IMS Tools Interoperability Guidelines

Version 1.0

Date Issued: 28 February 2006

IPR and Distribution Notices

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

IMS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on IMS's procedures with respect to rights in IMS specifications can be found at the IMS Intellectual Property Rights web page: http://www.imsglobal.org/ipr/imsipr_policyFinal.pdf.

Copyright © 2006 IMS Global Learning Consortium. All Rights Reserved.

If you wish to copy or distribute this document, you must complete a valid Registered User license registration with IMS and receive an email from IMS granting the license to distribute the specification. To register, follow the instructions on the IMS website: http://www.imsglobal.org/specificationdownload.cfm.

This document may be copied and furnished to others by Registered Users who have registered on the IMS website provided that the above copyright notice and this paragraph are included on all such copies. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to IMS, except as needed for the purpose of developing IMS specifications, under the auspices of a chartered IMS project group.

Use of this specification to develop products or services is governed by the license with IMS found on the IMS website: http://www.imsglobal.org/license.html.

The limited permissions granted above are perpetual and will not be revoked by IMS or its successors or assigns.

THIS SPECIFICATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE CONSORTIUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS SPECIFICATION.

Copyright © 2006 by IMS Global Learning Consortium, Inc.

All Rights Reserved.

The IMS Logo is a registered trademark of IMS Global Learning Consortium, Inc.

Executive Summary

The IMS Tools Interoperability (TI) approach addresses the growing demand for a reusable mechanism for integrating third-party tools with core LMS (Learning Management System) platforms. Tools can add specialist functionality to the LMS such as assessment or discipline-specific teaching aids. The approach recommended greatly simplifies this task whilst also offering a Web Services solution equally applicable to Java and .Net implementers. The reuse of a commonly understood approach across tools will eliminate the need for bilateral solutions, thus focusing investment on adding real value to the learner experience.

Whilst working on this approach in the IMS Global Learning Consortium, the participants also implemented a demonstrator for alt-*i*-lab 2005. Their implementations made use of the Web Services Description Language (WSDL) auto-generation tool developed by the IMS General Web Services project group. The use of this tool has allowed the TI approach to be specified using the Unified Modelling Language (UML), from which the tool produces a WSDL file that can be used with a variety of Web Services development environments. The fact that there exist working systems that have been publicly demonstrated prior to the release of this document will hopefully instill confidence in the approach for other adopters and implementers.

The main architectural goal is to outline a framework that will allow Tools to easily integrate into (one or more) LMSs. This will enable the LMS to present the Tool side-by-side with its native learning tools. A client of the LMS will therefore be able to use the Tool for teaching and learning within a delivery context as it would any native tools within the LMS. In order to achieve this goal, our architecture introduces the following concepts:

- Proxy Tool – as its name indicates, a Proxy Tool is a proxy or a facade in the LMS, for its associated real Tool. The architecture defines a standard mechanism for packaging a Proxy Tool for deployment to an LMS;
- Tools Interoperability Runtime (TIR) – the TIR is a collection of services implemented by any container (the LMS and Tool containers in this case) that will allow Proxy Tools to be deployed, configured, and launched from within that container. Thus, the TIR will include distinct services for managing the deployment, configuration, and launch of Proxy Tools from within or into the host environment, as well as services for receiving an outcome for the interaction.

Additionally, the architecture defines a core protocol for the interaction between the TIR/Proxy Tool and the Tool. The protocol utilizes a Service-oriented Architecture/Web Services paradigm based upon the IMS General Web Services (GWS) specification v1.0 that will:

- Facilitate a loose coupling between the TIR/Proxy Tool and the Tool;
- Allow for a layering of additional mechanisms, e.g., security profiles, outcome profiles, to the core protocol;
- Utilize XML as the base language along with WSDL for the services definition and SOAP for the base transport protocol.

This version of the Tools Interoperability framework will not provide a concrete architecture for integrating a Tool's user interface with that of the LMS. User authentication is handled by the LMS in each case, whilst the LMS authenticates itself to the Tool using a shared secret. The approach exploits a modular context profiling mechanism to pass additional information to the Tool:

- The LMS can include the user's IMS Accessibility Learner Information Package (ACCLIP) profile allowing the Tool to self-configure its user interface to the learner's precise needs.
- The optional Outcome profile states the results format required – currently a simple score, but could be HR-XML, IMS Question & Test Interoperability (QTI) Results, etc.

The approach has been designed to allow additional context profiles to be added in the future without impacting currently supported interoperability.

Table of Contents

1. Introduction

1.1 Overview

Many observers of the use of technology in teaching and learning believe that Learning Management Systems (LMSs) have been effective in supporting some aspects of the process, but much of the transformative process of technology in learning remains unfilled. One promising area is the development of specialized tools that extend LMS capabilities. In particular, tools that focus on specialized kinds of assessment, discipline-specific pedagogy, e.g., math tutoring, or pioneer new capabilities hold great promise. The current challenge is that innovative tools are often tightly coupled to one LMS. Faculty at other institutions find it almost impossible to use the innovations of their colleagues unless all are on the same LMS. No single company, project, discipline, or university can keep pace with the innovative potential of improved teaching and learning tools.

Therefore, the IMS Tools Interoperability (TI) Guidelines addresses the growing demand for a reusable mechanism for integrating third-party tools with core LMS platforms. The outcomes of this work offer hope of connecting a large number of innovative tools to a larger audience. The TI Guidelines start to remove the technical obstacles to provide greater tool mobility to the students and faculty who wish to use them.

While the initial scope of work is modest, it is anticipated that there will be opportunity for further projects that will build upon this initial work, each targeting more complicated connections, e.g., calendar, grade book, data exchange, etc., according to how these are prioritized.

1.2 Scope and Context

1.2.1 Approach

- The interoperating applications, the LMS and the Tool, will each host a Tools Interoperability Runtime (TIR) that will facilitate the hosting of, and access to, an external system deploying/running the Tool from within the LMS. The LMS TIR will enable the LMS to host and access external system applications via its Proxy Tool;
- A Tool will provide a proxy (Proxy Tool) via a deployment descriptor that will be hosted by the LMS/TIR;
- A Proxy Tool will be configured and managed locally via the LMS host. The LMS TIR will facilitate the launch, presentation and interactions supported at run-time between the LMS host and the Proxy Tool application. Proxy Tool applications can therefore be deployed “as is” without making extensive changes to core functionality.

1.2.2 Technology

- The guidelines are geared towards web applications and utilize Web Services standards. This implies that HTTP, SOAP, XML, WSDL, WS-Security are the key enabling technologies. The Web Service implementation is based upon the IMS General Web Services (GWS) [GWS, 05a], [GWS, 05b];
- The Proxy Tool is entirely a descriptor-based package, i.e., sans code. Thus, there is no particular requirement on any specific programming language;
- All launch and run-time interaction between the Proxy Tool and its host LMS application is managed by their respective TIR implementations that are based on Web Services and XML messages. The Web Services interfaces are designed to be flexible and easily extensible as the guideline evolves.

1.2.3 User Interface

- There is no “integration” at the user interface level, i.e., the Tool is expected to display its own user interface to the user, when the user chooses to use it (by navigating to the Proxy Tool in the hosting LMS/TIR);
- The Proxy Tool user interface can be presented via a separate browser window or embedded in the frameset of the host LMS application.

1.2.4 Security

- We defer to the IMS GWS best practices around WS-Security [GWS, 05c]. However, we do provide an optional and extensible security profile that utilizes a simple shared-secret approach as an initial approach that could be utilized by the interoperating applications to authenticate each other.

1.2.5 Deployment

- The guidelines recommend a simple deployment procedure, whereby an XML based deployment descriptor is all that is required to deploy a Proxy Tool to an LMS/TIR;
- Hot deployment of Proxy Tool contributions into the LMS TIR will be an optional (but highly recommended) feature.

1.2.6 Persistence

- The guidelines do not mandate a persistence model. All data and storage are private and localized for the interoperating applications. The LMS is not required to persist any data on behalf of the Proxy Tool/Tool service, other than the deployment descriptor, the resulting configuration per delivery context and outcome data (optionally) sent back by the Tool.

1.2.7 Logging

- The guidelines do not mandate a formalized error handling/logging model. It does provide recommendations for how common error scenarios should be handled and logged, but leaves the specifics up to the application and its logging framework.

1.3 Structure of this Document

The structure of this document is:

2. Use Cases and Requirements	Documents the use cases that have been addressed in developing the Tools Interoperability Framework and defines the resulting set of requirements;
3. Architecture and Approach	Describes the logical architecture, components, and services that comprise the Tools Interoperability Framework;
4. Tools Interoperability Model	Describes the logical model and the UML used to generate the Tools Interoperability XSDs;
5. Implementation Guidelines and Best Practice	Lists the best practice guidance for implementers, known issues with the current version of the WSDL, and topics identified for further development of the Tools Interoperability Framework;
6. Sample Implementations	Describes the test harnesses for testing Tools Interoperability web service implementations for LMS platforms and tools;
Appendix A Support Files	URIs for the corresponding WSDL and XD files, and the test harness software.

1.4 References

[AbsASCs, 03]	*IMS Abstract Framework: Applications, Services & Components 1.0,* C. Smythe, IMS/GLC, July 2003.
[AbsGloss, 03]	*IMS Abstract Framework: Glossary 1.0,* C. Smythe, IMS/GLC, July 2003.
[AbsWhite, 03]	*IMS Abstract Framework: Abstract Framework Whitepaper 1.0,* C. Smythe, IMS/GLC, July 2003.

[ACCLIP, 03] *IMS Learner Information Package: Accessibility for LIP 1.0.* Eds. M. Norton, J. Treviranus, IMS/GLC, June 2003.

[APG, 05a] *IMS Application Profile Guidelines Whitepaper: Part 1 Management Overview 1.1,* Eds. K. Riley, P. Hope, IMS/GLC, September 2005.

[APG, 05b] *IMS Application Profile Guidelines Whitepaper: Part 2 Technical Manual 1.1,* Eds. K. Riley, P. Hope, IMS/GLC, September 2005.

[GWS, 05a] *IMS General Web Services Base Profile v1.0,* C.Schroeder, J.Simon and C.Smythe, IMS/GLC, December 2005.

[GWS, 05b] *IMS Binding Auto-generation Tool-kit Manual v1.0,* C.Smythe, IMS/GLC, December 2005.

[GWS, 05c] *IMS General Web Services Security Profile v1.0,* C.Schroeder, J.Simon and C.Smythe, IMS/GLC, December 2005.

[LIP, 01] *IMS Learner Information Packaging 1.0,* Eds. C. Smythe, F. Tansey, R. Robson, IMS/GLC, March 2001.

[LIP, 05] *IMS Learner Information Packaging 1.0.1,* Ed. C. Smythe, IMS/GLC, January 2005.

[WSI, 04] *Web Services Interoperability Basic Profile Version 1.1,* Eds. K. Ballinger, D. Ehnebuske, C. Ferris, M. Gudgin, C.K. Liu, M. Nottingham, P. Yendurli, Web Services – Interoperability Organization, June 2004.

[WSS, 04] *Web Services Security: SOAP Message Security 1.0,* Eds. A. Nadalin, C. Kaler, P. Hallam-Baker, R. Monzillo, OASIS, March 2004.

1.5 Definitions

Authentication Component	A component that performs authentication functions, conforming to the corresponding component interface.
Component	A deployment unit that performs a specific function/s.
Component Interface	The interface for a component.
Configuration Settings	A set of properties/meta-data for a Proxy Tool specific to a LMS deployment.
CRUD	Create, Read, Update, & Delete.
Deployment Descriptor	A mechanism to describe the contents of deployable components. The application into which the component is deployed uses the descriptor to configure components to their environment.
Frame, iFrame	HTML display elements.
LMS	Learning Management System.
Operational Meta-data	Meta-data describing system components, e.g., LMS, Tool, and used to configure installation or launch settings (cf., IMS Meta-data used to describe learning resources).
Presentation Context	Context in which Proxy Tool is configured/capable of rendering its presentation/UI.
Proxy Tool	An entity that functions as a proxy for a tool within an LMS runtime.
Reference Context	Context in which Proxy Tool is instanced in the host LMS, i.e., course LO, toolbar/menu etc.
Runtime Container	The process on a computer executing the LMS or Tool. This can be a single process or a collective, e.g., clustered deployment.

Session	A mechanism to maintain state information for a user during an interaction with a system/Tool.
Single Sign-On (SSO)	The mechanism by which an entity is authenticated once in order to access multiple services/systems.
Tool	An entity that provides a specific learning enhancement and/or outcome.
Tools Interoperability Framework (TIF)	The underlying structure that aggregates all elements of the Tools Interoperability Guidelines.
Tools Interoperability Runtime (TIR)	The runtime environment provided by host systems to Proxy Tool deployable components.
Tool Meta-data	All inclusive term that includes individual aggregate subsections of meta-data associated specifically with deployment descriptor, configuration settings, etc.
User Role(s)	Instructor, Designer, Student, Administrator.
Web Service (Interface)	In our context, a software application/sub-system that provides learning management services. The interfaces and bindings are defined using XML and interactions with these services are likely done using XML-based messages.

2. Use Cases and Requirements

2.1 Use Cases

2.1.1 Affected Roles and Definitions

Role	Definition
LMS Administrator	The role responsible for controlling the administration of the LMS and its range of related tools.
LMS Instructor	The individual responsible for designing the course content and for deciding which tools are required to use that content.
LMS Learner/Student	The individual that is using the Proxy Tool to interact with the content controlled by the LMS.

2.1.2 High-level Use Case Scope

Figure 2.1 illustrates the use cases considered for the framework.

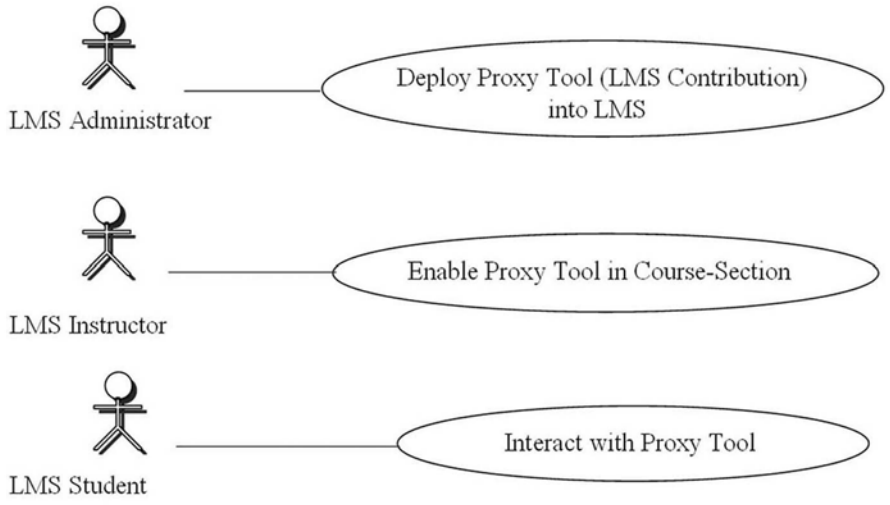

Figure 2.1 Use Case Scope.

2.1.3 Administrator Deployment of Proxy Tool

Use Case 1	Administrator Deployment of Proxy Tool
Level	Summary
Primary Actor(s)	LMS Administrator
Secondary Actor(s)	Tool developer, LMS
Trigger	Administrator needs to enable access to a specific Tool
Preconditions	• TIR installed on LMS • LMS configured to define Proxy Tools
Success Post-condition	• Proxy Tool definition created on the LMS

Use Case 1	Administrator Deployment of Proxy Tool
Failure Post-condition	• Proxy Tool definition not created on the LMS
Main Success Scenario	1) Administrator obtains deployment descriptor in package. 2) Administrator runs LMS-defined tool to process deployment descriptor. 3) LMS processes deployment descriptor to create LMS-defined data structures to create a Proxy Tool.
Variations	(a) Spec defines (and LMS supports) integration with existing delivery mechanisms, such as identifying Tool Proxies and Proxy Instances in an IMS Content Packaging descriptor. 1.a. Deployment descriptor is provided implicitly as part of a content package. 2.a. LMS package processing tool runs LMS-defined tool or takes other LMS-defined action, e.g., put installation event in a queue for subsequent administrator action.
Exception Conditions	• Tool already defined on the LMS • Incorrect TI guideline version • Unsupported interface • Unsupported authentication modes • Unreadable deployment descriptor

2.1.4 Instructor Configuration of Tool Invocation

Use Case 2	Instructor Configuration of Tool Invocation
Level	Summary
Primary Actor(s)	Instructor (Instructional Designer)
Secondary Actor(s)	LMS, Tool
Trigger	Instructor wishes to include a resource or activity available in a tool external to the LMS within some LMS-defined Deployment Context.
Preconditions	• Proxy Tool defined on the LMS • LMS configured to allow Proxy Tool instances in the current Deployment Context • Proxy tool supports current Deployment Context
Success Post-conditions	• Proxy Tool instance defined in the specified Deployment Context
Failure Post-conditions	• Proxy Tool instance not-defined in the specified Deployment Context • Deployment Context is not altered
Main Success Scenario	1) Instructor navigates to a Deployment Context in which she or he wishes to place a reference to a Proxy Tool. 2) Instructor activates LMS-defined tool to create Proxy Tool Instance. 3) Instructor enters appropriate meta-data for accessing a resource resident in the Tool. 4) LMS creates appropriate data structure for subsequent launch.

Use Case 2	Instructor Configuration of Tool Invocation
Variations	(a) Spec defines and tool supports runtime negotiation of launch parameters, possibly including transfer of control launch sequence in which Instructor interacts with tool for one or more Tool-hosted transactions, and the tool posts the results back to LMS. 2.a.1 LMS invokes the Tool to negotiate a launch URL for the specified deployment mode. 2.a.2 Tool verifies authentication and authorization, based on assertions in the tool invocation. Authentication workflow is dependent upon the mode of authentication used. 3.a.1 The tool provides a facility for authoring a resource or selecting an existing resource. 3.a.2 The tool calls back to the TIR Deployment Context service to register meta-data appropriate for the subsequent launch.
Exception Conditions	• Proxy Tool does not support specified Deployment Context • Deployment Context does not allow Proxy Tool Instances • Proxy Tool cannot call the TIR Deployment Context service

2.1.5 Instructor Testing of Configured Invocation

Use Case 3	Instructor Test of Deployment
Level	Summary
Primary Actor(s)	Instructor (LMS Learner)
Secondary Actor(s)	LMS, Tool
Trigger	Instructor has deployed a resource or Learning Activity and wishes to evaluate the presentation in the launch context of a learner.
Preconditions	• Tool Instance is deployed • LMS allows tool pseudo-student launch (what constitutes a pseudo-student has been agreed using some out-of-band mechanism).
Success Post-conditions	• Activity launched and Instructor directed to Tool Activity (via HTTP redirection, frame generation, or new window creation). Both LMS and tool honor pseudo-student and do not log results, etc. Note system logs would still run.
Failure Post-conditions	• LMS-defined error state and communication to user
Main Success Flow	1) Instructor visits a Deployment Context containing a Proxy Tool instance. 2) Instructor activates Proxy Tool instance. LMS implementation implicitly (role detection) or explicitly (special icon/action) determines whether the tool launch is for Instructor/Learner pseudo-context. 3) TIR reads Instance Deployment Descriptor. 4) TIR contacts tool's Launch Service, passing Context-dependent arguments, including arguments to indicate pseudo-context. 5) Tool returns URL for display to Instructor. 6) TIR renders URL (via an iFrame, standard frame, HTTP redirect, or new Window). 7) Instructor initiates Learning Activity in tool. 8) Tool authenticates any security assertions passed as part of the initiation (refer to Learner Invocation of tool (Use Case 4) for information on authentication variations). 9) Instructor completes Learning Activity in tool; the tool does not record any results.
Variations	Authentication variations – refer to Use Case 'Learner Invocation of Tool'.

Use Case 3	Instructor Test of Deployment
Exception Conditions	• Launch service cannot be resolved or contacted from LMS • Security assertion fails • Tool failure (general) • LMS Response Service cannot be resolved or is otherwise unavailable from the tool • Unsupported context (Instructor as pseudo-student)

2.1.6 Learner Invocation of Tool

Use Case 4	Learner Invocation of Tool
Level	Summary
Primary Actor(s)	Student (LMS Learner)
Secondary Actor(s)	LMS, Tool
Trigger	Student needs to access a resource or complete a Learning Activity that is (1) resident on an external tool and (2) made available as a Proxy Tool instance in the host LMS.
Preconditions	• Tool Instance defined in a deployment context available to the Student • LMS allows tool launch
Success Post-conditions	• Learner activity launched and user directed to Tool Activity (via HTTP redirection, frame generation, iFrame generation, or new window creation).
Failure Post-conditions	• LMS-defined error state and communication to user
Main Success Flow	1) Student visits a Deployment Context containing a Proxy Tool instance. 2) Student initiates a navigation event triggering tool launch, e.g., clicks on a link rendered by the LMS. 3) TIR reads Instance Deployment Descriptor. 4) TIR contacts Tool's Launch Service, passing Context-dependent arguments. 5) Tool returns URL for display to Student. 6) TIR renders URL (via an iFrame, standard frame, HTTP redirect, or new Window). 7) Student initiates Learning Activity in Tool. 8) Tool authenticates any security assertions passed as part of the initiation. 9) Student completes Learning Activity in Tool.
Variations	(a) Proxy Instance Supports (or spec defines) passing credentials directly to Tool (e.g., identity assertion or credential forwarding) instead of SSO implementation (e.g., tool verifies identity independently). 8.a. Security assertion happens as part of step 4. See Authentication examples, above. (b) Tool requires data retrieved from LMS to process and/or render resource or activity. 8.b. Tool queries LMS TIR to retrieve any relevant data required to render Learning Activity (e.g., course data, rostering, etc.). 9.b. Tool calls post-back service in LMS TIR to indicate completion and/or report results.
Exception Conditions	• Launch service cannot be resolved or contacted from LMS • Security assertion fails • Tool failure (general) • LMS Response Service cannot be resolved or is otherwise unavailable from the tool

2.2 Requirements

The requirements of TI from the perspective of an LMS are to:

a) Be capable of responding to user interaction to access a Tool, by sending the Tool a launch instruction;

b) Launch instruction may include initialization information such as context and user information;

c) Launch instruction may provide authentication credentials or an authentication assertion;

d) Provide a Tool with information about how to interrogate the LMS to find out information;

e) Be interrogated by a Tool to provide additional contextual information such as user information;

f) Receive information such as results from a Tool;

g) Send an instruction to a Tool to terminate the current session and revoke the authentication credentials provided;

h) Receive a termination notification from a Tool.

The requirements of TI from the perspective of a Tool are to be:

a) Hosted independently of the LMS;

b) Capable of receiving a launch instruction from an LMS;

c) Capable of processing authentication information provided by the LMS as part of the launch instruction;

d) Capable of creating and managing a user session after receiving a launch instruction from the LMS;

e) Capable of providing an LMS with a URL for accessing its web-based user interface;

f) Capable of interrogating an LMS for additional contextual information, such as user information;

g) Able to send information to the LMS such as results;

h) Able to destroy or revoke a user session after receiving a terminate instruction from the LMS;

i) Able to notify the LMS that it has terminated a session.

The following are out-of-scope for this version of TI:

- LMS and Tools that are not browser-based;
- Accessibility issues of either the LMS or the Tool;
- Security of the LMS – Tool connection;
- A more comprehensive tools interoperability definition, e.g., Multi-learner Tools, workflow enabled Tools.

3. Architecture and Approach

3.1 Overview

This section provides an overview of the proposed architecture of the Tools Interoperability framework. As a first step, we outline the problem domain to which this architecture applies. The domain under consideration in this case, has two primary entities – the LMS and the Learning Tool (Tool). The LMS is a platform that provides a core set of teaching and learning services and tools for the academic enterprise. The Tool typically provides specialized functionality to aid in teaching and learning. It is assumed that the architecture and approach here described is generally appropriate in the circumstances where the LMS and the Tool have separate runtime containers, i.e. they do not share a deployment context.

The main architectural goal therefore is to outline a framework that will allow Tools to easily integrate into (one or more) LMSs. This will enable the LMS to present the Tool side-by-side with its native learning tools. A client of the LMS will therefore be able to use the Tool for teaching and learning within a delivery context as it would any native tools within the LMS. In order to achieve this goal, our architecture introduces the following concepts, as depicted in Figure 3.1.

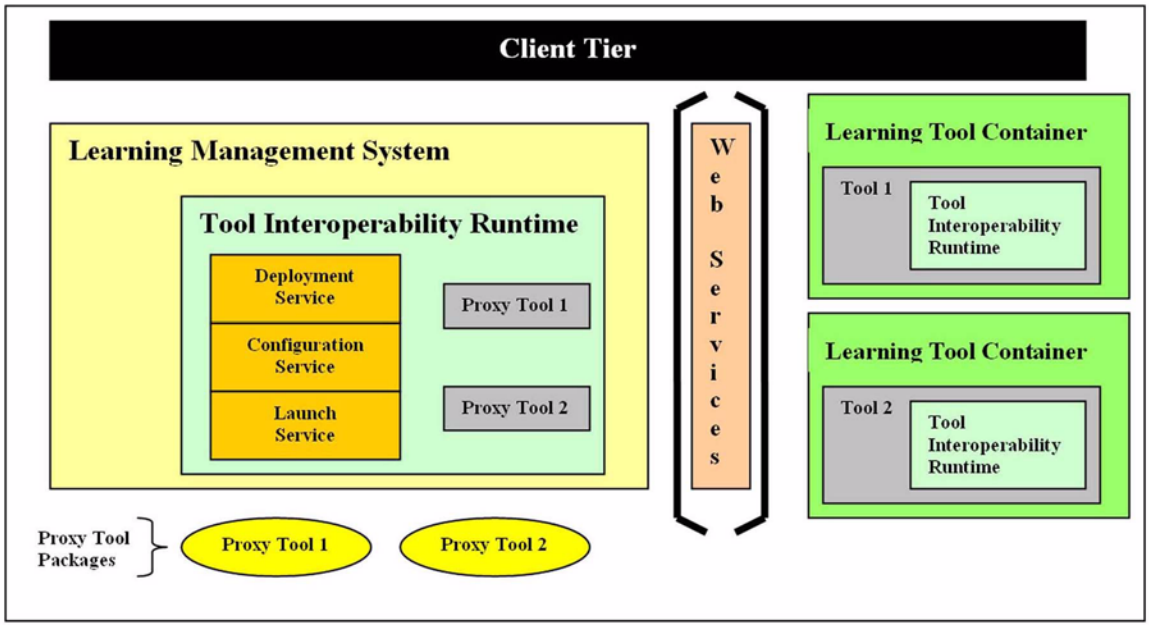

Figure 3.1 Tools Interoperability Logical Architecture.

- Proxy Tool – as its name indicates, a Proxy Tool is a proxy or a facade in the LMS, for its associated real Tool. The architecture defines a standard mechanism for packaging a Proxy Tool for deployment to an LMS;
- Tools Interoperability Runtime (TIR) – the TIR is a collection of services implemented by any container (the LMS and Tool containers in this case) that will allow Proxy Tools to be deployed, configured and launched from within that container. Thus, the TIR will include distinct services for managing the deployment, configuration, launch of Proxy Tools from within or into the host environment as well as services for receiving an outcome for the interaction.

Additionally, the architecture defines a core protocol for the interaction between the TIR/Proxy Tool and the Tool. The protocol will utilize a Service-oriented Architecture (SOA)/Web Services paradigm, i.e. will:

- Facilitate a loose coupling between the TIR/Proxy Tool and the Tool;
- Allow for a layering of additional mechanisms, e.g., security profiles, outcome profiles, to the core protocol;

- Utilize XML as the base language along with WSDL for the services definition and SOAP for the base transport protocol.

This version of the Tools Interoperability framework will not provide a concrete architecture for integrating a Tool's user interface with that of the LMS.

The following sections provide details of the various components of the Tools Interoperability framework.

3.2 Logical Components

3.2.1 Proxy Tool

A Proxy Tool is a facade for an instance of a corresponding Tool's runtime presentation, business logic, and persistence services. As mentioned earlier, the Tool itself is expected to be hosted external to the LMS environment.

A key design goal and assumption is for the Proxy Tool to require no specialized code, but instead have all the deployment, configuration, and runtime context encoded in its deployment package. Thus, for this version of the TI framework[1], the Proxy Tool is entirely a descriptor-based package, i.e. sans code. The following represent the various typical stages in the deployment and use of Proxy Tools (the specific details depend on the actual tools):

- Tool Developer/Supplier creates Proxy Tool deployment package;
- LMS Administrator deploys Proxy Tool deployment package;
- LMS TIRs deployment service loads the Proxy Tool, thus creating a Proxy Tool *Definition* within the LMS (during which validation of the deployment profile occurs, including a validation of profiles like security, outcome, user and delivery context asserted by the Proxy Tool deployment descriptor. The LMS can choose to not deploy the Proxy Tool if it does not support the required profiles);
- LMS Administrator configures the Proxy Tool for use within an Institution by updating its definition appropriately with LMS specific data;
- An instructor or course designer utilizes the Proxy Tool definition to create a Proxy Tool *Instance* within a delivery context (course). The instance inherits the full base configuration and is additionally customized by the instructor/designer for launch from within the specific delivery context;
- A learner in the course of a learning session is presented with the Proxy Tool instance (as a Learning Object or as part of one) and subsequently launches the Proxy Tool by selecting the URL provided by the LMS's UI;
- Tool's TIR validates and accepts launch in collaboration with the LMS's TIR directs the user to the Tool's UI, e.g., a redirect to a Tool specific URL;
- Learner uses the Tool and in doing so potentially generates an outcome;
- Tools's TIR sends the outcome to the LMS's TIR at the end of a learner's interaction with the Tool.

3.2.2 Proxy Tool Deployment Package

The deployment package is an archive containing a manifest and the Proxy Tool's deployment descriptor. At this time, no additional requirements are enforced on the Proxy Tool package. A specific LMS's TIR can choose to provide value-adds, such as support for digitally signed archives for Proxy Tools.

3.2.3 Proxy Tool Deployment Descriptor

The deployment descriptor is an XML document. The deployment descriptor specifies one or more **deployment profiles** for a Proxy Tool. Each deployment profile contains the following:

1. It can be envisioned in a future version, that the TIR can expose the deployment and configuration services as web services as well, in addition to the launch web service. This can allow a Tool's TIR to discover any other TIRs in its deployment environment, e.g., using a discovery mechanism like UDDI. Once discovered, an administrator can request a proxy for the Tool to be deployed into one or many of these other TIRs. This would eliminate the need for a deployment package for the Proxy Tool. This of course presumes that some mechanism for establishing trust among these TIRs is in place as well.

- **Core Settings**: These include core meta-data for the tool including a fully qualified name, description, version, source provider, etc. Additionally, it will specify addressing or locator specification for the host TIR and for the Tool and the security mechanism that is to be used during the launch of the Proxy Tool;
- **Contextual Settings**: These include information specific to the launch/delivery context. The Proxy Tool will assert requiring either or all of a specific type of user profile and delivery context profile during the launch;
- **OutcomeProfile**: The Proxy Tool will assert requiring the deploying TIR to support a specific type of outcome profile. The Tool will report an outcome conforming to this outcome profile;
- **SecurityProfile**: The Proxy Tool will assert requiring the deploying TIR to support a specific type of security profile. This profile will be used in the SOAP header for the launch request and outcome messages;
- **Tool Settings**: These include an arbitrary number of tool specific settings. These settings are managed by the TIR but are meaningful only to the Tool and as such expected to be passed to the Tool via the Proxy Tool core protocol during launch. A typical use of these settings is for the Proxy Tool to identify some resource on the tool as being the target of a particular launch e.g. display the resource specific to a given concept during the launch of an instance of the ConceptTutor Proxy Tool.

3.2.4 Tools Interoperability Runtime (TIR)

The TIR is a collection of services implemented by any container that will allow the container to participate in the Tools Interoperability Framework. The TIR services include:

- **Deployment Service:** The main function of this service is to interpret and load the Proxy Tool definition into the host TIR via its deployment descriptor. Thus this service is also expected to perform validation of the Proxy Tool settings in order to ensure correctness of and compatibility with the LMS's TIR;
- **Configuration Service:** The main function of this service is to manage the runtime settings of the Proxy Tool in order to provide the proper set of the same for/during any given launch context;
- **Launch Service:** This service provides two main services, depending upon the context:
  - **Proxy Tool Host:** Performs all the functions related to launch of a Proxy Tool, including generating the relevant Proxy Tool launch message, utilizing the appropriate security profile, etc.
  - **Tool:** Exposed as a web service that accepts launch messages from the LMS TIR, understands the security profile used therein and responds back to the LMS TIR using the Proxy Tool core protocol as to the status of the launch.
- **OutcomeService:** This service provides two main services, depending upon the context:
  - **Tool:** A web service client that generates outcome messages from the Tool's TIR conforming to a specific outcome profile type, for a given interaction of a user with the Proxy Tool/Tool, including utilizing the appropriate security profile;
  - **Proxy Tool Host:** Exposed as a web service that accepts outcome messages from the Tool TIR, understands the security profile used therein and responds back to the Tool TIR using the Proxy Tool core protocol as to the status of the outcome processing.

The TIR can be implemented by any container that wishes to participate in the Tools Interoperability Framework. The degree to which a TIR implements the various services (deployment, configuration, launch, outcome), dictate its level of participation in the interoperability framework. Thus, in our problem domain, the minimal TIR services that need to be implemented by the LMS and Tool are as follows:

- **LMS**: Deployment Service, Configuration Service, Launch Service;
- **Tool**: Launch Service.

The following sections provide additional details on various components of the TIR.

3.2.5 Tools Interoperability Runtime : Deployment Service

The TIR Deployment Service interprets and deploys Proxy Tools via the deployment descriptors so as to deploy as a runtime component within the host LMS environment and tool contexts. The following deployment scenarios should be supported by a TIR's deployment service:

- **Explicit:** A system or learning context administrator privileged role (or equivalent per LMS) explicitly deploys a Proxy Tool package, configures editable settings and enables the Proxy Tool. An instructor-privileged role subsequently uses an instance of the Proxy Tool within a delivery context;
- **Implicit:** The Proxy Tool deployment is referentially triggered as an element of a learning module or content package. Note that the reference within a learning module or content package would be to an instance of a Proxy Tool. The Proxy Tool definition itself would have to pre-exist, likely via a prior explicit deployment.

3.2.6 Tools Interoperability Runtime : Configuration Service

The host TIR configuration service facilitates the management of Proxy Tool settings, which in turn come into the TIR via the Proxy Tool deployment descriptor.

At minimum, the host TIR configuration services must provide a service level which enables all Proxy Tool settings to be opaquely loaded, persisted and ultimately made available to the Proxy Tool instance when launched and ultimately when in its execution context. Optionally, the LMS can provide a more extensive capability to system or learning context administrator privileged roles to leverage specialized presentation capabilities to add/modify such configuration elements via an application UI interaction.

3.2.7 Tools Interoperability Runtime : Launch Service

As outlined previously, the TIR launch service provides two main functions, depending upon the following contexts:

- **Proxy Tool Host:** Performs all the functions related to launch of a Proxy Tool, including generating the relevant Proxy Tool core protocol, delegation to the appropriate security mechanism, etc.
- **Tool:** Exposed as a web service that accepts launch messages from the LMS TIR, understands the security mechanism/token used therein and responds back with to the LMS TIR using the Proxy Tool core protocol.

3.2.8 Tools Interoperability Runtime : Outcome Service

As outlined previously, the TIR outcome service provides two main functions, depending upon the following contexts:

- **Tool:** A web service client that generates outcome messages from the Tool's TIR conforming to a specific outcome profile type, for a given interaction of a user with the Proxy Tool/Tool, including utilizing the appropriate security profile;
- **Proxy Tool Host:** Exposed as a web service that accepts outcome messages from the Tool TIR, understands the security profile used therein and responds back to the Tool TIR using the Proxy Tool core protocol as to the status of the outcome processing.

3.2.9 Tools Interoperability Runtime : Security Management

To provide the most flexibility for support of multiple forms of authentication and authorization models and capabilities offered via many LMS environments, and their associated broader enterprise deployment environments, the Proxy Tool deployment descriptor will assert a security profile.

From a practical, implementation standpoint, the various SOAP messages that form the Proxy Tool core protocol will utilize a SOAP header extension that will conform to the specified security profile. For example, using the SharedSecretSecurityProfile (a simple type of security profile specified by the Tools Interoperability base schema), a header containing a shared secret token will be inserted into the launch and outcome messages. The shared secret is configured in both interacting TIRs in advance.

3.2.10 Tools Interoperability Runtime : Session Management

Session management within the context of the Proxy Tool instance is purely an artifact/by-product of the interaction between the launch services on both the TIRs. Thus, during the launch, when the LMS TIR issues a launch request, the Tool TIR upon processing the message will typically perform any session setup necessary, prior to responding back to the LMS TIR with a launch directive, e.g., redirect the users browser to this URL.

All session timeout, implicit/explicit termination etc. is governed by the containers hosting the interacting TIRs and underlying implementation of session management which can accommodate (or not) the Proxy Tool instance sessions as appropriate to satisfy any special runtime requirements if any.

3.2.11 Logical Component Interaction

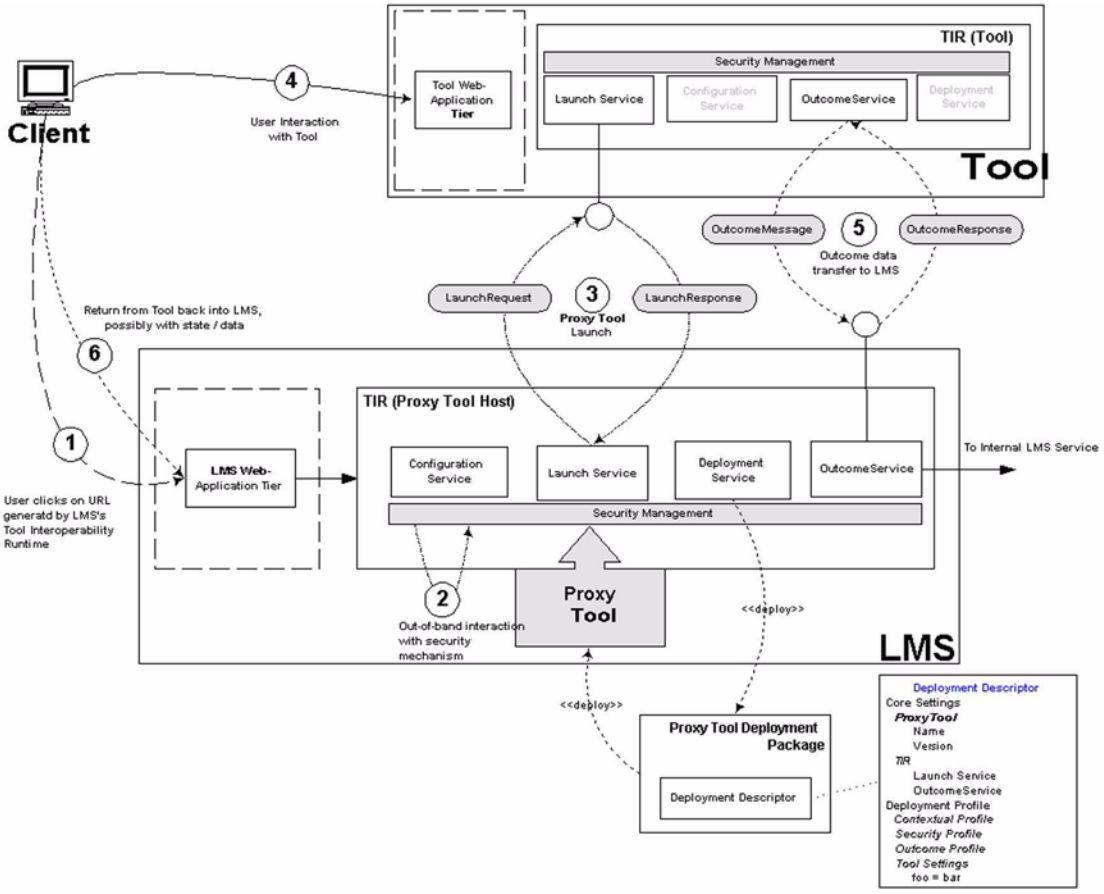

Figure 3.2 Tools Interoperability Interaction Diagram.

The individual steps in the interaction are as described below:

1) The end-user selects a link for the URL for the Proxy Tool generated by the LMS, in cooperation with its TIR, in the course of a teaching or learning session in the LMS. It is assumed that at some point in the past, the Proxy Tool has been deployed by an administrator and added to a delivery context by an instructor. The request is handled by the LMS's web-application tier and subsequently delegated to the LMS's TIR. The TIR delegates the request to its launch service, which collaborates with its configuration service to generate the appropriate settings for the launch message to be sent to the Proxy Tool;

2) The launch service then delegates to the security mechanism asserted by the Proxy Tool configuration. The launch message is passed along to the security mechanism[2], which performs any required authentication and subsequently inserts a security header into the launch message. This token could refer to an inline credential (e.g., shared secret) or to an external credential, e.g., cookie, that the Tools' TIR can use to authenticate the incoming launch message;

2. In a simplified approach, e.g., a shard secret being the only security mechanism asserted by the Proxy Tool settings, it is entirely possible that the LMS's TIR itself inserts the security token into the launch message as it will be available as part of the Proxy Tool settings. It is also possible to conceive Proxy Tool settings asserting other or multiple security mechanisms, in which case the security mechanism/s could act as intermediary endpoint/s for the message being passed from the LMS's TIR to the Tool's TIR.

3) The launch service, in collaboration with the security mechanism asserted, thus initiates a launch of the Proxy Tool in the host LMS, by sending a launch message to the Tool TIR's launch service. The Tool TIR's launch service delegates to the security mechanism asserted in the launch message to validate the inbound launch request and upon successful validation the Tool will perform any session setup necessary and the launch response message is sent back to the LMS's TIR. This message will indicate at the very minimum the status, i.e. success/failure, of the launch and if successful, the launch URL;

4) As a success result of (3), the user is now presented with the Tool's user interface, either in a new browser window or inline. The user interacts with the Tool and utilizes its specialized teaching/learning functionality. There is no interaction back from the Tool to the LMS TIR at this stage;

5) When the user completes a logical unit of work within the Tool, it may be appropriate for the Tool to return some outcome data to the LMS. The Tool delegates this to its TIR's outcome service, which generates an outcome message conforming to the mutually supported outcome and security profiles and sends it to the LMS TIR's outcome service. The LMS TIR's outcome service processes the outcome message (including the security header) and sends an outcome response back to the Tool's TIR informing it of the status of the outcome processing (success/failure);

6) The user completes the interaction with the Tool and returns to the LMS's delivery context. This is either by closing the new browser window or by selecting some other UI navigation control with the LMS (in the case the Tool was launched inline). Note that there is no specialized session handling implied by the TI framework, i.e., issues around session timeouts, for example, within the host LMS while the user is interacting with the Tool in a new window are not addressed by these guidelines.

3.3 Persistence Model

The overarching persistence model assumed for the Tools Interoperability Framework and implemented by the LMS TIR is one in which the Proxy Tool's associated Tool is fully responsible for the persistence of any data required to deliver its functionality, external to the LMS data tier. So, in essence, all data and storage are private and localized for each Proxy Tool and associated Tool service. The LMS is not required to persist any data on behalf of the Proxy Tool/Tool service (other than the deployment descriptor settings of course), unless it is explicitly noted as such by either the TIR interface, i.e., the Configuration Manager opaquely persists and returns Tool specific descriptor, and/or other defined Extended Service Interfaces.

3.4 Service Interface Model

The overall service interface model for the Proxy Tool, TIR and related services, and any supported extended service functionality is as follows:

- XML based manifest and data oriented / driven interface (i.e., descriptors imply invocation of appropriate TIR services and impose certain behavior);
- HTTP-based service/component oriented interfaces;
- SOAP/WSDL based service interfaces as appropriate.

3.4.1 Proxy Tool/Tool Interoperability Interfaces

All the Proxy Tool and TIR interfaces are a combination of XML-based manifest and data-driven interfaces and HTTP-based service/component oriented interfaces. The key aspects, namely the Proxy Tool launch and the (optional) outcome reporting are accomplished via web services. We generally expect these web services to conform to the IMS General Web Services Base Profile, with any exceptions explicitly noted.

3.5 Error Logging/Handling Model

This version of the Tools Interoperability framework does not mandate a formalized error logging/handling model. At the same time, it expects that the following error scenarios will be handled by the LMS, Tool, and/or the respective TIRs:

- **Proxy Tool Deployment**: The LMS's TIR will display and log appropriate error messages during the deployment of a Proxy Tool's package;
- **Proxy Tool Launch**: The LMS in collaboration with its TIR will display and log appropriate error messages during the launch of a Proxy Tool. The Tool in collaboration with its TIR will log corresponding messages in its deployment context for the same launch;
- **User Interaction with Tool**: The Tool will log any error messages using its normal mechanism for doing so, i.e., accessing a tool via a Proxy Tool or via a normal access are not different from the perspective of error logging/handling;
- **Proxy Tool Outcome**: The Tool in collaboration with its TIR will display and log appropriate error messages during the reporting of any outcome generated during he course of a users interaction with the tool. The LMS in collaboration with its TIR will log corresponding messages in its deployment context for the same outcome report.

3.6 User Authentication Considerations

There are two layers of authentication to consider in the interactions described by these guidelines: service to service authentication (meaning tool to LMS or LMS to tool authentication, validated at the Web Service layer) and user to service authentication (user to application, at the application layer). It is possible for the two to be tightly coupled, but not required.

For example, system-to-system authentication may occur in the launch request via a shared secret indicator embedded in the SOAP header, over SSL. Once the URL negotiation is complete and the user redirected to the tool; however, the tool may rely on application tier SSO that is independent of the Web Service layer. Other scenarios may include a case in which the tool is remotely hosted, with no shared infrastructure with the launching LMS. The tool may choose to trust the Web Service message, and use the profile provided for authentication, or it may choose to force authentication using a local provider.

4. Tools Interoperability Model

The TI Model consists of a set of schemas and a TIR with configuration, deployment, launch, and outcome services that process (consume/create) messages based upon these schemas.

4.1 Logical Model

The logical model of the Tools Interoperability Framework is as depicted in Figure 4.1 below.

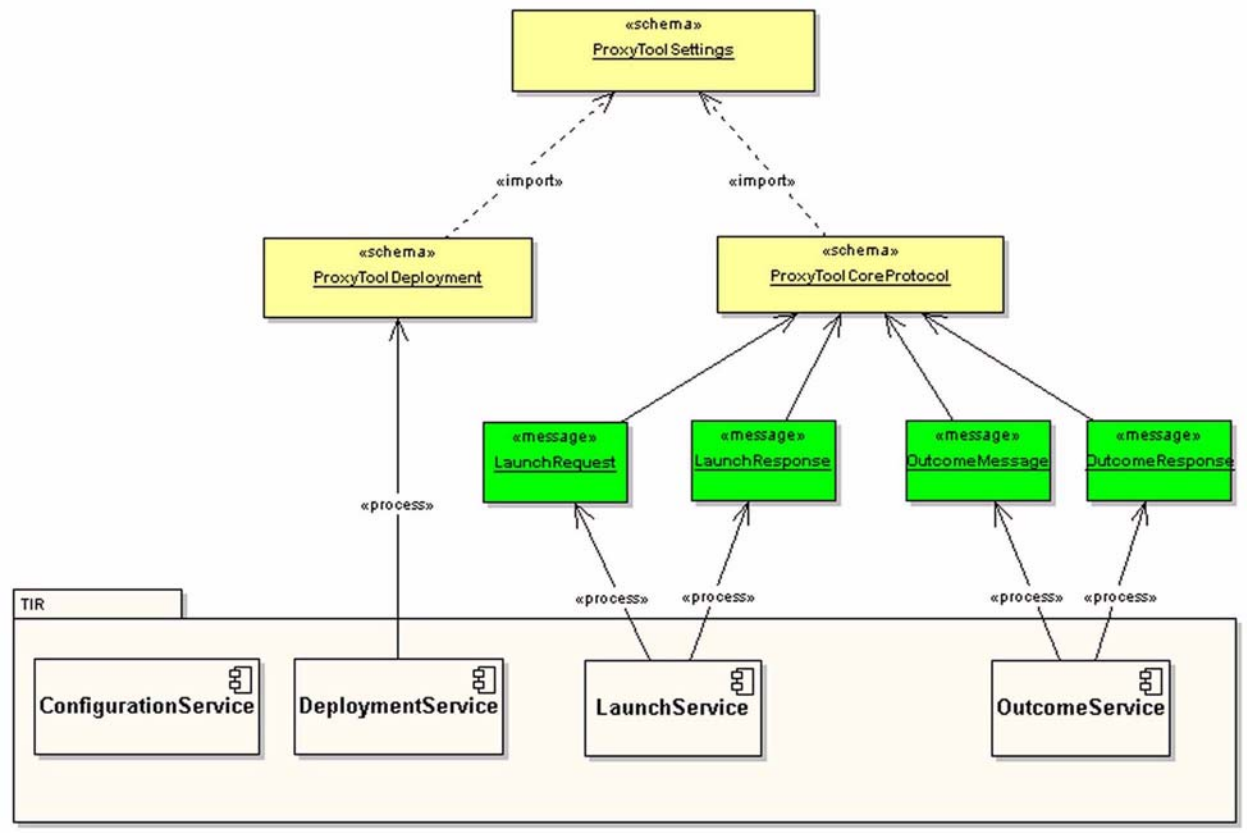

Figure 4.1 Tools Interoperability Schemas – Logical Model.

The model consists of:

- A base schema – ProxyToolSettings. This schema contains the definitions of all the types used in subsequent schemas;
- A deployment schema – ProxyToolDeployment. This schema utilizes the base types defined in ProxyToolSettings. It defines the schema of the deployment profile/s which is utilized in creating a deployment descriptor for a Proxy Tool;
- A core protocol schema – ProxyToolCoreProtocol. This schema utilizes the base types defined in ProxyToolSettings. It defines the schema of the launch and outcome messages that are utilized at runtime to launch the Proxy Tool and to deliver back an outcome respectively;
- A Tools Interoperability Runtime (TIR), which consists of four services – namely the deployment, launch, outcome, and configuration services as described previously in sub-section 3.2.5, sub-section 3.2.6, sub-section 3.2.7, and sub-section 3.2.8 respectively.

Note: As outlined in section 5 (Implementation Guidelines and Best Practices), the various schemas were inserted inline into the WSDL definitions of the launch and outcome services as a practical issue, with respect to support for schema import in commonly used tools.

In order to comply with the IMS General Web Services specification, the Launch and Outcome services were modeled using UML. The following diagrams depict the service and corresponding data model. Subsequent sections elaborate further on the various types defined in the data model.

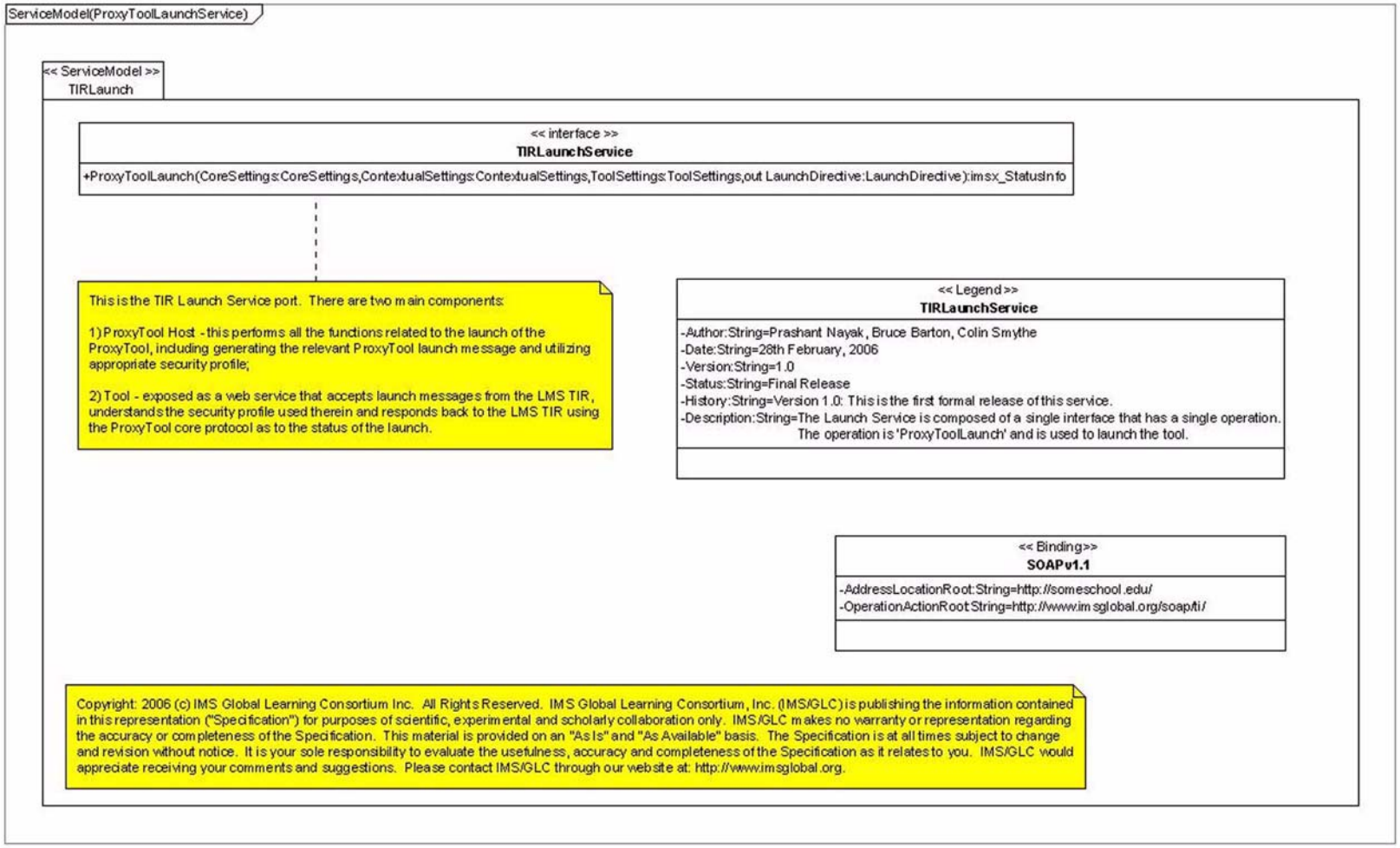

Figure 4.2 Launch Service UML Model.

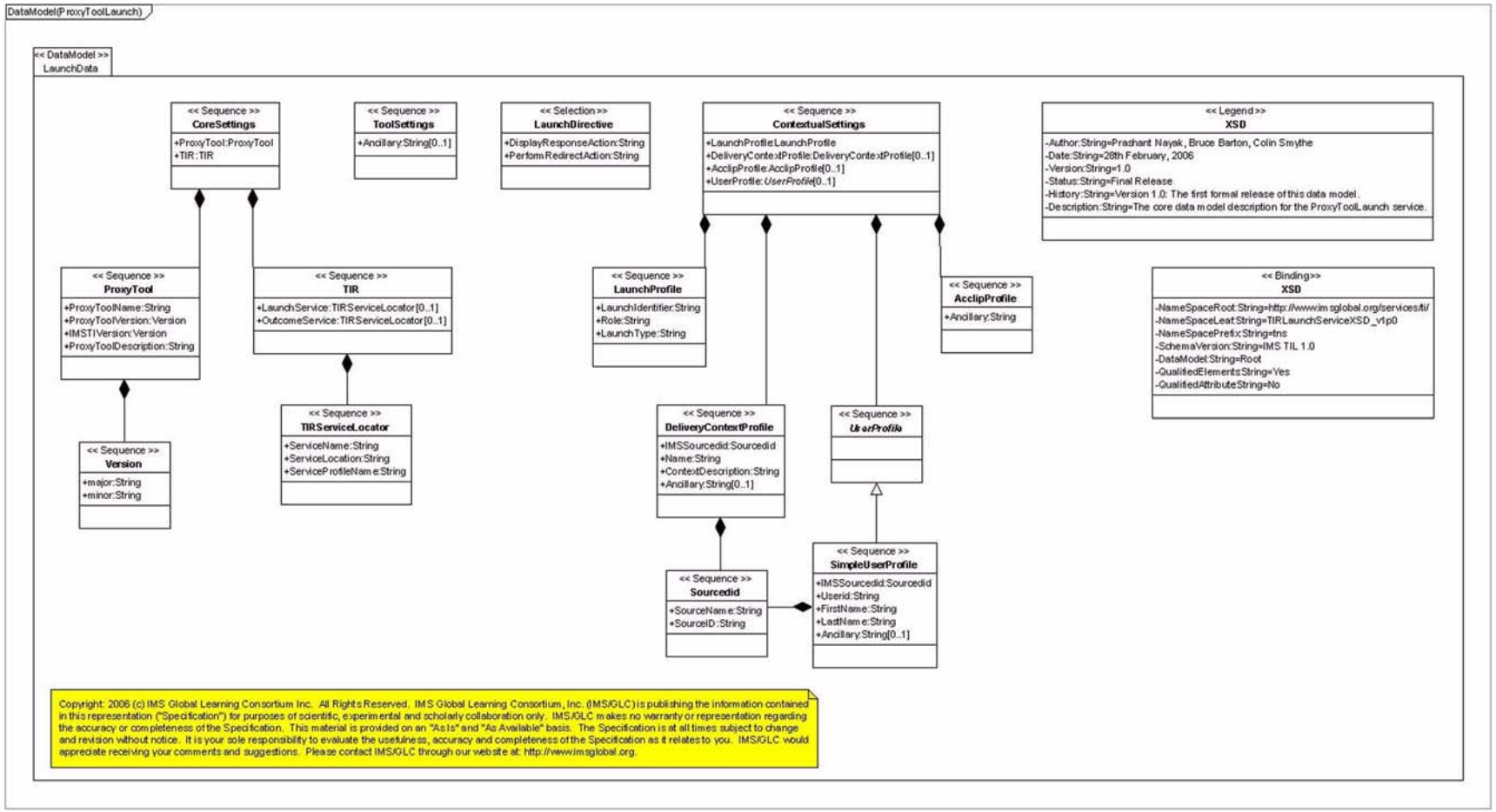

Figure 4.3a Launch Service UML Data Models.

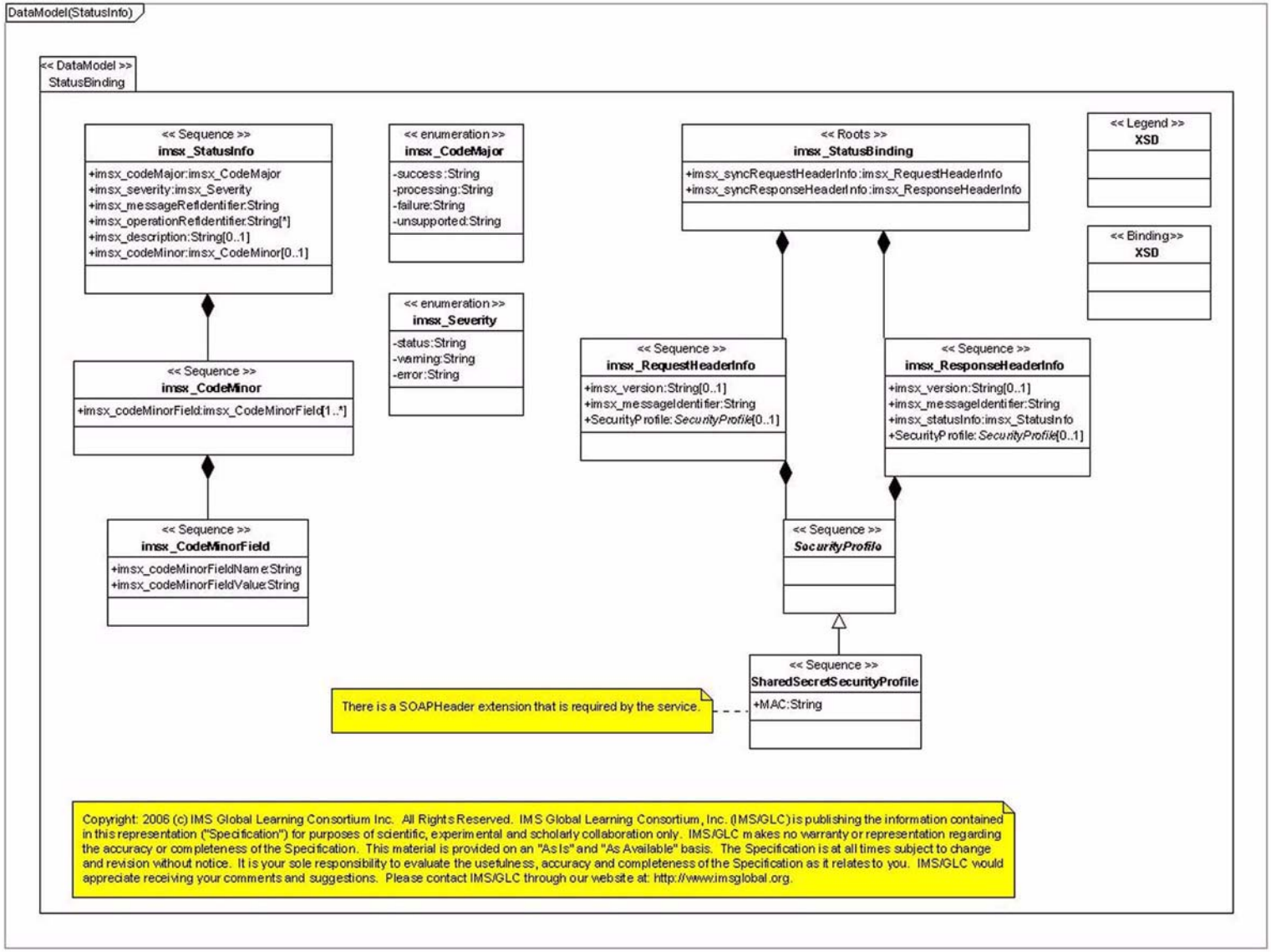

Figure 4.3b Launch Service UML Data Models.

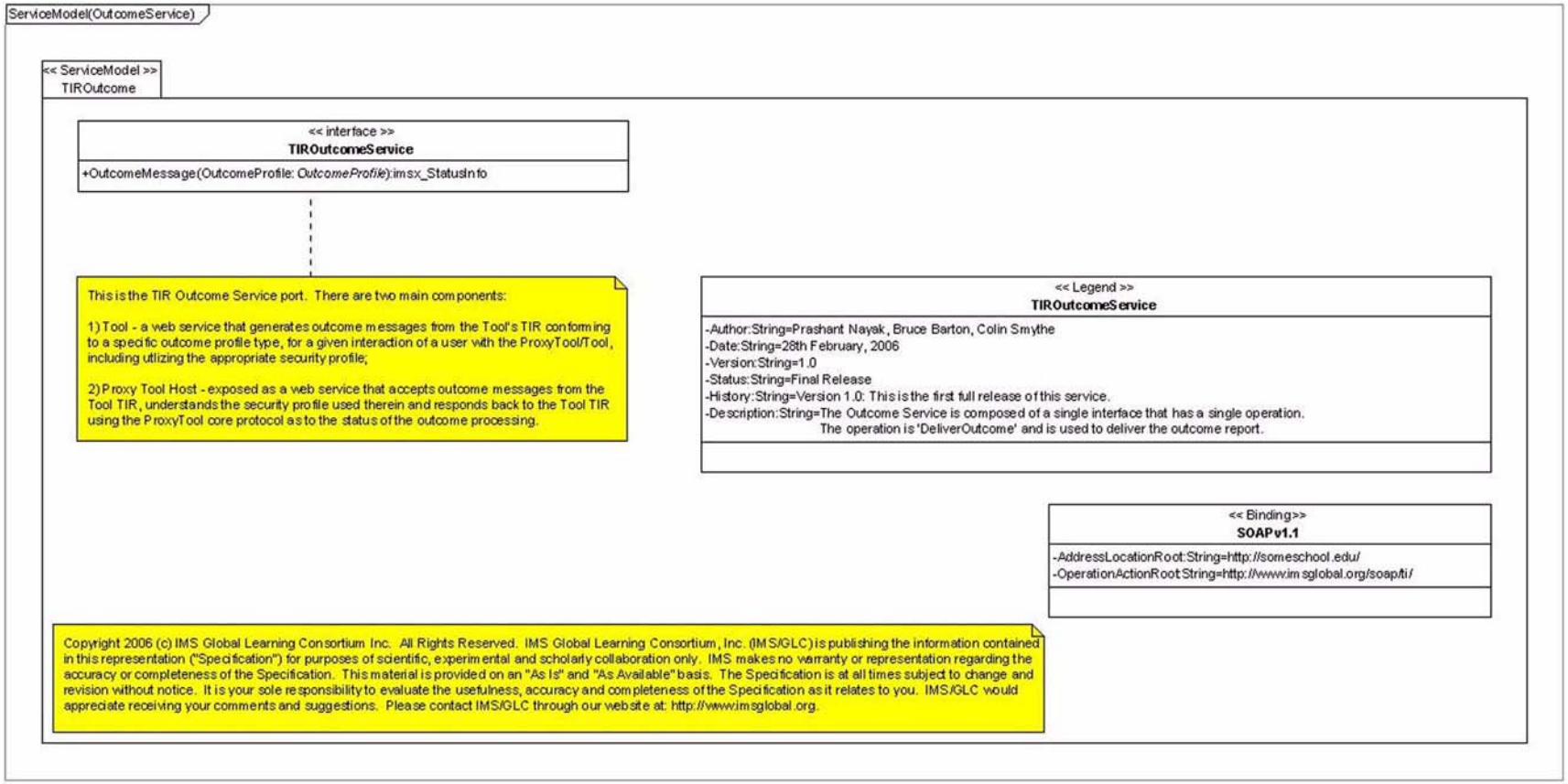

Figure 4.4 Outcome Service UML Model.

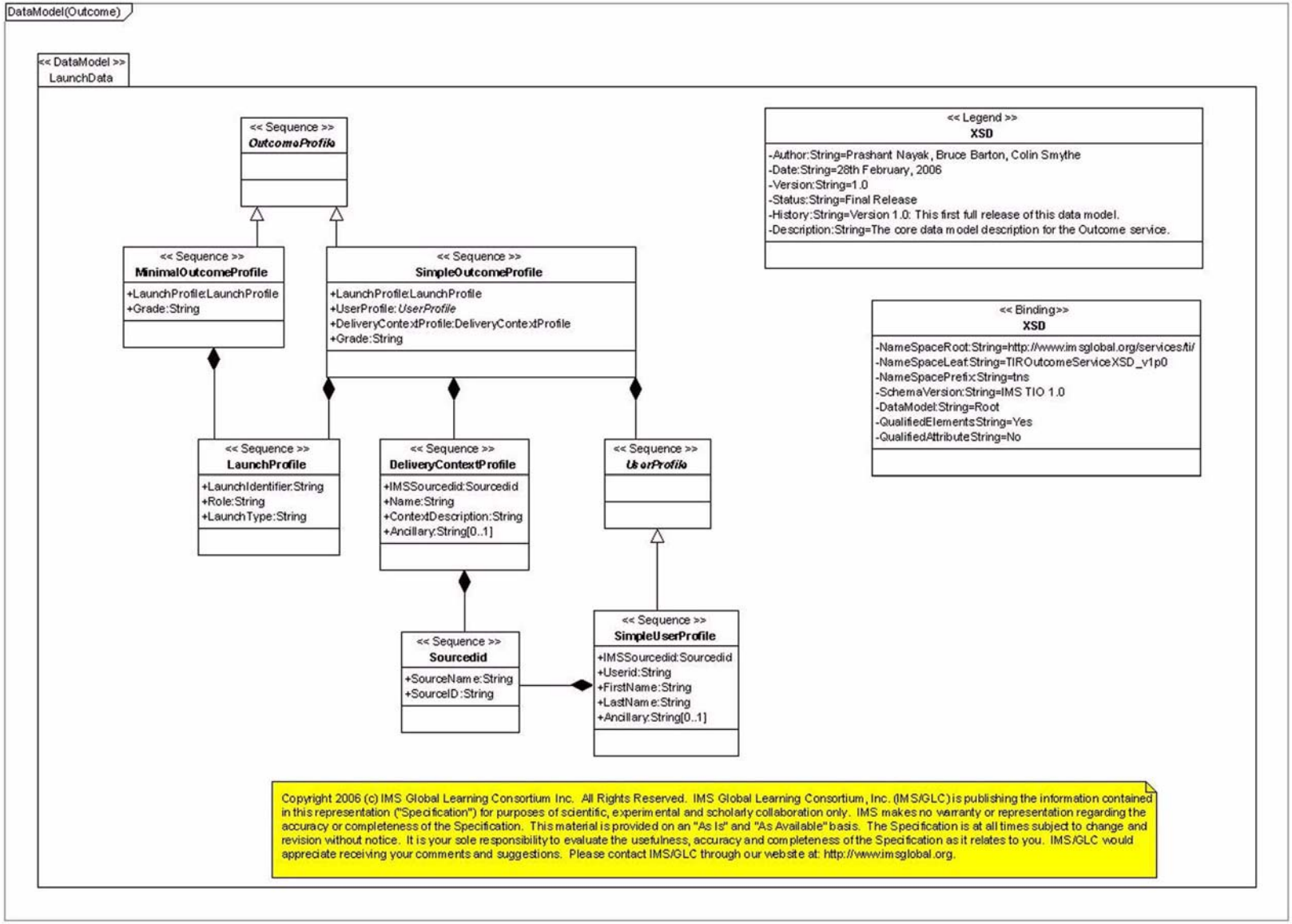

Figure 4.5a Outcome Service UML Data Models.

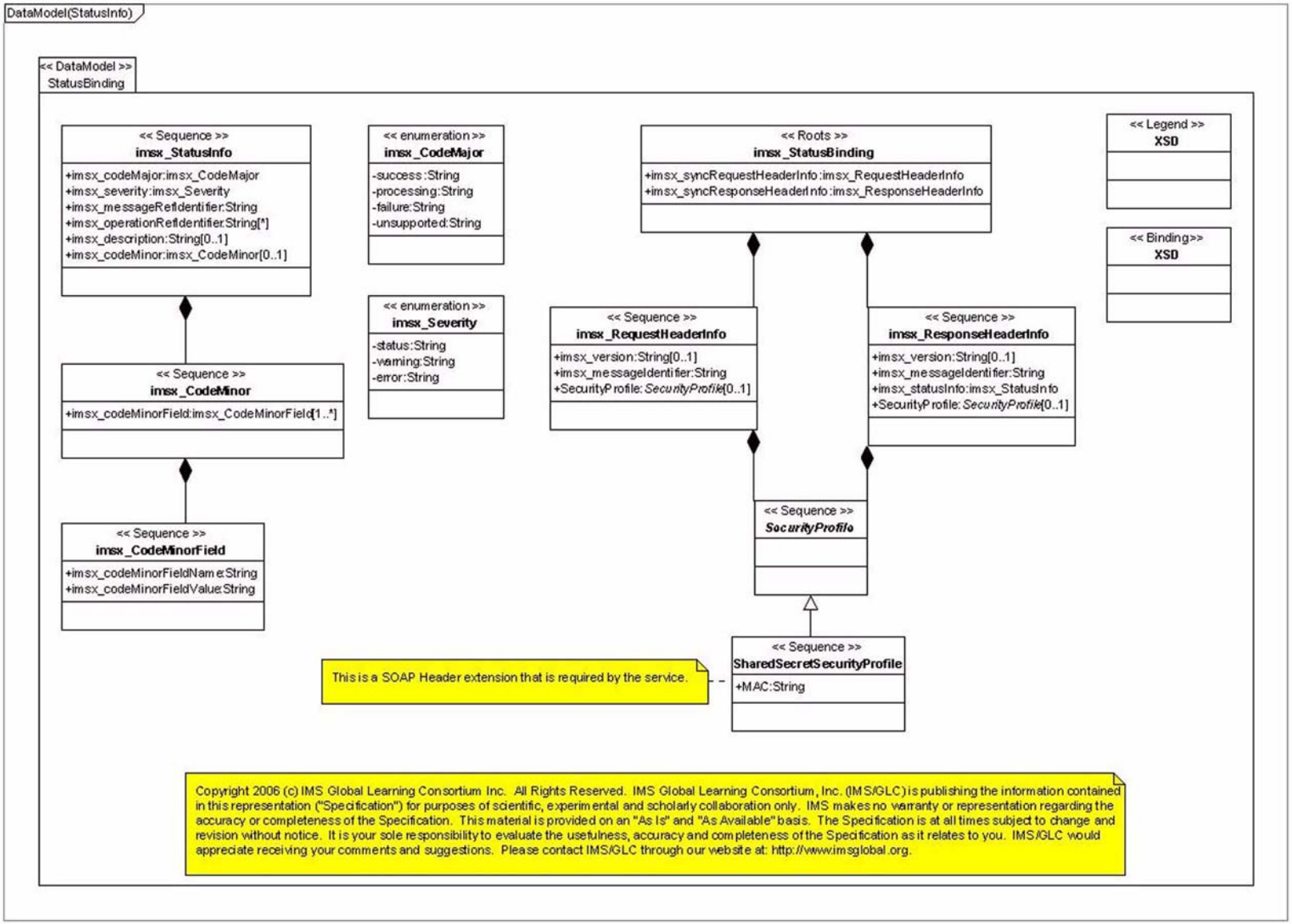

Figure 4.5b Outcome Service UML Data Models.

4.2 Base Types - ProxyToolSettings

The Tools Interoperability model defines several base types. This section outlines these types.

4.2.1 Core Settings

As depicted in Figure 4.6 below, the CoreSettings type is comprised of two sub-types – the ProxyTool type which specifies basic meta-data for the Proxy Tool (name, version, etc.) and a TIR type which defines the services that are required for the Proxy Tool within a TIR. At this point, only the externally exposed TIR services i.e. launch and outcome) are expected to be defined within any schema instance. The CoreSettings type is used primarily in the Proxy Tool deployment descriptor and also as part of the Proxy Tool launch.

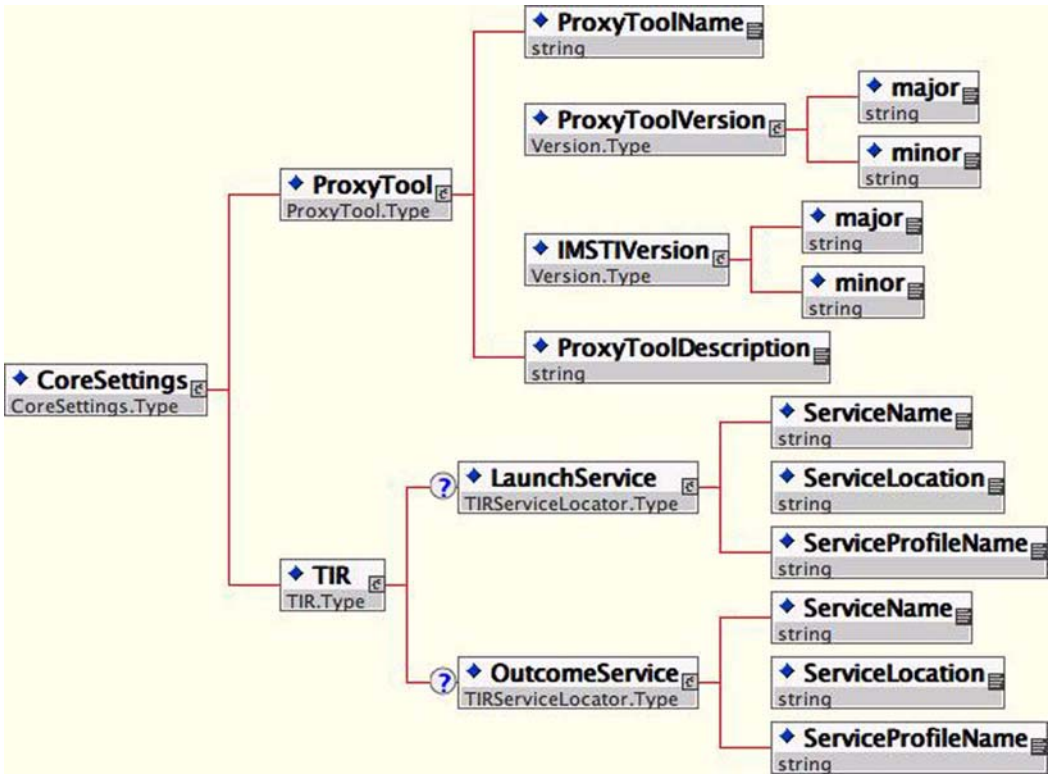

Figure 4.6 Proxy Tool - Core Settings.

4.2.2 Contextual Settings

The ContextualSettings type as depicted in Figure 4.7 is used to describe the context to which a particular Proxy Tool instance is deployed to and launched from. At deployment time, a Proxy Tool deployment descriptor can assert that certain contextual information has to be passed during a Proxy Tool launch. The LMS will validate that the required contextual information can be supported by it prior to deploying the Proxy Tool and will also send along the same during the launch.

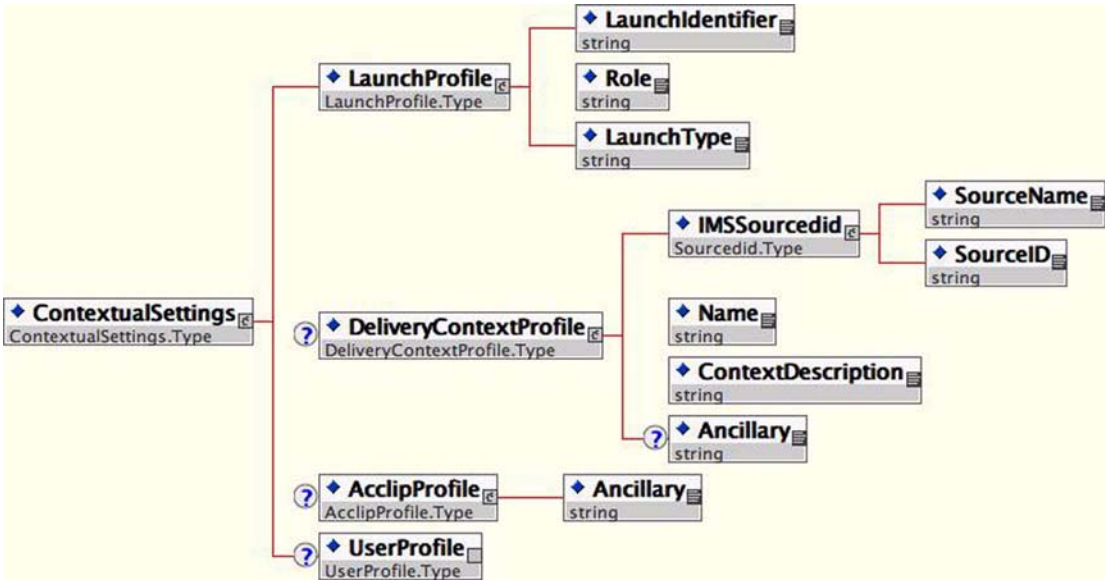

Figure 4.7 Proxy Tool - Contextual Settings.

The various elements of the ContextualSettings include:

- LaunchProfile – this includes a unique launch identifier (also used to correlate an outcome generated for a given launch, a role to be used for the launch (instructor, learner), and a LaunchType (normal, test);
- DeliveryContextProfile – this optional element provides information about the delivery context within which the Proxy Tool launch was initiated, e.g., a course or a section. Thus, this includes an IMS SourceId as well as a name, description, and an arbitrary number of ancillary (string) elements. Note that additional user profile types can be defined as required by the TI framework, e.g., based upon IMS Enterprise;
- UserProfile – this optional element provides information about the user for which the launch was initiated. This is a generic type and the TI model defines a single concrete type (SimpleUserProfile), as depicted in Figure 4.8 below:

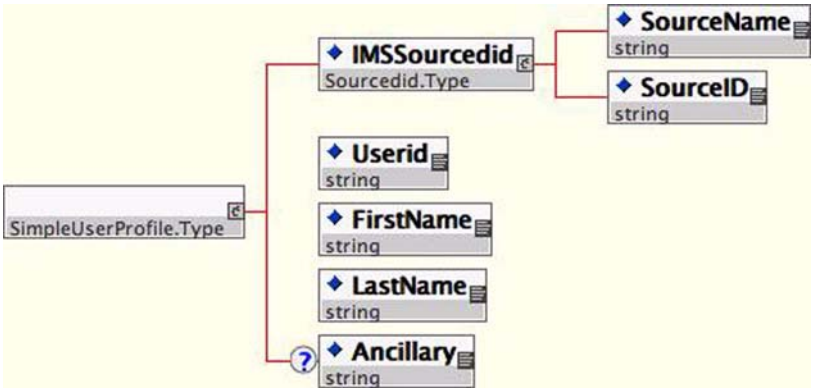

Figure 4.8 Proxy Tool Concrete UserProfileType - SimpleUserProfile.

The SimpleUserProfile includes the IMS source id for the user as well as base attributes (user id, first and last name and an arbitrary number of ancillary (string) elements). Note that additional user profile types can/will be defined as the need arises, e.g. a profile based upon IMS Enterprise.

- Other profiles – the contextual settings are extensible and could allow for additional profile types to be defined and specified by the TI framework as the need arises. This can include for example, LIP, ACCLIP profiles, etc.

4.2.3 Tool Settings

The ToolSettings type, as depicted in Figure 4.9 below, allows a Tool (developer) to specify settings specific to the Tool (potentially based on standards or custom schemas) as part of the Proxy Tool deployment descriptor.

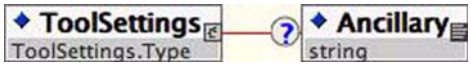

Figure 4.9 Proxy Tool – Tool Settings.

The deploying LMS is not required to perform any processing upon these settings beyond making them available for editing for a given Proxy Tool instance (within a given delivery context) and passing the same along during a launch for processing by the Tool's TIR.

4.3 ProxyTool Deployment

This schema reuses types from the base ProxyToolSettings schema to build deployment profiles for Proxy Tools.

4.3.1 ProxyToolDeployment Schema

As depicted in Figure 4.10, a Proxy Tool deployment descriptor allows specification of one or more deployment profiles for a given Proxy Tool.

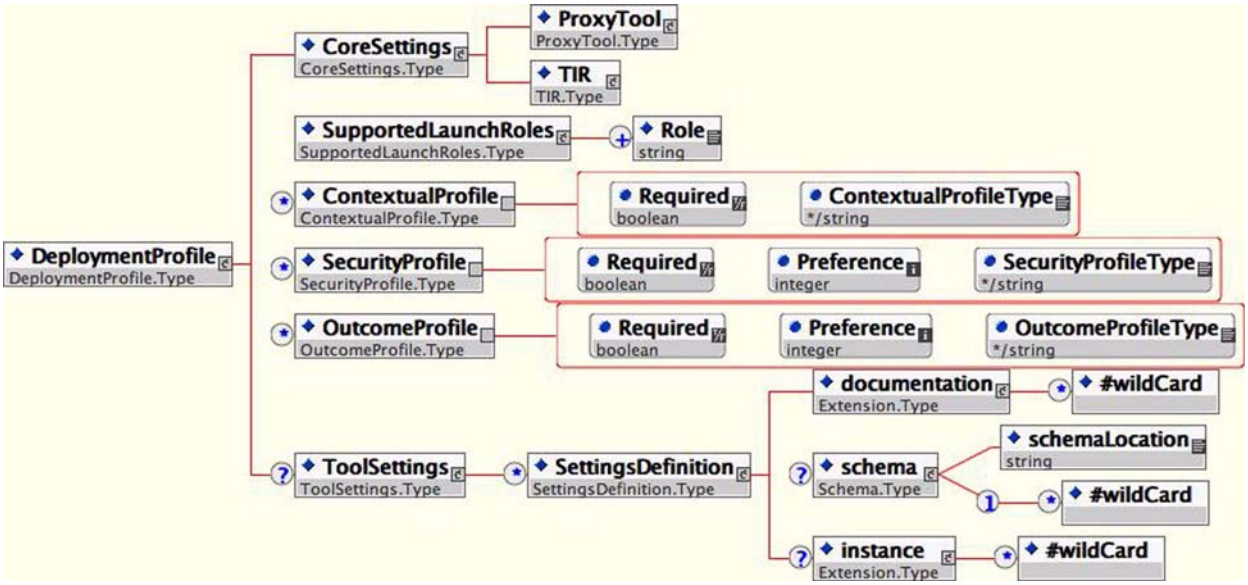

Figure 4.10 Proxy Tool – Deployment Profile.

Each deployment profile specifies the following required element:

- CoreSettings: The base set of meta-data for a Proxy Tool as defined in sub-section 4.2.1.

Additionally, each deployment profile optionally specifies the following:

- OutcomeProfile: One or more outcome profiles can be specified. By specifying these, the Proxy Tool is requiring the TIR deploying it to support the same and to pass along during the launch a TIR sub-element specifying the location of the outcome service that supports the outcome profile. Not that specifying outcome profiles is optional and there are valid use cases where no outcome is expected for a given Proxy Tool;
- ContextualSettings: The context specific settings that the Tool requires are passed along during the launch;
- SecurityProfile: One or more security profiles can be specified, thus requiring the TIR deploying the Proxy Tool to support the same and use it during the launch and outcome processing. Note that this element is optional and thus if none are specified, the cooperating TIRs can choose to use a standards based WS-Security profile (e.g., SAML);
- ToolSettings: The optional, Tool specific settings as specified in Figure 4.10 above.

4.4 Proxy Tool Launch

This schema primarily builds upon the base setting schema in order to provide the elements that will be used in constructing the Proxy Tool launch messages (launch and the corresponding response). It will add on types specific to the launch message and the subsequent response, including types for representing the status of the transaction, e.g., using the IMS GWS data structure for the same, etc.

The launch message will use this schema for constructing its primary payload. This payload will be encapsulated into a SOAP message. The security token (if a SecurityProfile is asserted by the Proxy Tool deployment profile) will be inserted into the SOAP header as a set of one-or-many header blocks, e.g., using the WS-Security schema.

4.4.1 Launch Request Schema

As depicted in Figure 4.11 below, the ProxyToolLaunchRequest comprises three types – the CoreSettings, ContextualSettings, and ToolSettings. Preceding sections have outlined these types in detail.

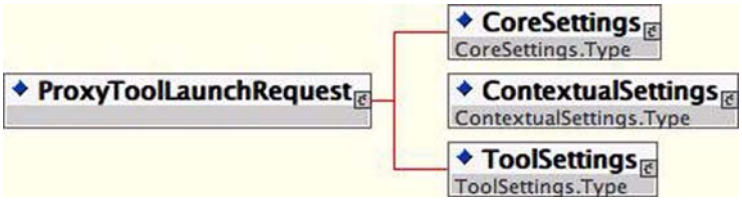

Figure 4.11 Proxy Tool Launch – Launch Request.

Note that the actual sub-elements of these types that are part of the launch schema instance are driven by the deployment profile for the Proxy Tool specified within its deployment descriptor. If the deployment descriptor specifies a particular OutcomeProfile and the LMS supports the same, then the CoreSettings will contain a TIR sub-element that specifies the location of the OutcomeService that supports this profile. Likewise, if the deployment profile specifies a particular SecurityProfile and the LMS supports the same, then the message will contain the SOAP header extension corresponding to that profile.

4.4.2 LaunchResponse Schema

As depicted in Figure 4.12 the LaunchResponse comprises a LaunchDirective that directs the launching TIR to perform either a redirect or to display a specific response to the end user. Note however that a LaunchDirective is only provided if the launch was successful. This, in turn, is specified by the StatusInfo IMS GWS SOAP header extension that is described in the next section.

Figure 4.12 Proxy Tool Launch – Launch Response.

4.4.3 StatusInfo Schema

The StatusInfo schema is as depicted below. As aforementioned, this is an element specified in the IMS GWS specification and is sent along as a SOAP header in the LaunchResponse.

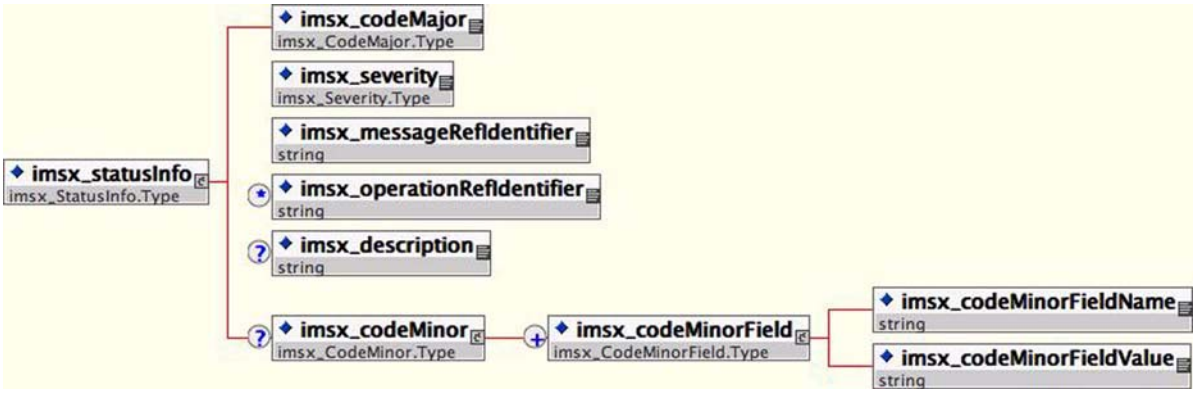

Figure 4.13 Proxy Tool – Reuse of IMS GWS StatusInfo Type.

4.4.4 SecurityProfile Schema

The TI model specifies a generic SecurityProfile type. In general, it is expected that cooperating TIRs will utilize standards-based security mechanisms, based upon SOAP header extensions for the launch and outcome messages. However, in order to provide a simple security mechanism for specific pairs of interoperating systems, we define a simple concrete type of security profile, namely the SharedSecretSecurityProfile, which relies on a shared secret between the two systems to assert identity. Note that we still specify the use of a SOAP header extension to encapsulate the shared secret element. Additionally, the Proxy Tool deployment descriptor will assert a SecurityProfile if the use of the shared secret profile is desired.

Figure 4.14 Proxy Tool – Shared Secret Security Profile (extends Security Profile).

4.4.5 LaunchProfile Schema

As depicted in Figure 4.16 below, the TI model utilizes a LaunchProfile, which includes a unique launch identifier (also used to correlate an outcome generated for a given launch, a role to be used for the launch (instructor, student) and a LaunchType (normal, test).

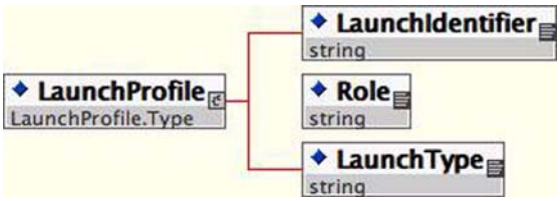

Figure 4.15 Proxy Tool – LaunchProfile.

The LaunchProfile is provided as part of the ContextualSettings during a Proxy Tool launch and as part of an OutcomeProfile during outcome reporting (with the same LaunchIdentifier to correlate an outcome to a launch).

4.5 Outcome Reporting

4.5.1 OutcomeMessage Schema

As depicted in Figure 4.16, the OutcomeMessage consists of a concrete instance of an OutcomeProfile type.

Figure 4.16 Proxy Tool Outcome Reporting – Outcome Message.

The Proxy Tool deployment profile specifies zero or more outcome profiles that it requires the LMS to support (zero if no outcome is expected from the Proxy Tool launch).

4.5.2 OutcomeProfile Schema

The OutcomeProfile is a marker type for various concrete outcome profile types. The TI model currently specifies two concrete outcome profile types – SimpleOutcomeProfile and MinimalOutcomeProfile. As depicted in Figure 4.17, the SimpleOutcomeProfile contains the following:

- LaunchIdentifier: Contained within the LaunchProfile, this is the same identifier sent by the LMS during the launch. This correlates the launch to the outcome;
- UserProfile: This corresponds to the profile that was specified during the Proxy Tool launch;
- DeliveryContextProfile: This corresponds to the profile that was specified during the Proxy Tool launch;
- Grade: This element specifies a simple grade or score as an outcome of the Proxy Tool launch (and subsequent user interaction with the Tool).

The MinimalOutcomeProfile depicted in Figure 4.18, is a minimalist version of the SimpleOutcomeProfile. It only contains the LaunchIdentifer and the Grade elements. It is expected that usage of this profile will benefit the Tool, with respect to not having to maintain and send back state e.g. user and delivery context, as part of the outcome. The launch identifier will usually suffice for the LMS to correlate the outcome to a previous Proxy Tool launch.

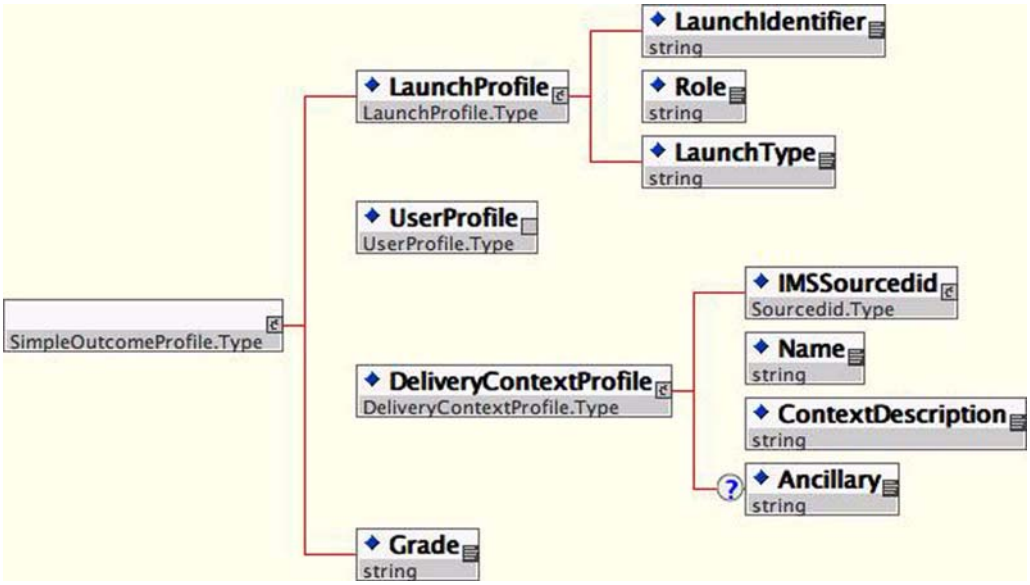

Figure 4.17 Proxy Tool Outcome Reporting – SimpleOutcomeProfile.

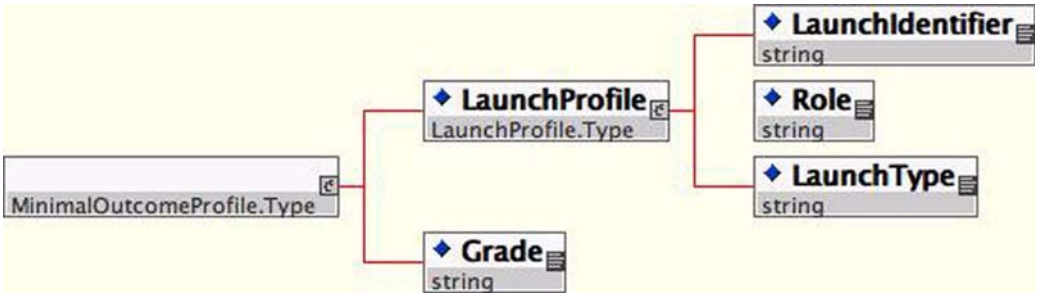

Figure 4.18 Proxy Tool Outcome Reporting - MinimalOutcomeProfile.

4.5.3 Outcome Response Schema

The OutcomeResponse is as depicted below. The response primarily comprises the StatusInfo SOAP header (not depicted, but as described in Figure 4.19).

Figure 4.19 Proxy Tool Outcome Reporting – Outcome Response.

4.6 ProxyTool Core Protocol

The Proxy Tool core protocol defines the sequence of interaction between two cooperating entities that support the TI framework via an implementation of the TIR. The protocol is simple and is defined as:

a) LMS's TIR sends a LaunchRequest message to the Tool TIR's Launch Service;

b) TIR Launch Service responds with a LaunchResponse message (fail, success, redirect to this URL, etc.);

c) The Tool at some logical point sends back an Outcome message – this step is optional because some tools might not report back an outcome;

d) The LMS's TIR processes the outcome and responds with an OutcomeResponse message.

4.6.1 Sequence Diagrams for Core Protocol

Figures 4.20 and 4.21 show possible implementation approaches both for the LMS and the Tool that are communicating using the TIR protocols.

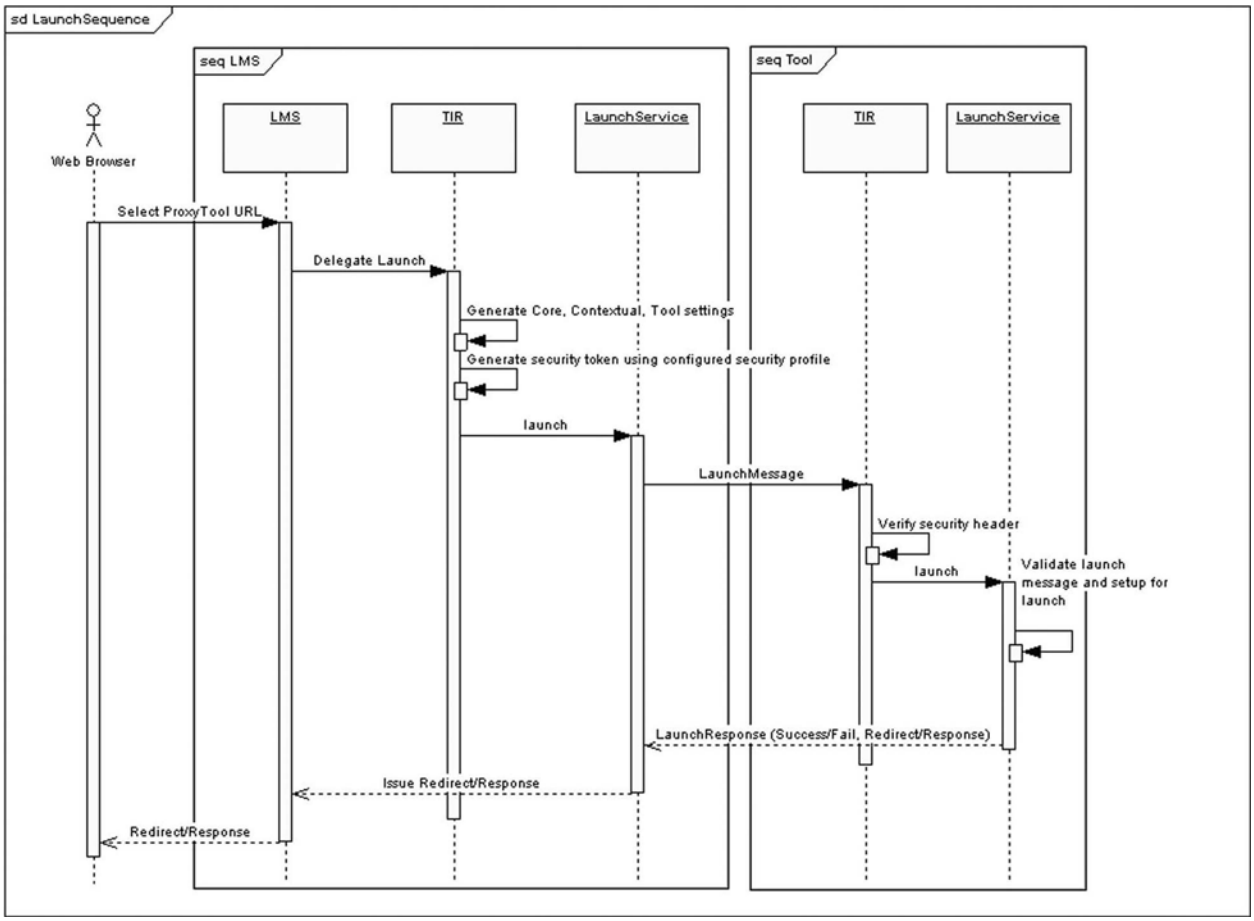

Figure 4.20 Proxy Tool Launch Sequence.

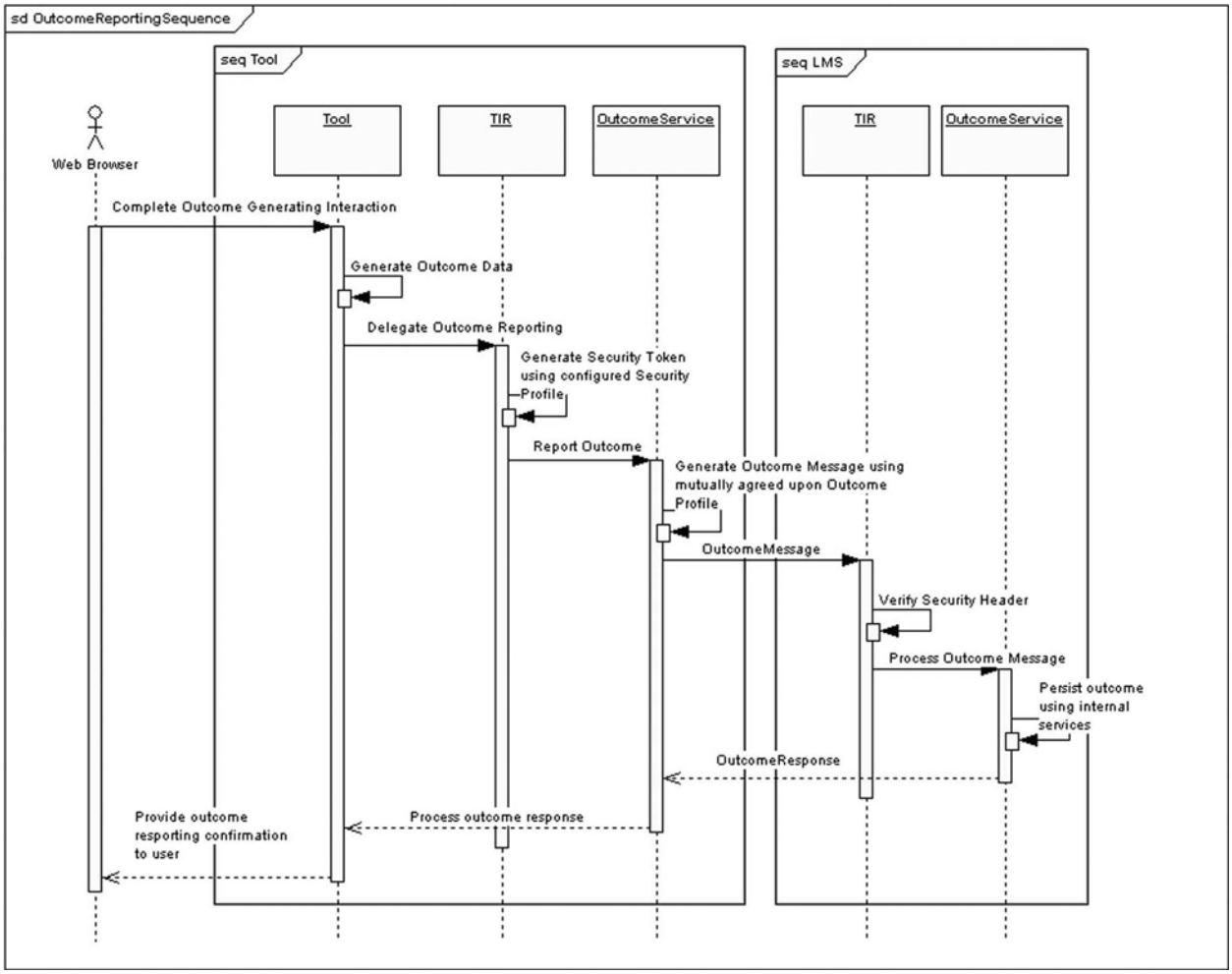

Figure 4.21 Proxy Tool Outcome Reporting.

4.7 Extensibility of Tools Interoperability Profiles

The TI schemas are designed to be extensible in order to support additional use cases as the guidelines are widely adopted in the future. This is accomplished by defining generic types of profiles that can be extended with concrete profiles and shows up in several places:

- Contextual Profile: The schema supports the addition of additional profiles such as LIP and ACCLIP profiles or others as the need arises in future versions of the specification. Additionally, the UserProfile is a generic type and TI currently provides a concrete SimpleUserProfile type (a simplified IMS Person type) and can be extended with additional concrete types;
- Outcome Profile: The schema currently supports two types of outcome profiles – the simple and minimal outcome profiles. It is fully expected that a future version of the TI guidelines will define additional outcome profiles, perhaps based on QTI or HR-XML schemas;
- Security Profile: The schema currently supports a concrete sub-type of security profile – the shared secret profile. Again, additional profiles can be defined in future versions of the guidelines if the need should arise (although in this case, it is expected that cooperating TIRs will widely adopt standards based WS-Security profiles instead).

5. Implementation Guidelines and Best Practices

5.1 General Best Practices

5.1.1 Security

Typically, when a LMS launches a Tool, the Tool does not independently authenticate the learner or verify the learner's authorization to use the Tool. The Tool, in effect, delegates authentication and authorization to the LMS. From the Tool's point of view all is well, so long as a trusted LMS initiates the launch.

Similarly, the LMS delegates to the Tool the responsibility for correctly recording and reporting the results, if any, of the learner's interaction with the Tool. Typically, these results are reported to the LMS through its Outcome Service and may be figured into a learner's final grade for a course, for certification, and the like. Again, all is well from the LMS's point of view so long as a trusted Tool reports outcomes.

In both the launch and outcome reporting scenarios each partner in the service call should verify the other's identity. This can be accomplished in many ways. One very simple method for accomplishing this is the SharedSecretSecurityProfile provided in these guidelines.

It is beyond the scope of these guidelines to provide a thorough analysis of the security context for tool interoperability and to recommend best practices for securing the interaction between the LMS and the Tool, and the learner's interaction with both. These guidelines describe message payloads for Web Services. Securing the Web Services themselves has been addressed elsewhere.

5.2 LMS TIF Implementation

5.2.1 State Management

The guidelines contain no requirements about time between a launch request and a posted outcome. Therefore, implementations should not rely solely on in-memory representation of context or launch data, such as Servlet sessions. Whichever outcome profile is specified in the Tool Deployment Descriptor, the LMS should be able to reconstruct enough data to successfully process the outcome request.

5.2.2 Management of Identifiable Attributes

Several identifiable attributes are specified as part of the Launch Profile. However, there may be attribute release policies in place at an institution that put restrictions on whether identifiable attributes may be released. Therefore, systems should provide controls for implementing those policies, and be able to alter Launch Requests to honor those policies. If attributes are released, a disclosure action is recommended.

5.2.3 Logging

Consideration should be given to the storage of runtime event messages emanating from the TIR as there is no provision in the guidelines for the TIR to provide its own logging persistence mechanism independent of the LMS.

5.2.4 Location of Proxy Tool Deployment Packages

The TIF should make provision for a Proxy Tool Deployment Package storage protocol organized in a manner that simplifies the identification of relevant deployment descriptors by tool administrators.

5.3 Tool TIF Implementation

The great advantage of the TIF for a Tool is that it provides the potential for a single, standard interface to LMSs and other management systems. Most Tools provide more benefit to their users if they are integrated within a wider system, but without a standard interface to work to, it's very time consuming for a Tool to integrate with all LMSs. If a Tool supports TIF, then once it is widely adopted, the Tool will then work with all LMSs that support TIG, and the corollary is that an LMS vendor needs only to support TIF and it will have access to all tools that support it.

In the commercial training world, the AICC HACP standard is very widely used to communicate between LMSs and content, but in the academic world this standard is not widely used, and there is no standard way of linking from LMS to Tool. The end result is that in 2005 only some tools and some management systems work together and when versions change, sometimes they stop working. Once TIF is established, it will be much easier for organizations to combine management systems and tools to get the best of each's functionality to work for them.

For a Tool to support TIF, it needs to provide at a minimum:

- A deployment descriptor, XML that describes what the Tool can do within TIF;
- A launch web service that can take input from the LMS.

Tool developers can either choose to build TIF into the core of their tool, or to build a wrapper program around their tool that interfaces with TIF in a standard way and then interacts with their tool in a proprietary way.

The deployment descriptor will typically be similar for different instances of the Tool. Some tools may automatically generate the deployment descriptor from their UI or with some configuration ability. Others may provide a sample deployment descriptor that the user edits to put in their URL.

It's impossible to be prescriptive about what the Tool will do on being called by the launch service as the nature of the TIF is such that it can support many different models of tools. One simple model of how a tool may work is in the example below. This is the scenario that was demonstrated at alt-*i*-lab in June 2005 and which many tools are likely to support with a role passed of student and a simple resource ID to identify the tool resource to be delivered.

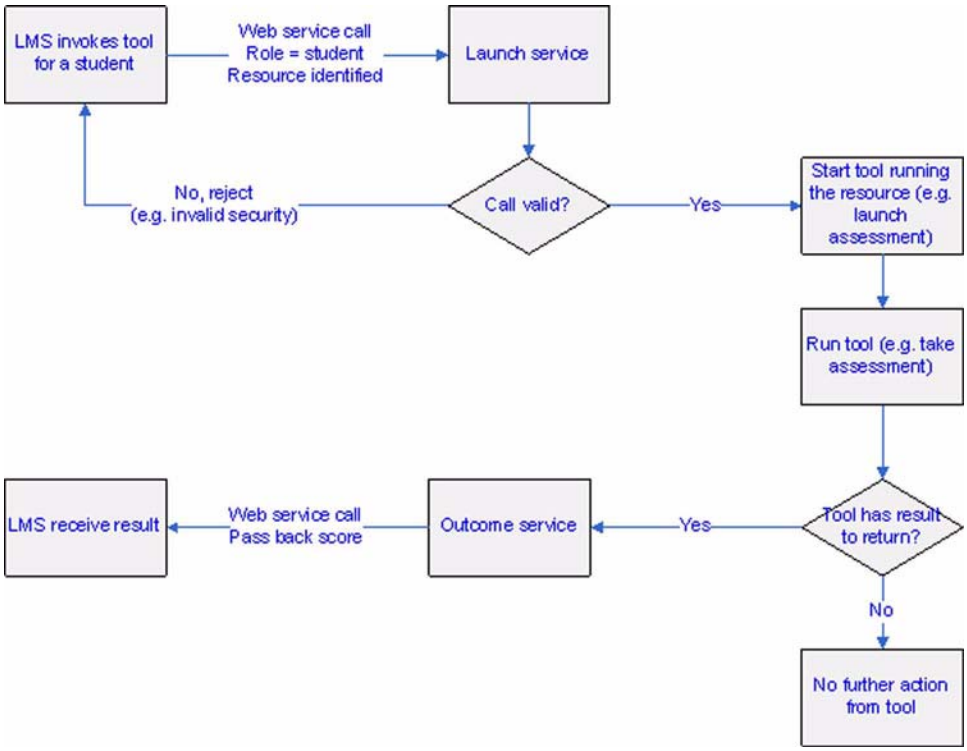

Figure 5.1 An Example of Tool Operation.

Note that not all tools will return a result via the outcome service. Some tools may be tutorial or informational only and not have a result to return, and even tools that do usually return a result, e.g., an assessment delivery system, may not always return a result, e.g., if the student abandons the tool by closing their browser.

The diagram in Figure 5.2 below shows a more complex possible tool set of interactions, where depending on the role and resource passed through from the LMS, different parts of the tool can be invoked. The diagram assumes an assessment management system.

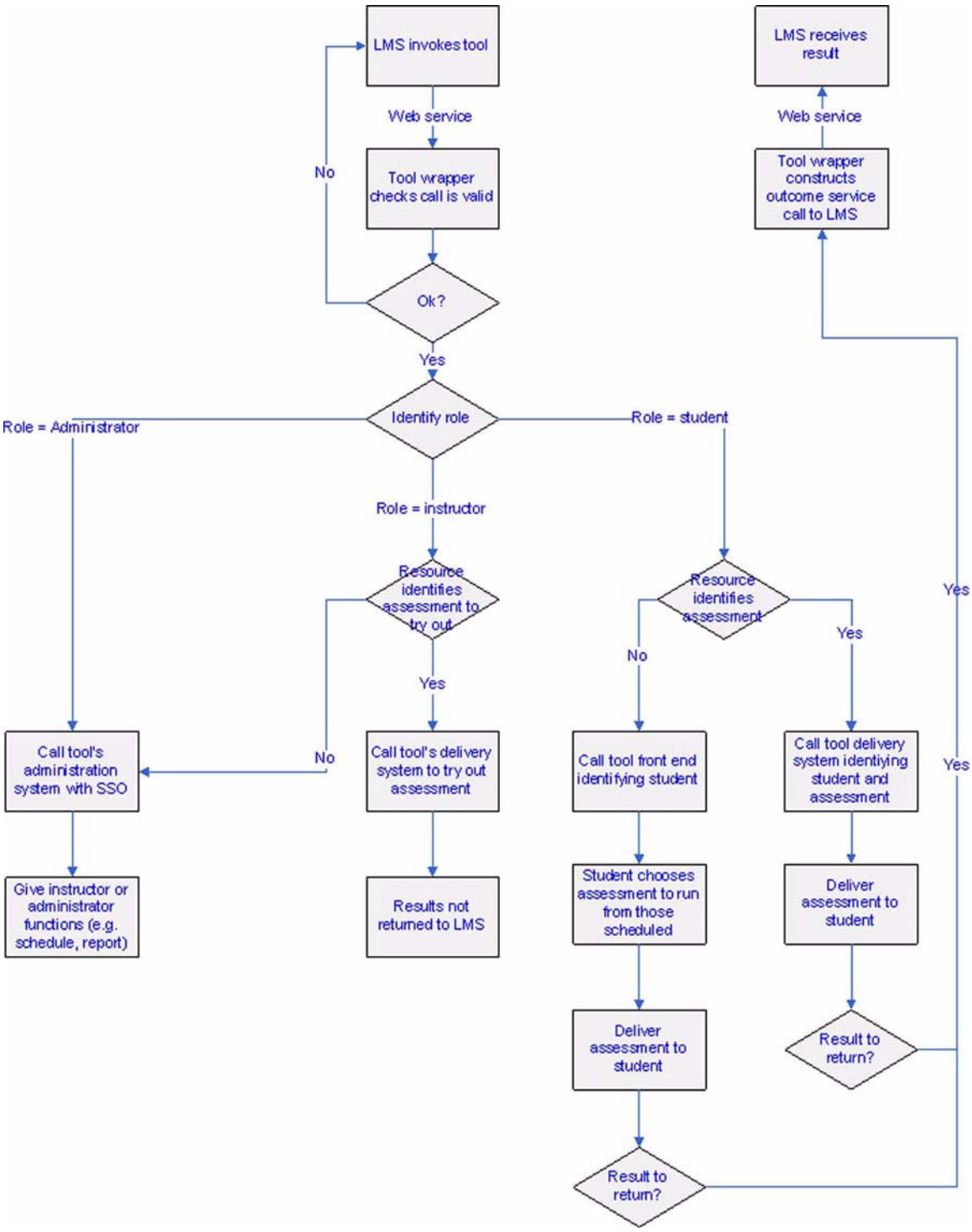

Figure 5.2 Impact of role and resource on the LMS/Tool interaction.

In this example, the tool has the following different possible actions depending on role and resource:

Role	Resource	Action
Administrator	(any)	Enter the tool as an administrator (run reports, schedule, author etc.).
Instructor	ID of assessment	Try out the assessment.
	Anything else	Enter the tool as an instructor (run reports, schedule, author etc.).
Student	ID of assessment	Deliver the assessment to the student.
	Anything else (SELECT recommended)	Sign the student into the tool and allow the student to choose from scheduled assessments and/or run reports.

Other variations of role, context and resource are possible. It's suggested that tools use the roles of Administrator, Instructor, and Student and map these onto meaningful roles where they can. Other roles should only be defined if essential as many LMSs will focus on these three roles. Some tools will choose to require administrators and instructors to be set up with privileges within their system to permit login. Others may give privileges based on the LMS role.

The resource ID should be the tool ID of an assessment or other resource. Typically, a numeric ID or GUID is used as resource ID. Special meanings of resource IDs can be interpreted by the tool; for instance a resource ID of SELECT might mean that the user should select the resource from those available. The tool documentation must identify all roles that are possible, as well as the meaning for resource ID.

It is recommended best practice that the Tool have an option to log all calls to and from the LMS, with the full contents of the XML received and sent, alongside the time and date this was done. This will be invaluable in diagnosing any problems that occur in operation.

5.4 Future Development

5.4.1 Presentation Logic

While the guidelines contain no requirements regarding the design or organization of presentation logic managed by the TIF it is recommended that the rendering of the TIR's graphical user interface (GUI) be given over to the LMS with the tool itself providing only an "abstract" expression of the GUI. Such an approach will permit the introduction of localized presentation elements without the need to rework the presentation logic contained within the TIR.

5.4.2 Handling Person Information

Future revisions of the guidelines will look to generalize and extend the representation of user profile data, to possibly incorporate existing standards (LIP) or arbitrarily negotiated formats (e.g., vcard).

5.4.3 Service Provisioning

Provisioning of TIF end points (Launch Service and Outcome Service) is currently defined "out of band" by the guidelines, requiring manual intervention on the part of Tool Administrators (generating or obtaining deployment descriptors) and LMS Administrators (installing deployment descriptors). Future work may look to emerging standards and best practices in service location and provisioning (e.g., UDDI, SPML, etc.).

5.4.4 Instructor Invocation of Tool (Resource Provisioning)

The following use case is not covered in the present version of the TIF, but has been identified as a desirable feature for future development.

Use Case 5	**Instructor Invocation of Tool (Resource Provisioning)**
Level	Summary
Primary Actor(s)	Instructor (Instructional Designer)
Secondary Actor(s)	LMS, Tool
Trigger	Instructor or Instructional Designer wishes to deploy a Proxy Tool resource into an LMS and needs to create or search for some resource provided by the tool.
Preconditions	• Proxy Tool defined and available in the system • LMS allows Tool launch for provision
Success Post-conditions	• Instructor activity launched, user directed to Resource Provisioning workflow (via HTTP redirection, frame generation, or new window creation), resource identifier posted back to the LMS.
Failure Post-conditions	• LMS-defined error state and communication to user
Main Success Flow	1. Instructor visits a Deployment Context containing a Proxy Tool instance 2. Instructor initiates a navigation event triggering tool launch (e.g., clicks on a link rendered by the LMS). 3. TIR reads Instance Deployment Descriptor. 4. TIR contacts Tool's Launch Service, passing Context-dependent arguments, identifying the launch as a resource provision. 5. Tool returns URL for display to Instructor. 6. TIR renders URL (via an iFrame, standard frame, HTTP redirect, or new Window). 7. Instructor interacts with workflow provided by tool to select or create a resource. 8. Tool authenticates any security assertions passed as part of the initiation. 9. Instructor completes selection activity in Tool. 10. Tool posts resource data back to LMS to create a suitable deployment context.
Variations	(a) Proxy Instance Supports (or spec defines) passing credentials directly to Tool, e.g., identity assertion or credential forwarding, instead of SSO implementation, e.g., tool verifies identity independently. 8(a) Security assertion happens as part of step 4. See Authentication examples, above (b) Tool requires data retrieved from LMS to process and/or render resource or activity.
Exception Conditions	• Launch service cannot be resolved or contacted from LMS • Security assertion fails • Tool failure (general) • LMS Response Service cannot be resolved or is otherwise unavailable from the Tool

6. Sample Implementations

6.1 Testing an LMS Implementation of the TIF

The LMS Test Harness is a lightweight implementation of the tool side of the TIF suitable for exercising an LMS implementation of the TIF. The test harness provides functionality for receiving a launch message, establishing a user session, delivering a simple assessment to the user's browser, and reporting the results of the assessment to the LMS's OutcomeService. The test harness delivers the functionality needed to complete a round trip from launch to outcome.

The test harness supports both the Minimal- and SimpleOutcomeProfiles and can be configured to require a UserProfile, DeliveryContextProfile, and AccessibilityProfile by altering its deployment descriptor. The tool also supports the use of SimpleSecurityProfile with the limitation that the MAC is generated against the messageIdentifier. Users should alter the deployment descriptor as needed to invoke the desired configuration. Note that the tool can only satisfy the requirements of the SimpleOutcomeProfile if both a SimpleUserProfile and DeliveryContextProfile have been supplied in the launch message.

The test harness has been built in Java using open source tools and is available in binary and source code distributions for use without a licensing fee. Minimal requirements for running the tool are Java 1.4.2 and a servlet container that supports the Servlet 2.4 specification. Beyond locating content files somewhere on the file system, the test harness makes no special demands of its host.

This test harness is designed to supplement general commercial and open source tools for testing web services. Its functionality is limited to that specified in the TIF. It does not analyze SOAP messages for compliance with published standards or best practices. Used in conjunction with such tools, the test harness provides a means of conveniently exercising a LMS implementation of the TIF.

6.2 Testing a Tool Implementation of the TIF

The Tool Test Harness is a lightweight implementation of the LMS side of the TIF suitable for exercising a Tool implementation of the TIF. The test harness provides functionality for loading a Tool's deployment descriptor, sending a launch message, accepting the launch response and redirecting the browser to the redirect URL, accepting an outcome message, and tracking and displaying launches and outcomes. The Proxy Tool test harness provides the functionality needed to complete a round trip from launch to outcome.

The Proxy Tool supports the configuration of roles in the deployment descriptor and allows the user to select among them when configuring a launch. The Proxy Tool supports both the Minimal- and SimpleOutcomeProfiles. Similarly, if the tool indicates support for more than one outcome profile in the deployment descriptor, the user can select among them in the Proxy Tool. When the deployment descriptor indicates that ContextualProfiles for User, DeliveryContext or Accessibility are required, the user will be prompted to supply this information when configuring a launch. The test harness also supports the use of SimpleSecurityProfile with the limitation that the MAC is generated against the messageIdentifier. Support for loading a tool's deployment descriptor is limited to a single deployment profile.

The test harness has been built in Java using open source tools and is available in binary and source code distributions for use without a licensing fee. Minimal requirements for running the tool are Java 1.4.2 and a servlet container that supports the Servlet 2.4 specification.

This test harness is designed to supplement general commercial and open sources tools for testing web services. Its functionality is limited to that specified in the TIF. It does not analyze SOAP messages for compliance with published standards or best practices. Used in conjunction with such tools, the test harness provides a means of conveniently exercising a Tool implementation of the TIF.

Appendix A – Support Files

A1 – WSDL and XSD Files

The following WSDL and XSD files are made available:

- TILaunchService WSDL file:
  http://www.imsglobal.org/services/ti/wsdl/TIRLaunchSyncSingle.wsdl
- TIOutcomeService WSDL file:
  http://www.imsglobal.org/services/ti/wsdl/TIROutcomeSyncSingle.wsdl
- Deployment Protocol XSD file:
  http://www.imsglobal.org/services/ti/xsd/imsti_ptdd_v1p0.xsd

A2 – Test Harness

The test harness (as described in section 6) is available at:

- TI Test Harness
  http://www.imsglobal.org/services/ti/software/titooltestharness.zip
- Proxy Tool Test Harness
  http://www.imsglobal.org/services/ti/software/tiproxytooltestharness.zip

About This Document

Title	IMS Tools Interoperability Guidelines
Editor	Kevin Riley (IMS, UK), Colin Smythe (IMS, UK)
Team Co-Leads	Bob Alcorn (Blackboard), Chris Vento (WebCT)
Version	1.0
Version Date	28 February 2006
Status	**Final Release**
Summary	The IMS Tools Interoperability (TI) approach addresses the growing demand for a reusable mechanism for integrating third party tools with core LMS platforms. These guidelines describe a Launch Service and the corresponding Outcomes Service that enable the LMS to launch a tool.
Revision Information	28 February 2006
Purpose	This document is circulated for formal adoption.
Document Location	http://www.imsglobal.org/ti/tiv1p0/imsti_guidev1p0.html

To register any comments or questions about this guideline please visit:
http://www.imsglobal.org/developers/ims/imsforum/categories.cfm?catid=17

List of Contributors

The following individuals contributed to the development of this document:

Name	Organization
Bob Alcorn (co-chair)	Blackboard
Chris Vento (co-chair)	WebCT
Kevin Riley	IMS, UK
George Ward	Cisco
Brad Wheeler	Indiana University
John Evdemon	Microsoft
John Kleeman	Questionmark
Niall Barr	Questionmark
Lydia Li	Stanford University
Stuart Sim	Sun Microsystems
Charles Severance	University of Michigan
Anthony Whyte	University of Michigan
Dirk Herr-Hoyman	University of Wisconsin – Madison
Bruce Barton	University of Wisconsin – Madison
Prashant Nayak	WebCT
Chris Etesse (reviewer)	Blackboard
Scott Wilson (reviewer)	JISC
Colin Smythe	IMS, UK

Revision History

Version No.	Release Date	Comments
Public Draft 1.0	11 November 2005	The first public release of the TI Guidelines.
Final Release 1.0	28 February 2006	The first final release of the TI Guidelines.

Index

IMS Global Learning Consortium, Inc. ("IMS/GLC") is publishing the information contained in this IMS Tools Interoperability Guidelines *("Document") for purposes of scientific, experimental, and scholarly collaboration only.*

IMS/GLC makes no warranty or representation regarding the accuracy or completeness of the Document.

This material is provided on an "As Is" and "As Available" basis.

The Document is at all times subject to change and revision without notice.

It is your sole responsibility to evaluate the usefulness, accuracy, and completeness of the Document as it relates to you.

IMS/GLC would appreciate receiving your comments and suggestions.

Please contact IMS/GLC through our website at http://www.imsglobal.org

Please refer to Document Name: IMS Tools Interoperability Guidelines

Date: 28 February 2006