	Adopted.
WSDL Version 1.1		
E0013	**WSDL extensions** – WSDL allows extension elements in certain places; use of such extensions requires out-of-band negotiation.	Adopted.
E0014	**Validation mode** – whether the parser used to read WSDL and XML Schema documents performs DTD validation or not.	Only XSD validation is supported.
E0015	**Fetching of external resources** – whether the parser used to read WSDL and XML Schema documents fetches external entities and DTDs.	Adopted.
E0016	**Relative URIs** – WSDL does not adequately specify the use of relative URIs for the following: soapbind:body/@namespace, soapbind:address/@location, wsdl:import/@location, xsd:schema/@targetNamespace and xsd:import/@schemaLocation. Their use may require further coordination; see XML Base for more information.	Adopted.
RFC2246: The TLS Protocol Version 1.0		
E0019	**TLS Cyphersuite** – TLS allows for the use of arbitrary encryption algorithms.	Adopted.
E0020	**TLS Extensions** – TLS allows for extensions during the handshake phase.	Adopted.
The SSL Protocol Version 3.0		
E0021	**SSL Cyphersuite** – SSL allows for the use of arbitrary encryption algorithms.	Adopted.
RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile		
E0022	**Certificate Authority** – the choice of the Certificate Authority is a private agreement between parties.	Adopted.
E0023	**Certificate Extensions** – X509 allows for arbitrary certificate extensions.	Adopted.

3.3 Base Profile Rules Derived from WS-I Simple SOAP Binding Profile

Table 3.4 summarizes the set of rules used in the WS-I Simple SOAP Profile v1.0 and their equivalent adoption status for the IMS GWS Base Profile. Within Table 3.4 the following conventions are used:

- The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119;
- Normative statements in the Profile, i.e., those impacting conformance, as outlined in "Profile Conformance", are presented in the following manner: Rnnnn*Statement text here*, where "nnnn" is replaced by the statement number. Each statement contains exactly one requirement level keyword, e.g., "MUST", and one conformance target keyword, e.g., "MESSAGE".

Table 3.4 Summary of the WS-I simple SOAP profile rules.

Rule	Description	IMS GWS Relevance
Message Serialization		
R9700	A MESSAGE MUST serialize the envelope as the exclusive payload of the HTTP entity-body.	Adopted.
R9701	A MESSAGE MUST serialize the envelope as XML 1.0.	Adopted.
R9702	A MESSAGE MUST have a "Content-Type" HTTP header field.	Adopted.
R9703	A MESSAGE's "Content-Type" HTTP header field MUST have a field-value whose media type is "text/xml".	Adopted.
R9704	An ENVELOPE SHOULD NOT contain the namespace declaration xmlns:xml="http://www.w3.org/XML/1998/namespace".	Adopted.
R4001	A RECEIVER MUST accept envelopes that include the Unicode Byte Order Mark (BOM).	Adopted.
R1010	RECEIVER MUST accept messages with envelopes that contain an XML Declaration.	Adopted.
R1012	A MESSAGE MUST serialize the envelope using either UTF-8 or UTF-16 character encoding.	Adopted.
R1018	A MESSAGE's "Content-Type" HTTP header field-value MUST indicate the correct character encoding, using the "charset" parameter.	Adopted.
R1019	A RECEIVER MUST ignore the encoding pseudo-attribute of the envelope's XML declaration in a message.	Adopted.
Bindings		
R9802	A <wsdl:binding> element in a DESCRIPTION MUST only use the WSDL SOAP Binding as defined in WSDL 1.1 Section 3.	Adopted.
R9800	In a DESCRIPTION WSDL binding extension elements and attributes which cause messages on the wire to be non-conformant to the Profile MUST NOT be used.	Adopted.
R9801	In a DESCRIPTION the WSDL MIME and HTTP GET/POST and DIME binding extensions MUST NOT appear in the SOAP binding.	Adopted.
R2209	A <wsdl:binding> in a DESCRIPTION SHOULD bind every <wsdl:part> of a <wsdl:message> in the <wsdl:portType> to which it refers to one of <soapbind:body>, <soapbind:header>, <soapbind:fault> or <soapbind:headerfault>.	Adopted.

3.4 Summary of Differences between IMS and WS-I Profiles

The key points for the IMS Base profile are that:

a) SOAP fault messages will not be defined in the binding WSDL. All status information is carried in the StatusInfo/StatusInfoSet object. If SOAP fault codes are received then these must be mapped to the StatusInfo/StatusInfoSet objects as defined by IMS [GWS, 05];

- The usage of WS-I Conformance Claims is not required. If these are used then they have implementation-specific relevance;
- UDDI is not a part of the profile;
- ‘Document-literal” encoding is used, i.e., the is no “rpc-lteral” encoding.

4. Relationship to the Abstract Framework

4.1 IAF Overview

The IMS/GLC Abstract Framework (IAF) can be represented as a layered model [AbsWhite, 03], [AbsASC, 03], [AbsGloss, 03], as shown in Figure 4.1, consisting of four layers:

- Application layer – this is a tool, system, agent, etc. that presents the appropriate application services to the user, i.e., an application manages the user interface. The application may use one or more application services but whenever possible the system composition should be hidden from the user;
- Application Services layer – a set of services that provide the required eLearning functionality to the applications. An application service may make use of one or more common services. Distributed application services communicate using via the Infrastructure Layer;
- Common Services layer – a set of services that are available to the application services. Common services may use other common services. Therefore, a common service is available to any other service;
- Infrastructure layer – this provides the end-to-end transaction and communications services for the application and common services.

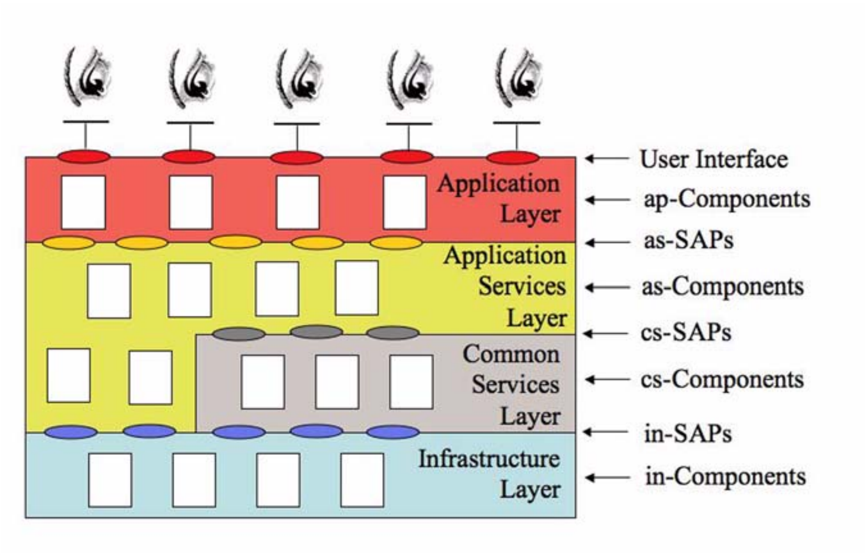

Figure 4.1 The abstract framework layered model.

Access to a service is through the appropriate Service Access Point (SAP). Each service has a single SAP. A Component may support one or more SAPs (in an object oriented representation, a SAP could be supported by one or more operators where the class is itself the definition of the service).

In a distributed implementation of this layered framework, as typified in a webs services environment, the interaction between the services would as shown in Figure 4.2. In this interaction framework there are three categories of interface that must be supported by the Infrastructure layer namely:

- The Application Services interface (A1) – this interface is used to provide interoperability between common application services, e.g., Enterprise-to-Enterprise systems. One form of the interface is based upon XML-messaging;
- The Common Services interface (A2) – this interface is used to provide interoperability with the set of common services that are made available to the application specific services, e.g., authentication and authorization for an Enterprise system;

- The run-time interface (B) – this interface is used to interconnect the client's run-time application with the remote service provider.

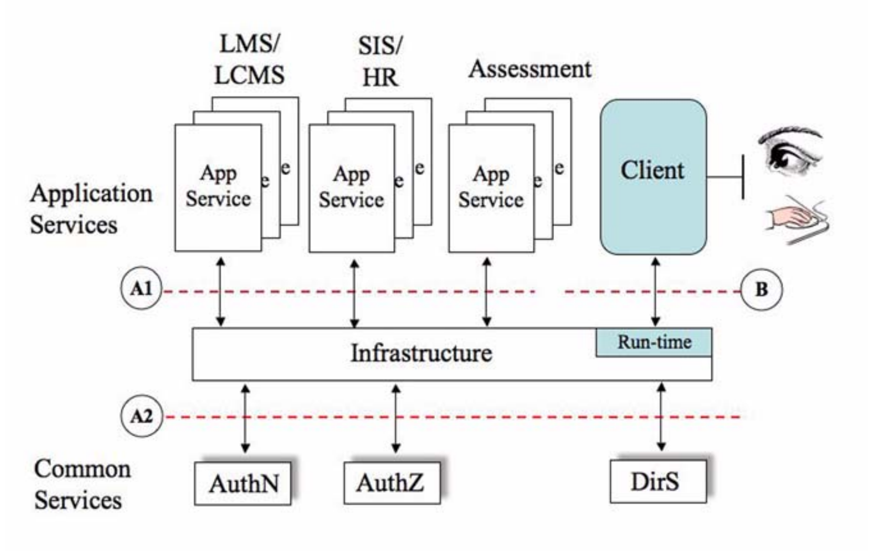

Figure 4.2 Abstract framework service interaction through the layers.

Figure 4.2 shows that there are two types of interaction behavior that need to be specified to ensure interoperability, namely:

- Message passing – where information is exchanged between systems using some form or another of message exchange. The content and sequence of the messages define the expected behavior. The communicating systems are expected to process and generate the messages as a response to known and understood events. The systems must be capable of handling unknown error conditions;
- Run-time – where the end systems have to reliably operate on information using some predefined algorithm. The data will determine the outcomes of the algorithms but the behavior is well defined for all possible outcomes.

The IMS/GLC specifications are focused on data exchange interoperability. To this end they define a data model of the information to be exchanged and a behavioral model that encapsulates the data model and constrains the way in which the data can be manipulated. An IMS/GLC information model is the manifestation of this behavioral and data description and an information model will consist of one or more IMS/GLC Components (these will realize either all, or part of, an Application Service). These components can then be realized in a variety of ways; the defined IMS/GLC method is as an XML-based binding. As such, this binding describes the way in which the data is exchanged in the form of XML messages/documents, however the actual transfer of these structures requires, at the very least, an appropriate communications system.

The exchange of the data between the XML components within the abstract framework is defined through the Infrastructure Layer. A schematic representation of the system components in this infrastructure description is shown in Figure 4.3. This representation assumes that the system is loosely coupled, e.g., as per web services.

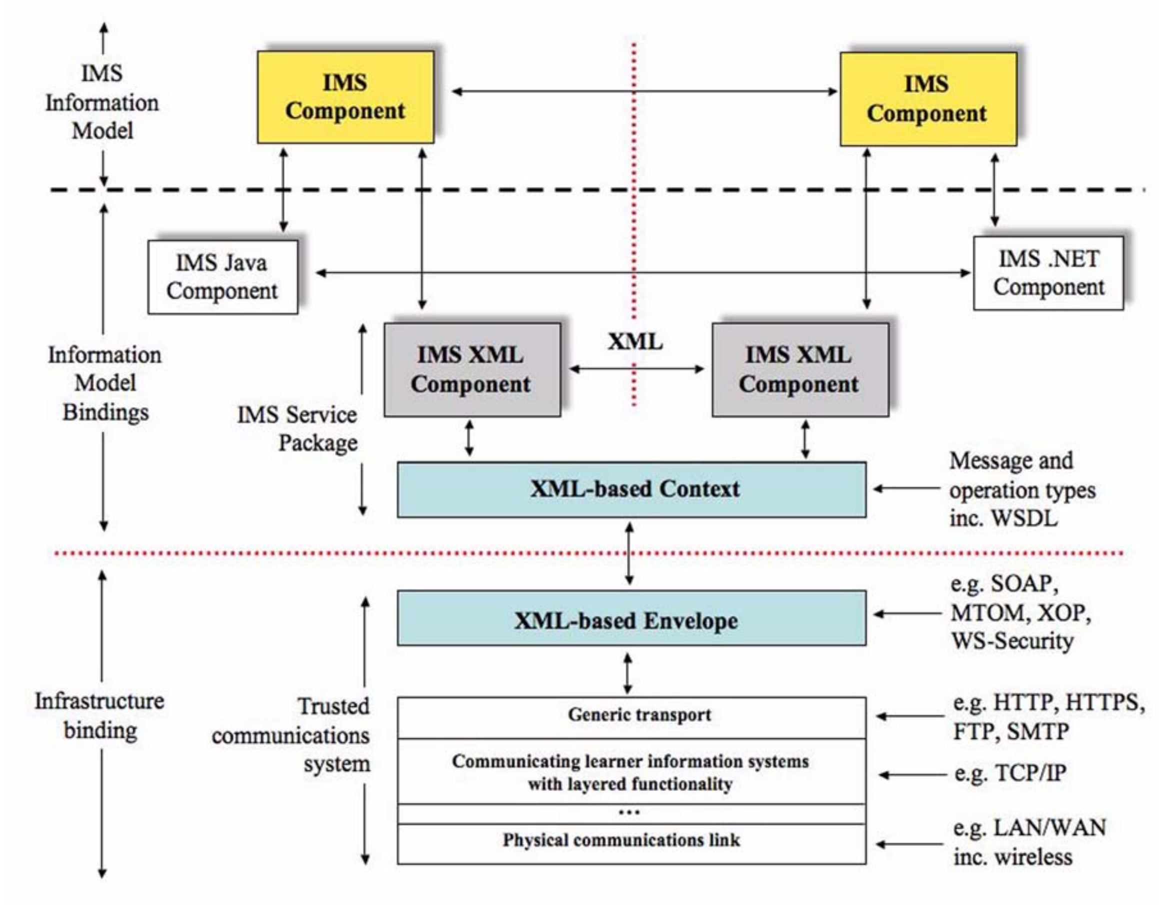
IMS Information Model
IMS Component
IMS Component
IMS Java Component
IMS .NET Component
Information Model Bindings
IMS Service Package
IMS XML Component
XML
IMS XML Component
XML-based Context
Message and operation types inc. WSDL
XML-based Envelope
e.g. SOAP, MTOM, XOP, WS-Security
Infrastructure binding
Trusted communications system
Generic transport
e.g. HTTP, HTTPS, FTP, SMTP
Communicating learner information systems with layered functionality
e.g. TCP/IP
Physical communications link
e.g. LAN/WAN inc. wireless

Figure 4.3 Infrastructure layer in the abstract framework.

In Figure 4.3 the key parts of this infrastructure are:

- IMS/GLC XML Components – the application and common services components that are combined to create the e-learning system required. It is assumed that these components exchange information in the form of XML documents;
- XML-based Context – the XML documents are transformed to XML messages which are then mapped onto a common XML messaging infrastructure that is designed to support the required end-to-end services, e.g., reliable data transfer, datagram, publish and subscribe, etc. The preferred context mapping language is the Web Services Description Language (WSDL);
- XML-based Envelope – the common XML messaging system can be supported using several types of XML envelope encapsulation, e.g., SOAP/SOAP message attachments (used to support file attachments such as images, etc.), eb-XML, etc. WSDL supports SOAP, ‘HTTP-Get’ and ‘HTTP-Post’ transport mechanisms;
- Generic transport – the envelope is then transported across the network using an appropriate end-to-end file transfer protocol, e.g., Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), etc.;
- Communications network – this is the actual data network that is used to physically transport the data from one system to another. This will almost certainly be based upon the ubiquitous Internet Transmission Control Protocol/Internet Protocol (TCP/IP) combination providing seamless interworking between wire-line and wireless networks.

4.2 Application Services Layer

4.2.1 Conventions for Definitions of End Points

The end-points for the communicating services are defined as the 'Initiator' and 'Respondent'[1]. The 'Initiator' is the system that starts the interaction, resulting in the transmission of a message, and the 'respondent' is the system that must process the message and may or may not return some information, i.e., it responds to the initiating interaction. The applications services are represented by Service Access Point, i.e., the abstract API. Successful communication requires the 'Respondent' to be able to identify what is being requested and to collate a sequence of instructions over a prolonged period of time, i.e., an asynchronous stream of interactions. Therefore the binding must support the following end-point identification information:

- System identification – the unique system identifier. This identifier may be used in one of several parts of the physical addressing system;
- Service identification – the service entities that are supporting the business process, e.g., the 'Person Management Service';
- Transaction identification – the service transaction required to support the business process, e.g., 'createGroup' for creating a 'Group' object;
- Operation identification – the operation of abstract-API call that is used to trigger the transaction(s) message exchange, e.g., 'createPerson' for creating a 'Person' object. Several transactions may be supported by a single 'operation', e.g., 'deletePersons' make contain a set of 'deletePerson' transactions;
- Object identification – the object identifier that is being manipulated by the operation, e.g., the unique identifier for a 'Person'.

4.2.2 Conventions for Definitions of Behaviors

The abstract API for an application service defines its behaviors as operations in UML Class Diagrams. UML Interaction Diagrams show how the messages representing these operations interact with messages from other behaviors. These behaviors contain:

- Supplied parameters – these are values/objects that are supplied to the operation and in UML these are denoted 'in' parameters;
- Returned parameters– these are values/objects that are returned by the operation and in UML these are denoted 'out' parameters.

These values/objects are represented as XML and must have an associated XSD to act as the control validation source. Each and every operation will have a 'Status' Object that must be returned. The 'Status' object contains the status information that describes the end state of the 'Respondent' once it has attempted to process the received message. This status may contain error information if the processing has resulted in an error state/condition in the 'Respondent'. The causes for error are:

- Business rule violations – this is related to the semantic correctness of the data. Errors include such things as invalid object identifiers, attempted access to restricted information, etc.
- Invalid data content – the data supplied has syntactic errors, e.g., the XML may be invalid. There is also the case where too much information is supplied, i.e., optional data elements are supplied for a service that can only store the mandatory fields;
- Unavailable resources – these arise where the end-system does not have the resources to support the request. This may be due to conditions such as insufficient data storage, busy message buffers, etc.
- Related common service failures – these errors occur when more than one service is required to interact to provide the correct end-to-end operation. Typical examples include authorization, authentication, etc. where the supplied information carried in the SOAP header will be invalid for the required transaction;
- Infrastructure service failures – failures of the underlying communications infrastructure. All of these errors will result in a SOAP Fault being received, i.e., there may be no other form of error information in the SOAP message.

1. The terminology 'Initiator' and 'Respondent' is taken from the Open Systems Interconnection seven layer Reference Model (OSI/RM) work.

Therefore the binding SOAP messages must support the following behavioral information:

- Service identification – the service being supported by the SOAP messages;
- Operation identification – the operation being requested. This operation may consist of more than one business transaction;
- Object identification – the unique identifier for the object. This identifier must be passed as one of the 'Supplied Parameters';
- Supplied parameter XML – the information that is sent from the 'initiator' to the 'respondent'. It is contained in the body of the corresponding SOAP message;
- Returned parameter XML – the information that is sent from the 'respondent' to the 'initiator'. It is contained in the body of the corresponding SOAP message. This excludes the status object information;
- Status object XML – this object contains all of the status information carried in the response-type messages. This status object is used to contain any error reporting information. It is contained in the header of the corresponding SOAP message;
- Message choreography – each message must carry a unique identifier to enable the different parts of the end-to-end choreography to be matched. All response-type messages must identify the corresponding request-type messages. This choreography is determined by the type of messaging model (see Section 5).

4.3 Common Services Layer

At the current time there is no recommendation by IMS on how the interaction between application and common services should be choreographed. Once the appropriate specifications have matured, e.g., WS-CDL and BPEL4WS, then the corresponding recommendations will be made.

4.3.1 Error and Exception Handling

The application services exchange status information using the SOAP header status information object, defined as part of the binding by IMS/GLC. This status object is used to contain any error state information. Further error information on the SOAP communication is also supplied in the form of SOAP fault messages; this fault information is again supplied within the SOAP header. The IMS GWS Base Profile does not describe how the status information should be processed and what resulting remedial action should be taken. In the case of error status information this means that the exception handling in the end-systems is undefined. In some implementations an external common service may be invoked to support exception handling.

4.3.2 Security

System security is of increasing importance in the design of the overall system architecture. From the perspective of the IMS GWS Base Profile many aspects of security within the system's architecture are out of scope. This functionality can be supported using the appropriate common services. The IMS GWS Security Profile [GWS, 05d] defines how security architectures are supported by the SOAP messaging mechanism. The implications for common service requirements from the perspective of security are:

- Session – session management could be supplied using a common service thereby enabling functionality such as secure/private application multicasting, replicated repository management, etc.;
- Authentication & Authorization – a single sign-on authentication and authorization service is highly desirable to enable end-systems to minimize the required user-interaction across multiple applications within and between enterprises;
- Encryption & Digital Signing – a key management common service, e.g., X509, should be used to support data encryption and digital signatures.

4.3.3 Routing

SOAP is a point-to-point protocol. It is possible to implement SOAP on top of many different communications architectures, e.g., message broker, store-and-forward, etc. This may require SOAP message routing/switching and the spoofing of end-to-end connectivity. The ways in which these services interact with the underlying IMS GWS Base

Profile messaging models (synchronous, asynchronous, etc.) is undefined but an implementation of an intermediate system must ensure that the behavior of the end-systems is consistent within the corresponding information model and binding specifications.

4.4 Infrastructure Layer

In Figure 4.3 the Infrastructure Layer begins with the XML-based Envelop sub-layer. For WSDL this takes the form of either SOAP or HTTP-POST or HTTP-Get. The generic transport sub-layer is assumed to be HTTP or HTTPS, on top of TCP/IP.

4.4.1 Quality of Service

The available Quality of service is that sustained by the TCP.

4.4.1.1 Best Effort

Most forms of TCP provide a ‘best effort’ service, i.e., there is no guaranteed minimum data rate or maximum end-to-end delay. Therefore, the quality of service perceived by the end-system is determined by: the other load on the network infrastructure (this will depend upon the time of day); the bandwidth of the network links and the error conditions on those links; the number and capabilities of the switches and the routers traversed; and to a lesser extent the physical propagation distance between the end-points.

4.4.1.2 Session Support

There is no session support within TCP

4.4.1.3 Duplicate Elimination

The byte streaming operation of TCP ensures that data duplication does not occur across the physical network infrastructure between the linked TCP end-points. However, if the high level protocols duplicate information then TCP will not detect this duplication. At present the IMS GWS Base Profile does not provide reliable SOAP messaging but does provide the mechanisms by which an implementation can create such a capability, e.g., unique message numbering – see Section 6.

4.4.1.4 Guaranteed Delivery

Once the TCP process accepts a data transfer request it guarantees data transfer or issues a data transfer failure status. Data errors do not occur because the streaming protocol retransmits data that is corrupted. However, if invalid or corrupted data is submitted to the TCP process then that information is guaranteed delivered reliably to the destination!

4.4.2 Message Packaging

4.4.2.1 Single XML Payload

The XML payload is carried in the SOAP body. The validity of this payload is determined by the associated XML control document (XSD).

4.4.2.2 Multiple Payloads

The IMS GWS Base Profile requires that only a single XML payload be supported in the SOAP body. Therefore, multiple XML payloads require the usage of an XML container XSD. The container is defined such that it supports one or more of the core data XML structures.

4.4.3 Generic Transport Layer

The SOAP messaging mechanism supports several transport protocols – in terms of the Internet Protocol Suite these transport protocols are formally named Application Protocols. All of these protocols operate using TCP/IP.

4.4.3.1 HTTP/HTTPS Conventions

These are the protocols recommended in the IMS GWS Base Profile for the exchange of the SOAP messages.

4.4.3.2 SMTP Conventions

This protocol is not supported as part of the IMS GWS Base Profile.

4.4.3.3 FTP Conventions

This protocol is not supported as part of the IMS GWS Base Profile.

5. Implementation Profiles

5.1 Messaging Models

5.1.1 Synchronous Request/Response Messaging Choreography

The message choreography for the synchronous message binding is shown in Figure 5.1. This diagram shows how the ‘Service Requester’ exchanges information with the ‘Service Provider’ using the corresponding communications handlers “Service Request Comms Hndlr’ and ‘Service Provider Comms Hndlr’ respectively.

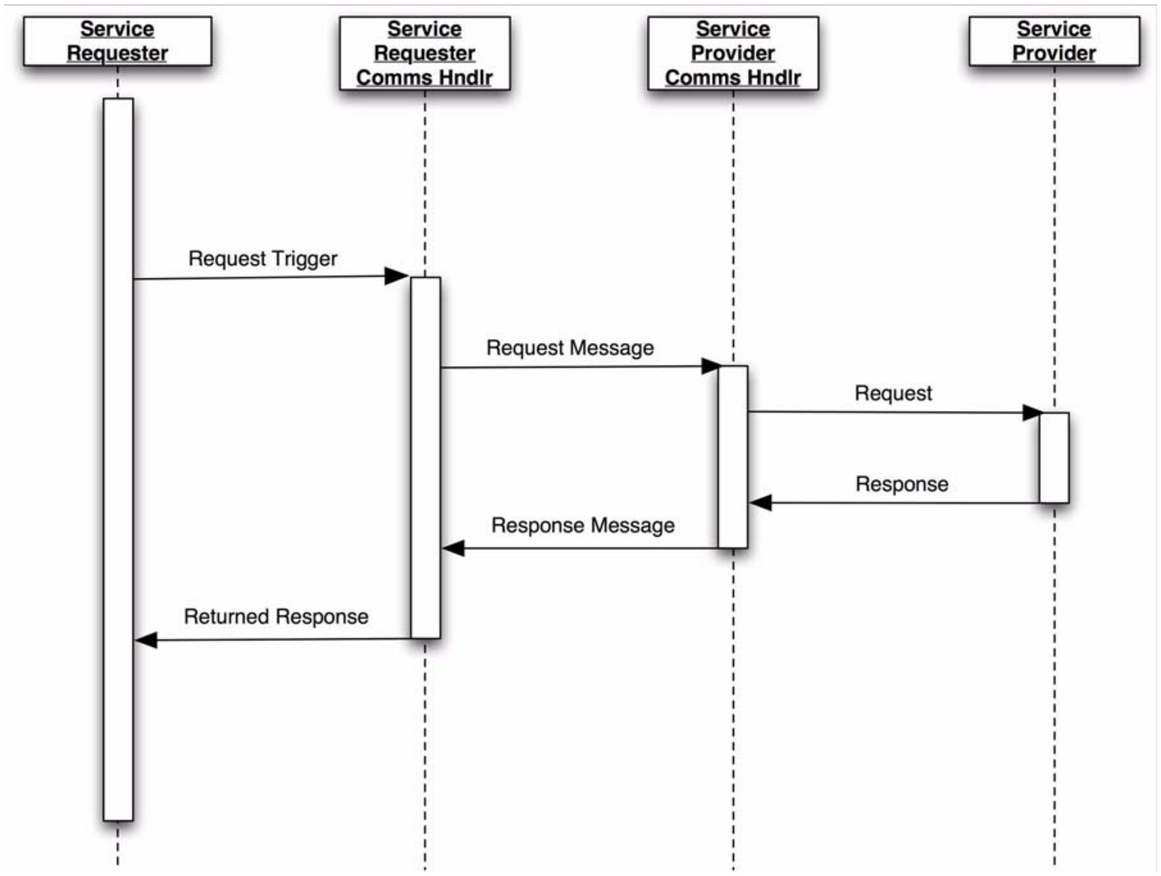

Figure 5.1 Synchronous request/response message choreography.

The message choreography for error-free operation is:

a) The ‘Service Requester’ triggers the request. This is a synchronous request and as such the ‘Service Requester’ is blocked until the ‘Service Provider’ has replied to the request;

b) The ‘Service Requester Comms Hndlr’ constructs and transmits the request message (for the Base Profile this takes the form of a SOAP message). The ‘Service Requester Comms Hndlr’ waits until the corresponding response message has been received (good system design requires that an appropriate fault recovery mechanism is implemented to avoid the system becoming dead-locked – this is outside the scope of the Base Profile);

c) When the ‘Service Provider Comms Hndlr’ receives the request it is processed and the encapsulated request is passed to the ‘Service Provider’;

d) The ‘Service Provider’ processes the request and constructs the appropriate response. Once this response has been accepted by the ‘Service Provider Comms Hndlr’ the ‘Service Provider’ is free to process other activities;

e) The Service Provider Comms Hndlr’ constructs and transmits the appropriate response message (for the Base Profile this takes the form of a SOAP message);

f) Eventually, the ‘Service Requester Comms Hndlr’ will receive the response message. This results in the corresponding response information being returned to the ‘Service Requester’. This unblocks the ‘Service Requester’, i.e., it is now free to issues further requests.

Error recovery is beyond the scope of this Base Profile definition. It is the responsibility of the implementation of this choreography to ensure the correct synchronous exchange of information. The actual WSDL/SOAP binding realization [GWS, 05d] makes available several features that can be used provide error recovery.

5.1.2 Asynchronous Messaging Choreography

At the current time IMS has not authorized the publication of the Asynchronous Communications messaging choreography for Final Release. This is because there are no validated implementations of this technique and there is work underway in W3C that could result in that IMS approach becoming proprietary. This work remains published as Public Draft material [GWS, 05f].

5.1.3 Polled Messaging Choreography

At the current time IMS has not authorized the publication of the Polled Communications messaging choreography for Final Release. This is because there are no validated implementations of this technique and there is work underway in W3C that could result in that IMS approach becoming proprietary. This work remains published as Public Draft material [GWS, 05f].

5.1.4 Publish and Subscribe

To be studied in a later version of this document.

5.2 Organizational Profiles

There are numerous issues to consider regarding of how web services are used and invoked based upon whether they requestor and provider cross either organizational bounds or “trust” bounds. The reader is encouraged to review both “intra” and “inter” organizational communication issues, since issues described in one may have some relevancy in the other.

5.2.1 Intra-organizational Communications

The issues for consideration are:

- Will the learner need a “Single Sign On” Experience?
- Can it be assumed that all systems “trust” each other within the Organization, e.g., can the organization truly be modelled as a single “Unit of Trust”?
- Is there a central system or “service” that can aggregate and serve as the central point for “Federated Identity”:
    - User identity management and authentication
    - User Attributes
    - User Entitlements (authorizations).

5.2.2 Inter-organizational Communications

The issues for consideration are:

- Are all providers of services pre-established, or are providers established dynamically through some lookup or registry?
  - What considerations exist for any transaction needs across the systems and/or organizations as part of "two-phase" commit?
  - Is there a need for a third party "certificates" to validate the identities of the service provider and requestor?
- Does one organization become the "system of record" for identity management? If so, does it:
  - Handle authentication, and communicate current status of user's authentication?
  - Handle "log out", and communication of that activity to the other organizations?
  - Have all needed entitlements information on the user?
  - Have all needed user attributes and entitlement information?
  - What user attributes can be shared?
- What are the parameters for a Service Level Agreement (SLA)?
  - Availability?
  - Accessibility?
  - Performance?
  - Reliability?
  - Security?
  - Agreed upon Roles/Requestors that can invoke the service, with what entitlements?
- Are all communications between Organizations that are enrolled in a "Circle of Trust", a.k.a. "Trust Chain;
  - Is there a need for "non-repudiation" for services requested and performed?
  - Does some of the data in the XML package need to be "blacked out" or encrypted before the request (data is sensitive and not needed as part of the service)?
  - Does the request/response need to be encrypted (SSL, TLS) to ensure confidentiality?
- Do some of the SLA and encryption constraints affect what messaging model should be used?

6. Best Practice Recommendations

6.1 Supporting Status/Error Codes and Exception Handling

When a behavior-based IMS information model is developed that is to be mapped onto SOAP messages then each operation is required to return status information. This status information provides contextual information about the completed success or otherwise of the operation. There are two types of status information that are available to the end-systems:

- Business transaction – these are the status reports that reflect the business logic of the transactions being exchanged by the end-systems. This status information will be contained within the message header under a specially defined data structure. The status information contained herein is also used to contain any error codes, i.e., error reporting is handled as a subset of status information reporting;
- SOAP fault– this is the SOAP fault codes that are reported by the SOAP messaging infrastructure and which are carried in the SOAP message headers. A two-tier fault message code is available with the top-level being a non-extensible enumerated list.

The status information for the business transactions is carried in a single status information object that contains the following sub-structures:

- CodeMajor – The major code assigned to the status block. This is a fixed enumerated list. This is used in conjunction with the ‘severity’. The list of codes is: Success, Processing, Failure and Unsupported.
- Severity – the severity of the status report. This is a fixed enumerated list. This is used in conjunction with the CodeMajor. The list of codes is: Status, Warning and Error.
- CodeMinor – this is a detailed report code structure that is used to identify specific causes of failure. A set of codes can be defined for each transaction – these will be defined as part of the corresponding Information Model.

The interpretation of the ‘CodeMajor/Severity’ matrix is shown in Table 6.1.

The IMS specifications do not describe how the status information is to be handled within an end-system, i.e., this will depend on how the abstract API is physically realized within an implementation. However, it is important that an implementation can:

- Combine the transaction status information and the SOAP fault error codes in a single integrated status reporting mechanism. Any other system failure information that is made available by an implementation should also use the same mechanism;
- Examine the status information reported after the completion of the appropriate phase of an operation and especially once the operation has been completed. This may require an explicit status information call or it may be reported as part of the API call;
- Differentiate the status information reports for each transaction within an operation. Remember that some specifications provide operations that can contain more than one transaction request and that a different status report may be given for each of those transactions.

Exception handling is the system’s response to a known or unknown error condition. Exception handling is outside the scope of the IMS GWS Base Profile specification. However, an error condition should not cause the end-systems to fail in an uncontrolled manner. The requirement for every operation to return status information is to enable an implementation to terminate in a controlled fashion.

Table 6.2 Interpretation of the 'CodeMajor/Severity' matrix.

Severity	CodeMajor			
	Success	**Processing**	**Failure**	**Unsupported**
Status	The request has been completed successfully.	The request is being processed by the target, i.e., the request has been received and acknowledged by the target communications handler.	The request has failed. The associated CodeMinor information contains the more detailed reason for the failure of the request. The associated business specification must define this set of codes.	Target does not support the requested operation.
Warning	Some of the request has been completed successfully, e.g., partial storage of the data structure sent.	The request is being processed (this does not imply reception by the target communications handler) but it has not yet been acknowledged as received by the target.	Not permitted.	Not permitted.
Error	Not permitted.	An error has been detected in the immediate transmission communications handler, i.e., the message has not left the end-system.	The request has failed but it was issued from the local communications handler. Detailed failure reports could be included. The SOAP fault codes are reported using this Severity/ CodeMajor value (supplied in the CodeMinor object).	Target does not recognize the requested operation, i.e., it is an unknown service extension.

6.2 Supporting Different Communications Messaging Models

The IMS approach to supporting different modes of communication is to hide this within the binding, i.e., the information model is agnostic of the form of communication model used in the binding. Therefore binding is designed with the following features:

- Each message has a unique identifier. Message identifiers are not necessarily sequentially allocated, i.e., the identifier is not a message sequence number;
- All response and acknowledgement messages will supply the initiating message identifier. This will enable the implementation to correlate the cause and effect messages;
- A message identifier information will be carried in the header of the appropriate message construct, i.e., SOAP messages in this case.

The following functional capabilities are not supported in the binding and as such an implementation must supply these as and when required (some of these capabilities may be supported as and when the appropriate web service specifications have been developed and ratified):

- Error detection and correction – all messages are assumed to be correct, i.e., there are no bit errors;
- Message loss and duplication avoidance – it is assumed that no messages are lost or duplicated by the communications infrastructure.

SOAP is either a single message or two-message communication mechanism, i.e., it does not enable a complex choreography for more than two messages. Therefore, asynchronous and polled communication message models require sophisticated usage of WSDL and its SOAP messaging definitions. These require multiple sets of WSDL file definitions that an implementation has to combine to create the full message choreography. This approach is fully explained in the IMS GWS WSDL Binding Guidelines [GWS, 05d].

6.2.1 Synchronous Messaging Status Codes

A single status code is returned in the response message for each transaction.

6.2.2 Asynchronous Messaging Status Codes

At the current time IMS has not authorized the publication of the Asynchronous Communications messaging choreography for Final Release. This is because there are no validated implementations of this technique and there is work underway in W3C that could result in that IMS approach becoming proprietary. This work remains published as Public Draft material [GWS, 05f].

6.2.3 Polled Messaging Status Codes

At the current time IMS has not authorized the publication of the Polled Communications messaging choreography for Final Release. This is because there are no validated implementations of this technique and there is work underway in W3C that could result in that IMS approach becoming proprietary. This work remains published as Public Draft material [GWS, 05f].

6.3 Versioning

The following notes on Versioning are based on the W3C’s notes on Versioning XML Languages as can be found from: http://www.w3.org/2001/tag/doc/versioning-20031003. In broad terms, the approaches to versioning are:

- None – no distinction is made between versions of the language. Applications are either expected not to care, or they are expected to cope with any version they encounter;
- Compatible – designers are expected to limit changes to those that are either backwards or forwards compatible, or both i.e.
    - *Backwards compatibility*: Applications are expected to behave properly if they receive an instance document of the “older” version of a language. Backwards compatible changes allow applications to behave properly if they receive an “older” version of the language
    - *Forwards compatibility:* Applications are expected to behave properly if they receive an instance document of the “newer” version of a language. Forwards compatible changes allow existing applications to behave properly if they receive a “newer” version of the language.
- Flavors – applications are expected to behave properly if they receive one of a set of flavors of the document type;
- Big bang – applications are expected to abort if they see an unexpected version.

There is no single approach that is always correct. Different application domains will choose different approaches. Just as there are a number of approaches, there are a number of strategies for implementing an approach. The Internet - including MIME, markup languages, and XML languages have successfully used various strategies, either singly or in combination such as XML Namespaces and Schema. For any given approach, some strategies may be more appropriate than others. Among the strategies we find:

- Must Understand – receivers must understand all of the elements and attributes received and are expected to abort processing if they do not. SOAP processors must understand headers that are explicitly identified to be mandatory;

- Must Ignore – receivers must ignore elements or attributes that they do not understand. Sometimes the must understand and must ignore approaches can be combined for more selective use. SOAP processors must ignore headers they do not recognize unless the header explicitly identifies itself as one that must be understood. There are 2 variations of the 'Must Ignore' strategy:
  - *Must Ignore All:* This variation on must ignore requires the receiver to ignore an element or attribute it does not understand and, in the case of elements, all of the descendents of that element. Most data applications, such as Web services that use SOAP header blocks or WSDL extensions, adopt this approach to dealing with unexpected markup
  - *Must Ignore Container*: This variation on 'Must Ignore' requires the receiver to ignore an element or attribute that it does not understand, but in the case of elements, to process the children of that element. The Must Ignore Container practice was described in HTML 2.0;
- Explicit Fallback – a language can provide mechanisms for explicit fallback if the extension is not supported. MIME provides multipart/alternative for equivalent, and hence fallback, representations of content. HTML 4.0 uses this approach in the NOFRAMES element. In XML, the XML Inclusions specification Xinclude provides a fallback element to handle the case where the putatively included resource cannot be retrieved.
- Explicit Testing – a language can provide a mechanism for explicit testing. The XSLT Specification provides a conditional logic element and a function to test for the existence of extension functions. This allows designers of style sheets to deal with different receiver capabilities in an explicit fashion.

Languages can choose a mixture of approaches. For example, XSLT provides both an explicit fallback mechanism for some conditions and explicit testing for others. The SOAP specification, another example, specifies 'Must Ignore' as the default strategy and the ability to dynamically mark components as being in the 'Must Understand' strategy.

6.3.1 IMS Versioning Solution

The IMS solution is to add a separate version structure that is contained in the SOAP header. This structure is optional and when it is absent the default version is GWS Base Profile v1.0.

6.4 Using the Messaging Questionnaire Template

The messaging questionnaire, supplied in Appendix B, looks at three messaging mechanisms:

- Request data update;
- Read request/response;
- Data monitoring.

The following sub-sections identify the set of questions that should be answered for each of the messaging mechanism listed above. The numbers shown in the square brackets ('[]') denote the equivalent question number in the Messaging Questionnaire. The set of questions identify two systems 'X' (the Source) and 'Y' (the Target) that are communicating using the corresponding messaging mechanism.

6.4.1 Use Case #1: Request Data Update / ACK (or NAK)

The key questions for systems that are based upon:

- 'X' sends Create / Update / Delete to 'Y';
- 'Y' sends back Response to 'X'.

1. [3] Accept only or Reject possible?

Can the 'Y' system reject the 'X' change? What if the version number is wrong or (worse) if the Request message doesn't pass the "well formed" or "valid" checks? What if the change is illegal, e.g., no such object?

Should all errors (system level, XML level, business level) be reported in the same way?

2. [4] Synchronous or Asynchronous Response?

Is the ‘Y’ response expected immediately? If not, how is the response correlated to the request?

Does the Response imply that the ‘Y’ database update has actually been made? If so, can ‘X’ always leave its Request pending until this occurs (even if ‘Y’ is in another organization)?

Alternatively, can the Change Request be acknowledged immediately, and the actual Response be returned later (asynchronously), after the ‘Y’ database update is actually done or if an error occurred while processing the Request Message, not done?

Is there a maximum time a given ‘X’ system is willing to wait for the Response and does the ‘Y’ system need to know this time? If so, and if not simply pre-wired, the infrastructure may have to support “session state”.

Is there ever a need for the ‘X’ system to cancel the Request before it is executed?

3. [5,6] What level of Quality of Service (QoS) is required?

If the Response message is lost, and ‘X’ times out and resends, does ‘Y’ need to detect delivery of a duplicate message, i.e., is it an error to issue this Update message twice? Should the infrastructure prevent duplicate message delivery?

Must the recipient (or the network) be active for the Request to successfully be made? Should the infrastructure support guaranteed message delivery (making life easier for the requesting ‘X’)?

4. [9,10,11] How is Partner Location determined?

Is the location of ‘Y’ “pre-wired” in ‘X’? If more than one ‘Y’ is connected to ‘X’, how does ‘X’ know which ‘Y’ to inform of the change?

Is there a “globally unique” Record ID? If so, is it global to ‘Y’ (and all its connected ‘X’ partners) or global across all possible -compliant ‘Y’? If the former, then a single ‘X’ cannot be connected to multiple ‘Y’.

5. [12, 13] Is a Data Monitoring capability ever necessary?

Does ‘X’ ever need to synchronize with data maintained in ‘Y’? In other words, are there ever any changes to ‘Y’ data **not** caused by ‘X’ that ‘X’ needs to be informed of? If so, than either ‘X’ must itself be a web service, and be known to ‘Y’, which will use the ‘X’ API to post such changes. Alternatively, ‘X’ must “subscribe” to posted changes from Y.

Is it required to allow a 3rd party (e.g., an “Event Logger”) to track all such changes (from either ‘X’ or ‘Y’)? If so, an infrastructure supporting some variant of the publish/subscribe model offers definite advantages.

6. [14,15,16,17] What level of Security is required?

Are there any security requirements for communication between ‘X’ and ‘Y’? Which direction? What type?

A. Encryption of data?
B. Authentication of ‘X’ as source of Change Request to ‘Y’?
C. Authorization of ‘X’ to issue the Change Request for this data?

How much of such security requirements should be supplied by the infrastructure?

7. [18, 19, 20] Is “One-shot” communication exchange enough or is Session support needed?

Do ‘X’ and ‘Y’ have longer running conversations? Is there a session established between them in which state is maintained? If so, what are some of the state variables for such a session?

A. Security “Cookie”?
B. Version number of each participant?
C. Optional services supported/not supported.

How much of the above information that both ‘X’ and ‘Y’ need to agree on, is discovered dynamically? How much of this information is pre-configured by the System Administrator?

8. [21, 22] How are Optional Elements resolved?

Are there any optional elements in the schema that define this exchange?

If so, how do the two parties ensure that there is no case in which one side does not support such an optional element, while the other side depends upon it?

9. [23] Can there be a range of different Transports used?

What is the range of different transports such 'X' to 'Y' communication could logically be expected to occur across?

A. HTTP;
B. HTTPS;
C. IIOP (Corba) or DCOM;
D. MOM (MQ/Series, MSMQ, JMQ, etc.);
E. SMTP (note – supports multiple binary payloads);
F. Other (FTP, etc.).

10. [28, 29] Is support for Batch Updates necessary?

Can the 'X' Update reference data in more than one object? If so, does the Update message schema support aggregates?

11. [30, 31, 32] Are there any Transaction considerations in this database update?

Can the 'X' Change Request ever be considered as part of a larger (perhaps multi-party) transaction? For example, can it propagate to a 'Z' system, which might have to complete its data updates before the 'Y' system could return an acceptance Response?

If so, how are the boundaries of the transaction identified in the documentation? Need any transaction identifier be conveyed on the wire?

12. [33] What does the Version Number mean?

Does the version number of the messages involved in this exchange refer to:

A. The particular version of the release supported by both sides?
B. The version of the data object being updated?
C. The version of the schema for the Change Request or Response message?

13. [34, 35] What are the Message Packaging requirements?

Can either the Request or Response consist of multiple independent payloads? Can the format of any of these payloads be other than XML?

14. [36, 37] Legality

Are there any requirements for non-refutability of either the Update Request or the subsequent Response? Are there any requirements for guaranteed non-alterability of message data?

15. [38, 39, 40, 41, 42] Operation Specifics

Are partial modifications to the data record possible? If so, must the original values of the elements being changed be included in the Change Request as well? If not, how are the "positions" of these elements within the total Object Schema indicated?

When the Change Request from 'X' indicates "Create", is the GUID for that object included in the message? Note that this could cause problems if 'Y' owns GUID creation for its objects. If it is NOT so included, is the value of the GUID that 'Y' assigns, returned in the Response message to 'X'?

6.4.2 Use Case #2: Read Request / Response

The key questions for system that are based upon:

- 'X' sends Read Request to 'Y';
- 'Y' sends back Response to 'X'.

1. [4] Synchronous or Asynchronous Response?

Is the ‘Y’ response expected immediately? If not, how is the response correlated to the request?

Is there a maximum time a given ‘X’ system is willing to wait for the Response and does the ‘Y’ system need to know this time? If so, and if not simply pre-wired, the infrastructure may have to support “session state”.

2. [5, 6] What level of Quality of Service (QoS) is required?

Must the recipient (or the network) be active for the Request to successfully be made? Should the infrastructure support guaranteed message delivery (making life easier for the requesting ‘X’)?

3. [9, 10, 11] What is known about the Partner?

Is the location of ‘Y’ “pre-wired” in ‘X’?

Is there a “globally unique” Record ID? If so, is it global to a single ‘Y’ or global across all possible -compliant ‘Y’? If the former, then a single ‘X’ cannot be connected to multiple ‘Y’.

If the requestor is a device (ex: for a Price Lookup, the client can be an in-store server, a POS terminal or a scanner) then the Response could be different (full information to single line display).

If the response does indeed depend upon knowing the type of requestor, than how does the responder get this information?

4. [12, 13] Is a Data Monitoring capability ever necessary?

Does X ever need to synchronize with data maintained in Y? In other words, are there ever any changes to Y data that X needs to be informed of? If so, than either X must itself be a web service, and be known to Y, which will use the X API to post such changes. Alternatively, X must “subscribe” to posted changes from Y.

Is it required to allow a 3rd party (ex: an “Event Logger”) to track all such Y changes? If so, an infrastructure supporting some variant of the publish/subscribe model offers definite advantages.

5. [14, 15, 16, 17] What level of Security is required?

Are there any security requirements for communication between ‘X’ and ‘Y’? Which direction? What type?

A. Encryption of data?
B. Authentication of ‘X’ as source of Change Request to ‘Y’?
C. Authorization of ‘X’ to issue the Change Request for this data?

How much of such security requirements should be supplied by the infrastructure

6. [18, 19, 20] Is “One-shot” communication exchange enough or is Session support needed?

Do the X and Y systems have longer running conversations? Is there a session established between them in which state is maintained? What are some of the state variables for such a session?

A. Security “Cookie”?
B. Version number of each participant?
C. Optional services supported/not supported.

How much of the above information that both ‘X’ and ‘Y’ need to agree on, is discovered dynamically? How much of this information is pre-configured by the System Administrator?

7. [21, 22] How are Optional Elements resolved?

Are there any optional elements in the XML schema that define this exchange?

If so, how do the two parties ensure that there is no case in which one side does not support such an optional element, while the other side depends upon it?

8. [23] Can there be a range of different Transports used?

What is the range of different transports such 'X' to 'Y' communication could logically be expected to occur across?

A. HTTP;
B. HTTPS;
C. IIOP (CORBA) or DCOM;
D. MOM (MQ/Series, MSMQ, JMQ, etc.);
E. SMTP (note – supports multiple binary payloads);
F. Other (FTP, etc.).

9. [24, 25] Is support for multi-object retrieval necessary?

Can the 'X' Request ask for information on more than one object? How flexible is the query (i.e., list of specific Record IDs, all the way up to full SQL capability on object collection)?

If multiple objects result, does the Response message schema support aggregation?

10. [33] What does the Version Number mean?

Does the version number of the messages involved in this exchange refer to:

A. The particular version of the release supported by both sides?
B. The version of the data object being requested?
C. The version of the schema for the Request or Response message?

11. [34, 35] What are the Message Packaging requirements?

Can either the Request or Response consist of multiple independent payloads? Can the format of any of these payloads be other than XML?

12. [36, 37] Legality

Are there any requirements for non-refutability of either the Request or the subsequent Response? Are there any requirements for guaranteed non-alterability of message data?

13. [30, 31, 32] Are there any Transaction considerations in this data request?

Can the 'X' Read Request ever be considered as part of a larger (perhaps multi-party) transaction? For example, can it propagate to system 'Z', which might have to supply some of its data before the 'Y' system could return an acceptance Response?

If so, how are the boundaries of the transaction identified in the documentation? Need any transaction identifier be conveyed on the wire?

6.4.3 Use Case #3: Data Monitoring

The key questions for system that are based upon:

- 'X' publishes an Event (usually indicating an update to its object(s);
- 'Y' and 'Z' receive the Event via a 3rd party Message Broker (MOM).

1. [1, 2] How did 'Y' and 'Z' get selected to receive the event message from 'X'?

Did 'Y' and 'Z' dynamically pre-subscribe to receive 'X's Event, or did they pre-subscribe to receive messages on a specific "topic" (typically changes to objects of a specific type) which 'X' was identified as "owning"?

If the latter, did 'X' have to pre-register to post events on this topic?

2. [6, 7, 8] What level of Quality of Service (QoS) is required?

Must each recipient (and the network) be active for the Event to successfully be generated?

Should the infrastructure support guaranteed message delivery (implying that a newly reactivated application receives all the events previously queued for it).

Must events be delivered in the order in which they were posted?

Does the recipient have to be notified when all events are successfully delivered?

3. [43, 44, 45, 46, 47] What level of Security is required?

Are there any security requirements for communication between the Message Broker and the ‘X’, ‘Y’ and ‘Z’ systems? Does the Message Broker need to enforce:

A. Encryption of data?
B. Authentication of ‘X’ as source of Change Request to ‘Y’?
C. Authorization of ‘X’ to issue the Change Request for this data?

Can the levels of security for event propagation be set by the issuing application ‘X’ or must all security levels be pre-configured by the administrator of the Message Broker?

4. [21, 22] How are Optional Elements resolved?

Are there any optional elements in the XML schema that define this Event?

If so, how do all the parties ensure that there is no case in which one side does not support such an optional element, while the other side depends upon it?

5. [26, 27] Is support for multi-object retrieval necessary?

Can the ‘X’ event report on changes to more than one object instance? If so, does the Event message schema support aggregates?

6. [33] What does the Version Number mean?

Does the version number of the Event involved in this exchange refer to:

A. The particular version of the release supported by both sides?
B. The version of the data object being reported?
C. The version of the Event message schema?

7. [34, 35] What are the Message Packaging requirements?

Can the Event message consist of multiple independent payloads? Can the format of any of these payloads be other than XML?

8. [36, 37] Legality

Are there any requirements for non-refutability of the Event? Are there any requirements for guaranteed non-alterability of message data?

9. [38, 39, 40] Operation Specifics

Are partial modifications to the object referred to in the Update Event possible? If so, do the original values of the elements being changed need to be included in the Update Event as well? If not, how are the “positions” of these elements within the total Object Schema indicated?

10. [30, 31] Are there any Transaction considerations in this notification of database update?

Can the Event ever be considered as part of a larger (perhaps multi-party) transaction? For example, can it generate other events that must be “tied” to the original?

If so, how are the boundaries of the transaction identified in the documentation? Need any transaction identifier be conveyed on the wire?

7. Extending the Base Profile

7.1 Proprietary Extensions

There are three methods by which the functional capability of the base profile can be extended:

a) Addition of new SOAP messages – the addition of new business transactions and the usage of new messaging models requires the creation of new SOAP messages. The addition of new business transactions should result in a change to the specification itself. Therefore, the application of the transformation rules will result in the automatic generation of the new SOAP messages. Support for a new messaging model, will require the creation and application of a new set of transformation rules;

b) Extensions to the SOAP header – the current IMS GWS Base Profile makes use of the SOAP header to contain the application-to-application transaction status information. Other specifications, e.g., WS-Security, also use the SOAP header to contain the appropriate information. It is recommended that proprietary extensions to the SOAP header maintain the current usage patterns;

c) Extensions in the data contained within the SOAP body – the SOAP body contains the XML instance that is used to represent the parameters defined for the transaction operations in the specification. There may be a need to add new parameters or to extend the XML structures of the current parameters. Extending the XML data structures must use the IMS extension mechanism otherwise the SOAP message receiver processor will not be able to marshal the messages, i.e., the receiver must know 'a priori' the XML structure of the received messages. New parameters may require a new XSD or at the very least will require a change to message structure XSD.

7.2 Further Work in V2.0

The areas for further work in the development of the IMS GWS Base Profile are:

a) Inclusion of conformance statements – the WS-I describes how conformance statements can be embedded in the WSDL description. These statements can be used to determine if the SOAP messages conform to the profile. WS-Policy is an alternative mechanism that is under development by W3C;

b) Adoption of SOAP 1.2 – the current profile is based upon SOAP 1.1 but SOAP 1.2 is now available. We will not consider changing the IMS GWS Base Profile until reliable tools are available to support using SOAP 1.2;

c) Adoption of WSDL 2.0 – W3C is developing WSDL 2.0 (this was originally referred to as version 1.2). Changing the intermediate representation should not alter the actual SOAP messages to be generated but will enable improved expression to allow more complex bindings to be described in the WSDL file. We will not consider changing the IMS GWS Base Profile until reliable tools are available to support using WSDL 2.0;

d) Adoption of UDDI 2.0 – UDDI 2.0 is not widely adopted. We will not consider adding the usage of UDDI to the IMS GWS Base Profile until there is sufficient practical experience of using UDDI;

e) Reliable messaging – reliable application-to-application communications requires the usage of the corresponding reliable SOAP messaging protocol. TCP does not operate at the right level in the communications stack to provide coverage of the SOAP messages;

f) Message choreography – the more complex messaging models, i.e., 'asynchronous', 'polled' and 'publish-and-subscribe' need to be supported (IMS proprietary implementations for asynchronous and polled are available in the Public Draft version of this document [GWS, 05a]). The usage of WSDL 2.0 may resolve this issue;

g) Service choreography – the interaction between the application and common services needs to be described using the appropriate service choreography. This will enable us to definitively specify how services may interact. The usage of BPEL4WS will be investigated for this choreography.

8. Conformance to the Base Profile

8.1 International Conformance Programme

The IMS International Conformance Programme has developed the Application Profile Guidelines [APG, 05a], [APG, 05b]. These guidelines describe how a specification can be tailored for a particular domain of usage by creating the appropriate Application Profile. A collection of Application Profiles is produced to create the corresponding Domain Profile (in many cases a domain of usage requires several related Application Profiles to be produced).

8.2 Base Profile Conformance

As per the Conformance Specification, the test specification is to be produced from two perspectives:

- 'Service Requester' – the driver and consumer of the service;
- 'Service Provider' – the provider of the service.

8.2.1 Service Requester Perspective

For behavior that is defined in the UML-interface diagram of the specification the following test set must be defined to create the corresponding SOAP/HTTP messages:

a) Unsupported service report – the service request is rejected because it is not supported by the service provider (this is also true for requests requiring service extensions);

b) Successful service provision with only mandatory elements – the service is correctly supported when the supplied data contains only the mandatory data elements;

c) Successful service provision with mandatory and optional elements – the service is correctly supported when the supplied data contains any combination of optional data with all of the mandatory data (this includes all of the optional data, any single piece of optional data and combinations of multiple optional data);

d) Successful service provision with mandatory and optional elements – the service is correctly supported when the supplied data contains any combination of optional data with all of the mandatory data (this includes all of the optional data, any single piece of optional data and combinations of multiple optional data);

e) Correct generation of error status codes – the requester must correctly handle the appropriate error codes when invalid data was supplied, when the system was in a state inconsistent with the request and when there was a failure of some component in the system, e.g., full database in the service;

f) Data overload handling – observation of how the requester handles situations in which it is given more data than it can support, e.g., it only supports the mandatory elements but it is also supplied with optional elements;

g) Support for extensions – to ensure that the requester correctly responds to the different status codes that are returned due to its original usage of extensions.

8.2.2 Service Provider Perspective

For behavior that is defined in the UML-interface diagram of the specification the following test set must be defined to create the corresponding SOAP/HTTP messages:

a) Unsupported service report – the service request is rejected because it is not supported by the service provider (this is also true for requests requiring service extensions);

b) Successful service provision with only mandatory elements – the service is correctly supported when the supplied data contains only the mandatory data elements;

c) Successful service provision with mandatory and optional elements – the service is correctly supported when the supplied data contains any combination of optional data with all of the mandatory data (this includes all of the optional data, any single piece of optional data and combinations of multiple optional data);

d) Successful sequential service provision – the service is correctly supported when the same behavior is invoked several times. This is to ensure that the same multiple behaviors result in the appropriate incremental record changes;

e) Correct generation of error status codes – the service must correctly report the appropriate the error codes when invalid data is supplied, when the system is in a state inconsistent with the request and when there is a failure of some component in the system, e.g., full database in the service;

f) Support for extensions:-

- Confirmation that the service provider correctly rejects unknown service extensions
- Confirmation that the service can either report that data extensions are not supported but the service operation is otherwise completed
- Confirmation that the full extension facility is supported.

8.3 Conformance for Extension Profiles of the Base Profile

Conformance to profiles that have extended the base profile is defined in the associated profile description document.

Appendix A – Glossary of Terms

Throughout the General Web Services documents a variety of key terms, concepts and descriptions have been introduced. These terms, concepts and descriptions and defined below but where appropriate the normative definition from the IAF Glossary is referenced [AbsGloss, 03].

Abstract Framework	*See Abstract Framework Glossary.*
Application Services	*See Abstract Framework Glossary.*
Asynchronous Communications Messaging Model	This is a request/response message exchange where each phase is individually acknowledged. The Service Requester issues the request message and then waits for the acknowledgement from the Service Provider. The Service Requester is unblocked once the acknowledgement is received. At some later time the Service Provider will issue the response message. Once the response message been received by the Service Requester, a confirmation acknowledgement is returned to the Service Provider.
Authentication	*See Abstract Framework Glossary.*
Authorization	*See Abstract Framework Glossary.*
Business Process Execution Language for Web Services (BPEL4WS)	BPEL4WS (or BPEL) defines a notation for specifying business process behavior based on Web Services. This notation is called *Business Process Execution Language for Web Services*. Processes in BPEL4WS export and import functionality by using Web Service interfaces exclusively. Business processes can be described in two ways. *Executable* business processes model actual behavior of a participant in a business interaction. *Business protocols*, in contrast, use process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior. The process descriptions for business protocols are called *abstract processes*. BPEL4WS is intended to model the behavior of both executable and abstract processes. BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions. BPEL4WS defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces.
Common Services	*See Abstract Framework Glossary.*
IMS Binding Auto-generation Toolkit (I-BAT)	The I-BAT is the set of tools that are used to support the generation of bindings for specifications that have been developed using UML (both WSDSL and XSD bindings are supported). The I-BAT supports the usage of the IMS UML Profile and uses XSLs to convert the XMI of the UML description into the corresponding WSDL/XSD binding.
IMS GWS Base Profile	The General Web Service Base Profile provides a basic structure for the definition of Web Services. It consists of a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications that promote interoperability. The GWS Base Profile addresses the most common problems experienced when implementing web service specifications. The GWS Base Profile defines the selection of mechanisms within referenced specifications that are well understood, widely implemented and useful. The GWS Base Profile promotes interoperability for web service based specification implementations on different software and vendor platforms. The Base Profile focuses on a core set of web service specifications and the most common problems experienced implementing the identified web service specifications. The GWS Base Profile is based upon the WS-I Basic Profile v1.1 and the WS-I Simple SOAP Binding Profile v1.0.

IMS GWS Addressing Profile	This is the extension profile to *IMS GWS Base Profile* that defines how the services can be defined with an addressing system that is independent of the transport mechanism. This profile is based upon the *WS-Addressing* work from W3C.
IMS GWS Attachments Profile	This is the extension profile to *IMS GWS Base Profile* that defines how non XML-based data can be passed with the SOAP message, e.g., video file, image file, etc. This profile is based upon the *MTOM* work from W3C.
IMS GWS Security Profile	This is the extension profile to *IMS GWS Base Profile* that defines how security information is to be passed within the SOAP messages. It does not define a security architecture, instead it defines how the information in the SOAP messages is extended to support the security architecture used by an implementation. Therefore, the Security Profile can support many different security architectures. This profile is based upon the *WS-Security* work from W3C.
Message Transmission Optimization Mechanism (MTOM)	MTOM is one of the W3C message attachment approaches to enable SOAP messages to contain non-XML objects. MTOM is a development of SOAP with Attachments (SOAPwA) and is proposed as a replacement for the original SOAPwA specification.
Polled Communications Messaging Model	This is a request/response message exchange in which the response message is only returned when the Service Requester issues a poll message. The Service Requester issues the request message and then waits for the acknowledgement from the Service Provider. The Service Requester is unblocked once the acknowledgement is received. At a later time the Service Requester issues a poll message. The Service Provider may or may not reply with the response message. The Service Requester can issue any number of poll messages.
Representational State Transfer (REST)	The next generation of web services will likely adhere to an architectural style called Representational State Transfer (REST), the underlying architectural model of the current Web. REST is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.
SOAP	SOAP is a messaging protocol for XML documents which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment. It is a stateless, one-way message exchange mechanism, but applications can create more complex interaction patterns by combining such one-way exchanges with features provided by an underlying transport protocol and/or application-specific information.
SOAP with Attachments	SOAP With Attachments (SOAPwA) extends SOAPv1.1 by providing a MIME binding to support multiple message payloads, while ignoring the convention by which Remote Procedure Call (RPC) arguments may be marshalled and unmarshalled in XML. SOAPwA is especially suited for use cases where the two communicating parties are NOT located within the same organization, and the exchange paradigm is therefore more one of asynchronous Document Exchange across the Internet, than synchronous Remote Procedure Call within a single business (or University) enterprise.
Synchronous Communications Messaging Model	This is a simple request/response message exchange between the Service Requester and the Service Provider. The Service Requester issues the request message and then waits for the response message from the Service Provider. The Service Requester is blocked until the response message is received.

Web Service Architecture	The Web Service Architecture (WSA) is a normative document that seeks to provide a context and model for understanding Web services and to facilitate placing Web services specifications and technologies within a larger Web Services framework and with other technologies outside of WSA. The WSA's goal is to promote interoperability through the common definition of a web service and its core concepts and relationships. The purpose of WSA is not to specify how Web services are implemented or to impose restrictions on combination or coordination of Web services. WSA provides discussion of the WSA's core concepts and relationships from various perspectives.
Web Services Choreography Description Language	Web Services Choreography Description Language (WS-CDL) is an XML-based language that describes peer-to-peer collaborations of Web Services participants by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal. The Web Services specifications offer a communication bridge between the heterogeneous computational environments used to develop and host applications. The future of E-Business applications requires the ability to perform long-lived, peer-to-peer collaborations between the participating services, within or across the trusted domains of an organization. The Web Services Choreography specification is targeted for composing interoperable peer-to-peer collaborations between any type of Web Service participant regardless of the supporting platform or programming model used by the implementation of the hosting environment.
Web Services Description Language (WSDL)	A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types that are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding and a collection of ports defines a service.
WS-Addressing	WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.
WS-Attachments	WS-Attachments defines an abstract model for SOAP attachments and based on this model defines a mechanism for encapsulating a SOAP message and zero or more attachments. SOAP attachments are described using the notion of a compound document structure consisting of a primary SOAP message and zero or more related documents known as attachments.
WS-I Basic Profile	The WS-I Base Profile, created by the Web Services Interoperability Organization, consists of a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications that promote interoperability. This profile recommends the usage of XML Schema v1.0, HTTPv1.1, WSDLv1.1 and UDDI v2.0.
WS-Policy	Web Services Policy Framework (WS-Policy) allows the description of specific Web services meta-data (so-called policies) used for expressing the capabilities, requirements, and general characteristics of Web Services. The WS-Policy specification suite defines policies for simple general purposes like indicating languages and specifying general message preconditions, and more importantly an accompanying WS-Security Policy specification also defines policies for security requirements.

WS-Reliability	The purpose of WS-Reliability is to address reliable messaging requirements, which become critical, for example, when using Web Services in Business-to-Business applications. SOAP [SOAP1.1] over HTTP [RFC2616] is not sufficient when an application-level messaging protocol must also address reliability and security. While security is getting traction in the development of Web Services standards, reliability is not. This specification is intended as an initial proposal for defining reliability in the context of current Web Services standards. The specification borrows from previous work in messaging and transport protocols, e.g., SOAP, and the ebXML Message Service. It proposes appropriate modifications to apply this work to Web Services.
WS-Security	WS-Security defines a standard way to incorporate security information into a SOAP message using existing security standards for confidentiality, integrity, non-repudiation, authentication and authorization. WS-Security provides a method for representing security information in a SOAP message. WS-Security defines a way to pass security tokens, such as a simple username, SAML, X.509 certificates and Kerberos tickets, a mechanism using XML Signature to digitally sign all or part of a SOAP message, a mechanism using XML Encryption to encrypt part of a SOAP message and a method for attaching signature and encryption headers to a SOAP message.
WS-Security Minimalist Profile	The WS-Security Minimalist Profile defines a subset of the Web Services Security: SOAP Message Security (WS-Security) specification that constrains the core specification so that messages received by resource-limited platforms can be processed efficiently. The WS-Security specification describes a flexible method for securing SOAP Messages, providing message integrity and confidentiality, and exchanging security information through SOAP Messages. WS-Security can be used to support a wide variety of security models. WS-Security supports multiple security token formats, multiple trust domains, multiple signature formats, multiple encryption technologies and end-to-end message content security. WS-Security and WS-Security Minimalist Profile provide a framework and syntax to enable applications to exchange SOAP messages in a secure manner. The use and implementation of WS-Security or WS-Security Minimalist profile does not negate the need to ensure that the systems constructed are not vulnerable to attacks. IMS will make use of the WS-Security Minimalist Profile recommendations in the secure version of the GWS Base Profile.
WS-I Simple SOAP Binding Profile v1.0	The WS-I has separated out the messaging technology specific information in the WS-I Basic Profile documentation. The WS-I Simple SOAP Binding Profile describes the usage of SOAP messaging with the Basic Profile.
XML Schema	XML Schema Definition (XSD) is the primary XML binding control document format of IMS (at present these bindings are working to the May 2001 version of XML Schema) [XSD, 01]. The XSD defines elements, their content models, and attributes. It also defines the standard IMS vocabularies. The XSD defines the element types and attribute groups separately from the elements.

Appendix B – Messaging Questionnaire Template

The messaging questionnaire work aid is shown in Table B1[2]. Whenever possible, this questionnaire should be completed for each use-case. This set of responses is then analyzed to determine the set of bindings available to support the set of required features.

Table B1 Messaging questionnaire template.

		Infrastructure Questions - Applies to all transactions		
		Type of Transaction Set (fill out only one column; select only one of items A | B; select one each of A | B, C | D.)		
No.		**Request Data Update / ACK (or NAK)**	**Read Request / Response**	**Data Monitoring**
1	Event receivers pre-subscribed to	N/A	N/A	Source's Events | Source's Topic/Objects
2	If Topics/Objects, then must Source pre-register events on the Topics?	N/A	N/A	Y | N
3	Can the target system reject a request?	Y | N	N/A	N/A
4	Sync or Asynch	Sync | Async	Sync | Async	N/A
5	QOS - Duplicates	Target detects | Infrastructure. Prevents	Target detects | Infrastructure. Prevents	N/A
6	QOS - Connection	Target must be running | Infrastructure Guarantee	Target must be running | Infrastructure Guarantee	Recipients must be running | Infrastructure Guarantee
7	QOS - Must message delivery order be the same as posting order	N/A	N/A	Y | N
8	QOS - Notify Recipients when all messages are delivered?	N/A	N/A	Y | N
9	Target Identification	"Pre-wired" | Discovered	"Pre-wired" | Discovered	N/A
10	If multiple targets, how selected?	?	?	N/A
11	Scope of Record ID	Target & Partners | Internet	Target & Partners | Internet	N/A
12	Source need to synchronize with Target?	Y | N	Y | N	N/A
13	3rd party event logger	Y | N	Y | N	N/A
14	Encryption required	Y | N, direction X>Y | X<Y | X<>Y, type	Y | N, direction X>Y | X<Y | X<>Y, type	N/A
15	Source authenticated to Y	Y | N	Y | N	N/A
16	Source authorized to Y	Y | N	Y | N	N/A
17	Security provided by infrastructure	None | Encryption, Authentication, Authorization	None | Encryption, Authentication, Authorization	N/A
18	Stateful sessions	Y | N	Y | N	N/A
19	If stateful, then based on what?	Cookie | other??	Cookie | other??	N/A
20	If stateful, then how is basis of state id'd?	Dynamically Discovered | Configured	Dynamically Discovered | Configured	N/A
21	Optional elements in schema	Y | N	Y | N	Y | N
22	Optional element resolution strategy	?	?	?
23	Transport Requirements	HTTP, HTTPS, IIOP (Corba), IIOP DCOM, MOM - MQ/Series, MOM-MSMQ, MOM-JMQ, SMTP, Other?	HTTP, HTTPS, IIOP (Corba), IIOP DCOM, MOM - MQ/Series, MOM-MSMQ, MOM-JMQ, SMTP, Other?	N/A
24	Multiple object retrieval possible	N/A	Y | N	N/A
25	How flexible is query?	N/A	List | Full SQL	N/A
26	Can source report on multiple objects	N/A	N/A	Y | N
27	If yes, does message schema support aggregates	N/A	N/A	Enterprise Schema Does
28	Can Source send batch to Target	Y | N	N/A	N/A
29	If batch is yes, then does schema support aggregates	Enterprise Spec does	N/A	N/A
30	Can transaction/event be part of larger transaction?	Y | N	Y | N	Y | N
31	If so, how are transaction boundaries identified	?	?	?
32	If part of larger transaction, then is a 3rd system or more involved?	Y | N	Y | N	N/A
33	Version Number Context	Application | Object | Schema	Application | Object | Schema	Application | Object | Schema
34	Multiple Payloads	Y | N	Y | N	Y | N

2. The Messaging Questionnaire shown in Table B1 is available as an MS Excel workbook.

About This Document

Title	IMS General Web Services Base Profile
Editor	Colin Smythe (IMS)
Team Co-Leads	Cathy Schroeder (Microsoft), James Simon (SUN Microsystems)
Version	1.0
Version Date	19 December 2005
Status	**Final Specification**
Summary	This document presents the IMS General Web Services Base Profile (GWSBP). The Base Profile seeks to promote interoperability across web specification implementations on different software and vendor platforms. The Base Profile is focused on a core set of web service specifications and the most common problems experienced implementing the identified web service specifications. It is not a goal of the Base Profile to create a 'plug-and-play' architecture for web services or to guarantee complete interoperability. The Base Profile addresses interoperability at the application layer, in particular, the description of behaviors exposed via Web Services and assumes the interoperability within the infrastructure is sufficient and well-understood.
Revision Information	19 December 2005
Purpose	This document is circulated for public adoption. This document is to be adopted by IMS and all other organizations that wish to construct service-based interoperability specifications using Web Services.
Document Location	http://www.imsglobal.org/gws/gwsv1p0/imsgws_baseProfv1p0.html

To register any comments or questions about this specification please visit:
http://www.imsglobal.org/developers/ims/imsforum/categories.cfm?catid=20

List of Contributors

The following individuals contributed to the development of this document:

Name	Organization
Fred Beshears	UC Berkeley
John Evdemon	Microsoft Corp.
Ron Kleinman	SUN Micrsosystems Corp.
Sherman Mohler	Cisco Learning Institute, Inc.
Cathy Schroeder	Microsoft Corp.
James Simon	SUN Microsystems Corp.
Colin Smythe	Dunelm Services Ltd.
Scott Thorne	MIT

Revision History

Version No.	Release Date	Comments
Base Document v1.0	25 August 2003	The version of the Base Document submitted for voting to the IMS Technical Board.
Public Draft v1.0	31 January 2005	This is the first version of the General Web Services Base Profile released for public adoption. This document will remain in Public Draft form for approximately 12 months. This will allow the many specification and standardization activities in the field of Web Services to mature before final evaluation and adoption by IMS.
Final v1.0	19 December 2005	This is the first formal version of the Final Release.

Index