



IMS GLC Learning Tools Interoperability (LTI) Implementation Guide

Version 2.0 Draft

IPR and Distribution Notices

Date Issued: 16 August 2011

Latest version: <http://www.imsglobal.org/lti/>

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 © IMS Global Learning Consortium 2010. All Rights Reserved.

If you wish to distribute this document or use this document to implement a product or service, you must complete a valid license registration with IMS and receive an email from IMS granting the license. 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 Licensee organizations 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/lti/ltiv1p0/ltiv1p0specllicense.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.

Join the discussion and post comments on the Basic LTI Public Forum:
<http://www.imsglobal.org/community/forum/categories.cfm?catid=44>

**© 2010, 2011 IMS Global Learning Consortium, Inc.
All Rights Reserved.**

The IMS Logo is a trademark of the IMS Global Learning Consortium Inc.
Document Name: IMS GLC Learning Tools Interoperability Basic LTI Implementation Guide Final v1.1
Revision: 23 June 2011

TABLE OF CONTENTS

1	INTRODUCTION	4
1.1	STRUCTURE OF THIS DOCUMENT	5
1.2	REFERENCES.....	5
2	LTI MESSAGES	6
2.1	COMMON PARAMETERS.....	6
2.2	CUSTOM PARAMETERS	7
2.3	BASIC LTI LAUNCH REQUEST	8
2.4	TOOL PROXY DEPLOYMENT REQUEST	9
2.5	TOOL PROXY REREGISTRATION REQUEST	9
2.6	SENDING MESSAGES.....	9
3	UNDERSTANDING THE TOOL PROXY.....	12
3.1	LINK RESOLUTION	22
3.2	PARAMETER SPECIFICATION.....	ERROR! BOOKMARK NOT DEFINED.
3.3	CAPABILITIES AGREEMENT	ERROR! BOOKMARK NOT DEFINED.
3.3.1	<i>Variable Substitution.....</i>	<i>Error! Bookmark not defined.</i>
4	ESTABLISHING AN INTEGRATION CONTRACT	24
4.1	REQUEST ACCESS TO A TOOL	24
5	USE CASES	25
6	LTI MESSAGE ITEMS.....	31
6.1	LTI TOOLPROXYDEPLOYMENTREQUEST AND TOOLPROXYREREGISTERREQUEST MESSAGE.....	32
6.2	LTI LAUNCH MESSAGE	34
7	LTI TOOL CONSUMER PROFILE.....	37
8	LTI SECURITY MODEL	39
8.1	LTI CREDENTIAL MANAGEMENT	39
8.2	OAUTH MESSAGE SIGNING FOR X-WWW-FORM-ENCODED MESSAGES	39
8.3	OAUTH MESSAGE BODY SIGNING FOR APPLICATION/XML MESSAGES	40
9	REPRESENTING BASIC LTI LINKS IN A CARTRIDGE	42
10	USING SERVICES	44
10.1	TOOL MANAGEMENT SERVICE	ERROR! BOOKMARK NOT DEFINED.
10.2	LIS OUTCOMES SERVICE.....	44
6.1.1	<i>replaceResult.....</i>	47
6.1.2	<i>readResult.....</i>	<i>Error! Bookmark not defined.</i>
6.1.3	<i>deleteResult</i>	<i>Error! Bookmark not defined.</i>
APPENDIX A – LTI STANDARD VOCABULARIES		49
A.1	CONTEXTTYPE VOCABULARIES	49
A.1.1	<i>LIS vocabulary for ContextType</i>	49
A.2	ROLE VOCABULARIES	49

A.2.1	<i>LIS vocabulary for System Role</i>	49
A.2.2	<i>LIS vocabulary for Institution Role</i>	49
A.2.3	<i>LIS vocabulary for Context Role</i>	50
APPENDIX B - IMPLEMENTATION PRACTICE		52
B.1	AUTHORING LINKS WITH LINK-LEVEL CREDENTIALS	52
B.2	SECURITY POLICY / SANDBOXING LAUNCH REQUESTS	54
B.3	ROLES	54
B.4	NON-CONTEXT LTI LAUNCHES	54
B.5	LTI SAMPLE LAUNCH	54
B.6	CONFORMANCE	55
B.7	ADMINISTRATOR / INSTRUCTOR USER INTERFACES / CUSTOM PARAMETERS	55
B.8	TP URL REMAPPING IN THE TC	56
APPENDIX C - CUSTOM PARAMETER SUBSTITUTION		57
C.1	LTI USER VARIABLES	57
C.2	LIS PERSON VARIABLES	57
C.3	LIS COURSE TEMPLATE VARIABLES	58
C.4	LIS COURSE OFFERING VARIABLES	58
C.5	LIS COURSE SECTION VARIABLES	59
C.6	LIS GROUP VARIABLES	59
C.7	LIS MEMBERSHIP VARIABLES	60
C.8	LTI LINEITEM VARIABLES	60
C.9	LIS RESULT VARIABLES	60
ABOUT THIS DOCUMENT		67
	LIST OF CONTRIBUTORS	67
REVISION HISTORY		68

1 Introduction

IMS is developing Learning Tools Interoperability (LTI) specifications to allow remote tools and content to be integrated into a Learning Management System (LMS). This document brings a subset of those specifications together in an implementation guide that defines a profile of LTI.

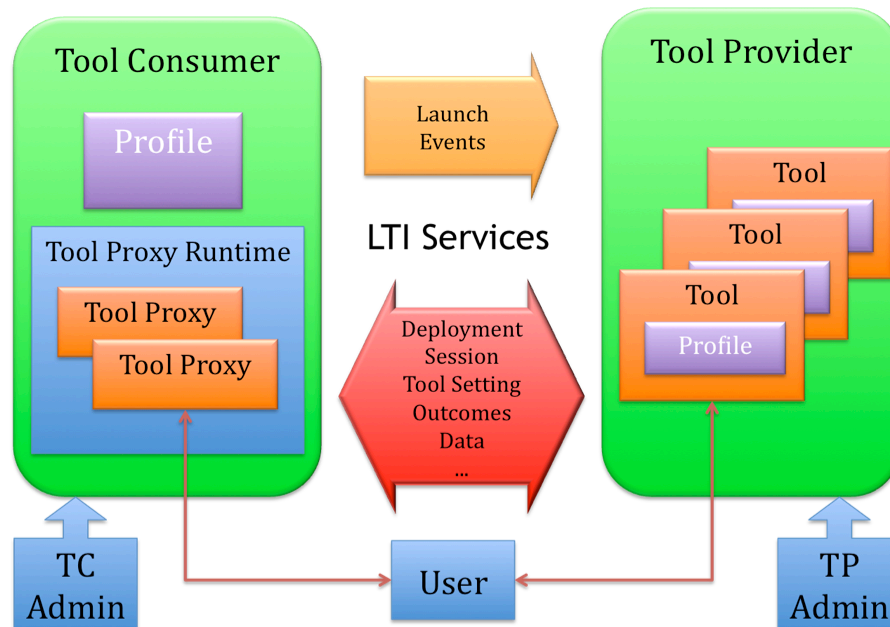


Figure 1.1: Overview of Learning Tools Interoperability.

Throughout this document, we use specific terminology to describe the two main pieces of software involved in LTI. What we traditionally think of as the "Learning Management System" (LMS) is referred to as the "Tool Consumer" (TC) as it "consumes" the tool. The external tool or content is called the "Tool Provider" (TP) as it "provides" the tool for use in the Tool Consumer. Example Tool Providers might include an externally hosted testing system or a server that contains externally hosted premium content.

This document describes two kinds of message flows. When the Tool Consumer (TC) is communicating with the Tool Provider (TP), the messages are signed POST messages transmitted through the user's browser often in an iframe. This "message through the browser" pattern is common for example when an application is using Twitter or Google for its sign on. The second kind of message is where the TP is calling a service within the TC using an XML request and response web service pattern to a service URL.

This implementation guide uses the OAuth 1.0a protocol (<http://www.oauth.net>) to secure its message interactions between the TC and TP. OAuth requires a key and shared secret to sign messages. The key is transmitted with each message, as well as an OAuth-generated signature based on the key. The TP looks up the secret based on the provided key and re-computes the signature and compares the recomputed signature with the transmitted signature to verify the sender's credentials.

As a best practice, the TP should isolate data based on the key. The TP must decide exactly how the key is used to isolate data. For example, the TP might maintain a table that maps multiple keys into a single data silo. Or, the TP might arrange to use the same key repeatedly in all cases where data are to belong to the same data silo.

The TC can make choices as to how it manages credentials (keys and secrets) within its system. LTI has four patterns for the credentials: (1) the TC-wide credential for a particular TP domain which is set by the TC administrator and used for all launches to a particular TP domain, or (2) the TC-wide credential for a particular TP URL which is set by the TC administrator and used for all launches to a particular TP URL, or (3) each LTI link is protected by its own credential, or (4) the link is associated with a Tool Proxy that manages the credentials. The

first and second patterns allow for a more seamless integration between a TC-instance and TP-instance from an instructor's perspective. The third pattern allows instructors to "mash up" Basic LTI links.

LTI has support for the TP to call IMS Learning Information Services (LIS) when those services can be made available to the TP. LTI does not require LIS services, but the TC can send LIS key information to the TP using values in the LTI Launch Request.

This document uses the term "context" where you might expect to see "course". A context is roughly equivalent to a course, project, or other collection of resources with a common set of users and roles. The word "context" is used instead of "course" because a course is only one kind of context (another type of context would be "group").

1.1 Structure of this Document

The structure of this document is:

2	USE CASES	A listing of the use cases describing the usage scenarios covered in this implementation guide
3.	LAUNCH DATA	A description of the data items that are passed as part of the POST data when an LTI launch is performed;
4.	SECURITY MODEL	The definition of the security environment for LTI;
5.	TOOL CONSUMER PROFILE	The description of the format of the Tool Consumer Profile
6.	REPRESENTING BASIC LTI LINKS IN A CARTRIDGE	A description of the Basic LTI link for inclusion in an IMS Common Cartridge;
7.	USING LEARNING INFORMATION SERVICES WITH LTI	A description of how to use LTI with IMS Learning Information Services;
	APPENDIX A LTI STANDARD VOCABULARIES	A reference to the LTI specification's standard vocabularies;
	APPENDIX B IMPLEMENTATION PRACTICE	A non-normative discussion and recommendations to help guide implementations.

1.2 References

As a subset of the LTI v1.0 specification, this document refers to other LTI documents that are not yet published publicly, as they are in draft state at the time of the completion of this specification. These LTI documents are listed here but will not be available publicly until they also reach completed status.

[CC, 08a]	<i>IMS GLC Common Cartridge v1.0</i> , IMS Global Learning Consortium, K.Riley, October 2008. http://www.imsglobal.org/cc/index.html .
[LIS, 10a]	<i>IMS GLC Learning Information Services Overview v2.0 Public Draft</i> , L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, March 2010. http://www.imsglobal.org/lis/index.html .
[LTI, 10 CCI]	<i>IMS GLC Learning Tools Interoperability Common Cartridge Interaction v1.0</i> , G.McFall, M.McKell, L.Neumann, IMS Global Learning Consortium, 2010. http://www.imsglobal.org/lti/index.html .
[LTI, 10 MSS]	<i>IMS GLC Learning Tools Interoperability Messages v1.0</i> , G.McFall, M.McKell, L.Neumann, IMS Global Learning Consortium, 2010. http://www.imsglobal.org/lti/index.html .
[LTI, 10 TMT]	<i>IMS GLC Learning Tools Interoperability Tool Management v1.0</i> , G.McFall, M.McKell, L.Neumann, IMS Global Learning Consortium, 2010. http://www.imsglobal.org/lti/index.html .

TODO: NEED GENERAL WEB SERVICES REFERENCE

2 LTI Messages

A *Message* is an HTTP request that passes from a Tool Consumer to a handler within the Tool Provider system. LTI 1.2 defines three different types of messages:

- Basic LTI Launch Request
- Tool Proxy Deployment Request
- Tool Proxy Reregister Request

Each message type specifies a set of required and optional parameters.

Section 2.1 describes the parameters that are common to all of the message types. Section 2.2 explains how custom parameters can be included in a message

Sections 2.3, 2.4 and 2.5 discuss parameters that are specific to the individual message types, and Section 2.6 discusses the mechanics of sending a message.

If a TC wants to send any additional or proprietary parameters, it should prefix all parameters not described herein with "ext_".

TODO: Add brief paragraph to introduce the concept of an HTTP POST Binding for messages and the distinction between standard parameters and custom parameters.

2.1 Common Parameters

lti_message_type=basic-lti-launch-request | ToolProxyDeploymentRequest | ToolProxyReregistrationRequest

This indicates the type of the message. This allows a TP to accept a number of different LTI message types at the same endpoint (a.k.a. handler).

lti_version=LTI-1p0

This indicates which version of the specification is being used for this particular message. This parameter is required in all messages.

user_id=0ae836b9-7fc9-4060-006f-27b2066ac545

Uniquely identifies the user. This should not contain any identifying information for the user. Best practice is that this field should be a TC-generated long-term “primary key” to the user record – not the “logical key”.

roles=Instructor

A comma-separated list of URN values for roles. If this list is non-empty, it should contain at least one role from the LIS System Role, LIS Institution Role, or LIS Context Role vocabularies (See Appendix A). The assumed namespace of these URNs is the LIS vocabulary of LIS Context Roles so TCs can use the handles when the intent is to refer to an LIS context role. If the TC wants to include a role from another namespace, a fully-qualified URN should be used. Usage of roles from non-LIS vocabularies is discouraged as it may limit interoperability.

launch_presentation_locale=en-US

Language, country and variant as represented using the IETF Best Practices for Tags for Identifying Languages (BCP-47) available at <http://www.rfc-editor.org/rfc/bcp/bcp47.txt>

launch_presentation_document_target=iframe

The value should be either ‘frame’, ‘iframe’ or ‘window’. This field communicates the kind of browser window/frame where the TC has launched the tool.

launch_presentation_css_url=

This is a URL to an LMS-specific CSS URL. There are no standards that describe exactly what CSS classes, etc. should be in this CSS. The TP should send its standard CSS URL that it would apply to its local tools. The TC should include styling for HTML tags to set font, color, etc and also include its proprietary tags used to style its internal tools.

Someday perhaps we will come up with a cross-LMS standard for CSS classes to allow a tool to look "built-in" with only one set of markup, but until that happens, the **launch_presentation_css_url** allows tools a chance to adapt their look and feel across LMS systems to some degree.

launch_presentation_width=320

The width of the window or frame where the content from the tool will be displayed.

launch_presentation_height=240

The height of the window or frame where the content from the tool will be displayed.

launch_presentation_return_url=http://lmsng.school.edu/portal/123/page/988/

Fully qualified URL where the TP can redirect the user back to the TC interface. This URL can be used once the TP is finished or if the TP cannot start or has some technical difficulty. In the case of an error, the TP may add a parameter called **lti_errormsg** that includes some detail as to the nature of the error. The **lti_errormsg** value should make sense if displayed to the user. If the tool has displayed a message to the end user and only wants to give the TC a message to log, use the parameter **lti_errorlog** instead of **lti_errormsg**. If the tool is terminating normally, and wants a message displayed to the user it can include a text message as the **lti_msg** parameter to the return URL. If the tool is terminating normally and wants to give the TC a message to log, use the parameter **lti_log**. This data should be sent on the URL as a GET – so the TP should take care to keep the overall length of the parameters small enough to fit within the limitations of a GET request.

2.2 Custom Parameters

In addition to standard parameters, like the ones described in the previous section, a Message may contain custom parameters that come from two possible sources:

1. The Tool Profile (see Section 3.4).
2. The creator of a link (see Appendix B.7)

Section 10 explains how custom parameters are represented when a link is stored in a Common Cartridge.

When there are **custom** name / value parameters in the launch, a POST parameter is included for each custom parameter. The parameter names are mapped to lower case and any character that is neither a number nor letter in a parameter name is replaced with an "underscore". So if a **custom** entry was as follows:

```
Review:Chapter=1.2.56
```

Would map to:

```
custom_review_chapter=1.2.56
```

Creators of Basic LTI links would be well served to limit their parameter names to lower case and to use no punctuation other than underscores.

If these custom parameters are included in the LTI link, the TC must include them in the launch data or the TP may fail to function.

TC implementations may have the ability to make value substitutions for custom parameters as described in Appendix C. For example, if a custom parameter was:

```
xstart=$CourseSection.timeFrame.begin
```

the parameter would be

```
custom_xstart=2012-04-21T01:00:00Z
```

Note that a DateTime data type in IMS LIS represents a combined date and time in the format of ISO 8601 i.e. 'YYYY-MM-DDThh:mm:ssTZD'. The time is denoted in Coordinated Universal Time (UTC) with TZD denoting the time zone offset in hours and minutes with respect to UTC.

It is important for a TP not to depend on the TC supporting any particular parameter substitution. If a TC that did not support parameter substitution were to see the above custom parameter, it would simply send

```
custom_xstart=$CourseSection.timeFrame.begin
```

as the parameter (i.e. send the parameter unsubstituted). It the responsibility of the TP to deal with both kinds of launches from TCs (i.e. with and without substitution available). The Tool Consumer Profile contains a list of all variables that the TC understands. If the TP registers a Tool Proxy with the Tool Consumer, there will be no ambiguity about which variables are supported and which are not. See Section **Error! Reference source not found.** for more details.

2.3 Basic LTI Launch Request

The Basic LTI Launch Request is used to launch a Tool from the Tool Consumer. In addition to the common parameters described in Section 2.1, the Basic LTI Launch Request information model includes the following additional parameters.

context_id=8213060-006f-27b2066ac545

(Optional)

This is an opaque identifier that uniquely identifies the context that contains the link being launched.

context_type=CourseSection

(Optional)

This string is a comma-separated list of URN values that identify the type of context. At a minimum, the list MUST include a URN value drawn from the LIS vocabulary (see Appendix A). The assumed namespace of these URNs is the LIS vocabulary so TCs can use the handles when the intent is to refer to an LIS context type. If the TC wants to include a context type from another namespace, a fully qualified URN should be used.

resource_link_id=88391-e1919-bb3456

(Required)

This is an opaque unique identifier that the TC guarantees will be unique within the TC for every placement of the link. If the tool / activity is placed multiple times in the same context, each of those placements will be distinct. This value will also change if the item is exported from one system or context and imported into another system or context.

tool_consumer_instance_guid=lmsng.school.edu

(Optional)

This is a unique identifier for the TC. A common practice is to use the DNS of the organization or the DNS of the TC instance. If the organization has multiple TC instances, then the best practice is to prefix the domain name with a locally unique identifier for the TC instance. This parameter is recommended.

Message Type	lti_message_type = basic-lti-launch-request
Required Parameters	lti_message_type, lti_version, user_id, roles, resource_link_id
Optional Parameters	context_id, context_type, launch_presentation_locale, launch_presentation_document_target, launch_presentation_css_url, launch_presentation_width, launch_presentation_height, launch_presentation_return_url, tool_consumer_instance_guid
Optional Parameters (Deprecated)	context_title, context_label, resource_link_title, resource_link_description, lis_person_name_given, lis_person_name_family, lis_person_name_full, lis_person_contact_email_primary, user_image, lis_person_sourcedid, lis_course_offering_sourcedid, lis_course_section_sourcedid, tool_consumer_instance_name, tool_consumer_instance_description, tool_consumer_instance_url,

	tool_consumer_instance_contact_email
--	--------------------------------------

See Appendix D for a discussion about the deprecated parameters.

2.4 Tool Proxy Deployment Request

This message initiates the Tool Proxy Deployment workflow within the TP as described in Section ???. In addition to the common parameters described in Section 2.1, the Tool Proxy Deployment Request includes the following additional parameters.

reg_password=918283799228bbejsdh999288 (Required)

This is the secret used to sign the RegisterToolRequest when the TP performs the operation. This will typically be a single use secret and only can be used in the context of this particular ToolProxyDeploymentRequest message. This parameter is required.

tc_profile_url=http://lmsng.school.edu/imslti/tool_consumer_profile.xml (Required)

This URL specifies the address where the TP can retrieve the TC tool profile. This URL must be retrievable by a GET request by the TP. If the URL is protected from retrieval in general, the TC must append necessary parameters to allow the TP to retrieve the URL with nothing more than a GET request. It is legal for this URL to contain a security token that is changed for each ToolProxyDeploymentRequest so the TP must retrieve the tc_profile_url on each request. This parameter is required.

launch_presentation_return_url=http://lmsng.school.edu/admin/continue_proxy.php (Required)

The Tool Provider redirects the user's browser back to this URL when the TP has completed the tool deployment process. The TP should redirect to this URL regardless of whether the deployment process succeeds or fails. In addition to the log and message parameters, described above, the Tool Provider appends the following HTTP query parameters

status=success | failure

tool_guid=<globally unique identifier for the Tool Proxy>

where the value for the tool_guid parameter is given by the return value from the registerTool operation if the operation was successful. Typically, this action redirects the administrator's browser to the Tool Console within the Tool Consumer system where the Tool Proxy can be made available. This parameter is required.

Message Type	lti_message_type = ToolProxyDeploymentRequest
Required Parameters	lti_message_type, user_id, roles, reg_password, tc_profile_url, launch_presentation_return_url
Optional Parameters	launch_presentation_locale, launch_presentation_document_target, launch_presentation_css_url, launch_presentation_width, launch_presentation_height

2.5 Tool Proxy Reregistration Request

The Tool Proxy Reregistration Request triggers the process to update a Tool Proxy that was previously deployed. This workflow is described in Section ???.

2.6 Sending Messages

The Tool Consumer must implement a *Tool Launch Service*. This service takes an LTI Link as input and produces an LTI Message encoded as an HTML Form. The typical workflow for launching a Tool can be summarized by the following sequence of events:

1. An end user (typically an Instructor or Learner) clicks on an LTI Link within the Tool Consumer web site.
2. The browser sends an HTTP request to the *Tool Launch Service* within the Tool Consumer. This request identifies the LTI Link that was clicked. (The link may be identified either by reference or by value.)
3. The Tool Launch Service constructs an HTML Form that represents the message that should be sent to the Tool Provider.
4. The HTML Form is loaded into the browser and auto-submitted to the appropriate Message Handler within the Tool Provider system.

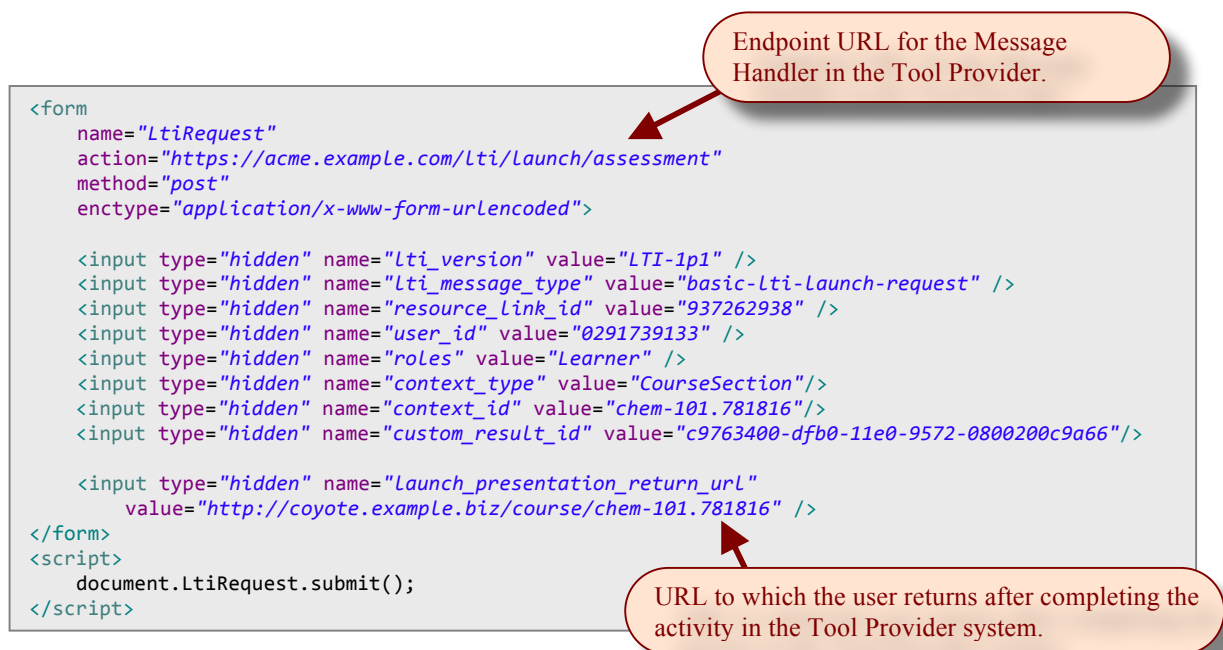


Figure 1 Sample Tool Proxy Deployment Request

When creating the HTML Form at Step 3 in the workflow, parameters must be rendered in accordance with the rules discussed in Section 2.2.

If the LTI Link is associated with a Resource Type, the Tool Launch Service must implement the following business rules:

1. **Discover Message Handler endpoint.** The Tool Launch Service discovers the endpoint URL for the Message Handler by inspecting the relevant Tool Profile. See Sections 4.1 and 3.4.5 for more details.
2. **Apply Capabilities.** If the Message Handler has enabled any capabilities, the business rules for those capabilities must be applied. For example, if the `Result.autocreate` capability is enabled, the Tool Consumer must create an LIS Result (and possibly a LineItem) prior to launching the Tool. (See Section 3.3.3 for a discussion of this particular capability.)

If the LTI Link is *not* associated with a Resource Type, then the link must define the Message Handler URL explicitly.

In all cases, the Message Handler URL is set as the `action` attribute of the HTML form.

TODO: Add discussion about oauth security parameters.

TODO: Add discussion about sending ToolProxyDeploymentRequest.

3 Tool Proxy

The two parties in an LTI integration (Tool and Tool Consumer) may exchange information about themselves in the form of Product Profiles. These profiles, together with security details, are aggregated to create an integration contract known as a ToolProxy. Figure 2 illustrates the top-level structure of a Tool Proxy.

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "guid" : "e8359010-009f-11e1-be50-0800200c9a66",
  "tool_consumer_profile" : { },
  "tool_profile" : { },
  "security_contract" : { }
}
```

Figure 2 Tool Proxy synopsis

We'll have more to say about the security contract later, but for now we focus on the Tool Profile and the Tool Consumer Profile.

3.1 Common Elements in a Product Profile

The Tool Profile and Tool Consumer Profile have several elements in common including:

- `product_instance`
 - `product_info`
 - `support`
 - `service_provider`
- `service_offered`

Much of this information is optional. The contents of these common elements are described below.

3.1.1 `product_instance`

```
{
  "guid" : "f188d950-dd73-11e0-9572-0800200c9a66",
  "product_info" : { },
  "support" : { },
  "service_provider" : { }
}
```

Figure 3 Synopsis of `product_instance`

The `product_instance` element represents a deployed instance of a given product (Tool or Tool Consumer). This element contains general information about the product (see `product_info`), plus optional contact information for support and optional details about the service provider that hosts the deployed application.

The deployed instance is identified by a `guid`. Typically, this will be a UUID, but it can be any globally unique string.

3.1.2 `product_info`

The `product_info` element contains generic information about a given product (Tool or Tool Consumer) such as the product name and description. Two products that belong to the same *product family* regarded as different versions of the same product, and the `version` field within the `product_info` specifies exactly which version is being used for the current integration. The product family is identified by a code that is unique within the namespace of some vendor. In Figure 4, the product family is identified by the code “`omega`”, and the vendor is identified by the code “`lms.example.com`”. The vendor code must be globally unique. As a best practice, this value should match an Internet domain name registered to the vendor.

Notice that the XML snippet contains additional information about the vendor including the vendor's name, description, website, and contact information. It also contains a timestamp. It is possible for multiple product profiles to encapsulate information about the same vendor. The timestamp allows you to determine which vendor record is most current.

```
{
  "product_name" : {
    "default_value" : "Omega LMS"
  },
  "version" : "2.3",
  "description" : {
    "default_value" : "Omega LMS is a fictitious Learning Management System"
  },
  "product_family" : {
    "code" : "omega",
    "vendor" : {
      "code" : "lms.example.com",
      "timestamp" : "2011-12-31T12:00:00",
      "name" : {
        "default_value" : "LMS Corporation"
      },
      "description" : {
        "default_value" : "LMS Corporation is a fictitious vendor of a Learning Management System"
      },
      "website" : "http://lms.example.com/sales",
      "contact" : {
        "email" : "support@lms.example.com"
      }
    }
  }
}
```

Figure 4 Example of product_info

3.1.3 support

```
{
  "email" : "techsupport@university.edu"
}
```

Figure 5 Example of support

The `product_instance` element may contain contact information for support, as shown in Figure 5. In LTI 1.2, email is the only means of contact.

3.1.4 service_provider

```
{
  "guid" : "yasp.example.com",
  "timestamp" : "2001-12-31T12:00:00-05:00",
  "provider_name" : {
    "default_value" : "Your Application Service Provider"
  },
  "description" : {
    "default_value" : "YASP is a fictitious application service provider"
  },
  "support" {
    "email" : "support@yasp.example.com"
  }
}
```

Figure 6 Example of service_provider

A service provider is responsible for hosting one or more product instances. The service provider is not necessarily a commercial entity. For example, if a Tool or Tool Consumer is locally hosted at a university, the service provider may be identified with the given university, a department within the university, etc. Notice that the service provider also contains an optional email address for support. Since the service provider may host multiple product instances, customers should use the more specific support email address (see Section 3.1.3) if one is defined.

Information about the service provider is entirely optional.

3.1.5 service_offered

```
{
  "contract" : "http://purl.org/ims/lti/v1/service/ResultService",
  "method" : [ "GET", "PUT"],
  "endpoint_url" : "http://lms.example.com/services/"
}
```

Figure 7 Example of service_offered

A given product (Tool or Tool Consumer) may offer a set of services to its integration partner. In this case, the product profile contains one or more `service_offered` elements like the one shown in Figure 7.

Every type of service has a URI which identifies the contract for that service. In the example above, the contract URI is <http://purl.org/ims/lti/v1/service/ResultService>. This example corresponds to the service that is described in Section 11.2.

A service contract specifies a number of methods on the service interface. A given party may wish to offer only a subset of those methods to its integratin partner. For example, a Tool Consumer might expose read methods for user data, but it might not allow the Tool Provider to create or update user records within the TC system. The service methods that are available to the integration partner are listed explicitly within the `service_offered` element.

Notice that the `service_offered` element also defines the URL for the service endpoint.

3.2 Tool Consumer Profile

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "@subject" : "urn:lti:ToolConsumerProfile:alpha.university.edu",
  "product_instance" : { },
  "service_offered" : [ ],
  "capability" : [ ]
}
```

Figure 8 ToolConsumerProfile synopsis

A Tool Consumer Profile contains information about one particular instance of a Tool Consumer product. In addition to the common elements discussed in Section 3.1, the Tool Consumer Profile announces the set of capabilities offered to the Tool Provider.

Each Tool Consumer Profile is identified by a URN specified by the `@subject`. This URN is generated from the template shown in Figure 9

```
urn:lti:ToolConsumerProfile:?tool_consumer_instance_guid
```

Figure 9 URI Template for Tool Consumer Profile

The `@context` parameter is required only if the Tool Consumer Profile is the first object in the serialized data. If the Tool Consumer Profile appears as a property of some other object, then the `@context` may be omitted because it will be defined by the parent entity. For example, if a serialized Tool Proxy may reference a Tool Consumer Profile as shown in Figure 10. In this case, the details of the Tool Consumer Profile are omitted. Only its URN is provided.

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "tool_consumer_profile" : { },
  "tool_profile" : { "@subject" : "urn:lti:ToolConsumerProfile:alpha.university.edu" },
  "security_contract" : { }
}
```

Figure 10 Reference to TC Profile by a Tool Proxy

3.3 Capabilities

A “capability” is a formal definition for some pattern of behavior. Each capability is identified by a URI. Ideally, the URI should point to a resource that provides a description of the capability. Those descriptive resources are not published yet. Consequently, the URI for a capability should be regarded as an opaque identifier. LTI 1.2 defines three broad kinds of capabilities:

- Variable Expansion Capabilities
- Messaging Capabilities
- Outcomes Capabilities

The Tool Consumer advertises the capabilities that it supports, and the Tool Provider chooses the capabilities that it wishes to enable. These decisions are recorded in the Tool Proxy.

rdf

Figure 11 Capabilities declared in a Tool Consumer Profile

Figure 11 shows how the TC declares the capabilities that it supports by listing them within the Tool Consumer Profile. Technically, each capability is identified by a fully-qualified URI, but in the JSON-LD binding they are identified by simple names (without a namespace prefix). The JSON-LD context (specified by the `@context` property) defines a mapping from the simple names to the fully qualified URI.

3.3.1 Variable Expansion

For each variable named in Appendix C, there is a corresponding formal capability which signifies that the TC has the ability to expand the specified variable. For example, the variable `$Person.name.given` has a corresponding capability identified by the URI <http://purl.org/ims/Lti/v1/variable#Person.name.given>.

The URI for all the other variables follows a similar pattern. To form the capability URI, simply append the name of the variable minus the dollar sign to the namespace <http://purl.org/ims/Lti/v1/variable#>.

Capabilities for variable expansion do not need to be enabled explicitly within the Tool Proxy. The mere use of a variable implies that the TP wishes to enable the capability.

3.3.2 Messaging Capabilities

Each message type in the LTI standard has a simple name and a fully qualified URI. To construct the full URI, simply append the simple name to the namespace <http://purl.org/ims/lti/v1/message#>. For example, the fully qualified URI for the `basic-lti-launch-request` is <http://purl.org/ims/lti/v1/message#basic-lti-launch-request>.

Formally, each message type is also a capability. The TC announces that it has the ability to send messages of a given type by listing the simple name of the message type within the capability set. **Error! Reference source not found.** illustrates an example of one such capability. Messaging capabilities do not need to be enabled explicitly within the Tool Proxy. The existence of a handler in the Tool Profile for a given message type implies that the TP wishes to enable the capability.

3.3.3 Outcomes Capabilities

Suppose that a Tool implements an assessment instrument such as a test, quiz, homework assignment, etc. The Tool may want to send results from the assessment activity back to the Tool Consumer. This exchange can be accomplished through the use of the Result Service described in Section 11.2. However, to use the Result Service the Tool needs an identifier for a Result object.

A Tool Consumer may have the ability to auto-create a Result object and pass its identifier in a message that launches the Tool. The TC announces this capability in the Tool Consumer profile as shown in Figure 12.

The Tool Provider enables this capability within the scope of one or more specific Message Handlers that appear in the Tool Profile. See Section 3.4.4 for more information about enabling capabilities.

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "@subject" : "urn:lti:ToolConsumerProfile:alpha.university.edu",
  "product_instance" : { },
  "service_offered" : [ ],
  "capability" : [
    "Result.autocreate",
    ...
  ]
}
```

Figure 12 The Result.autocreate Capability

The `Result.autocreate` capability implies the following rules:

1. The TC can associate individual LTI Links with a Message Handler defined in a Tool Profile. (This is required for LTI 1.2 conformance.)
2. The TC can associate individual LTI Links with an LIS LineItem.
3. Suppose the `Result.autocreate` capability is enabled by a given Message Handler.
 - a. The TC agrees to create a new LineItem and associate it with the LTI Link prior to launching the Tool (provided that the link is not already associated with a LineItem).
 - i. The title of the new LineItem shall match the title of the LTI Link.
 - ii. The dataSource GUID of the LineItem shall be set equal to the Tool Proxy GUID.
 - b. Suppose the link is activated by a Learner.
 - i. Prior to launching the Tool, the TC agrees to create a new Result object associated with both the LineItem and the Learner (provided that the Learner does not already have a Result defined for the LineItem). The new Result object shall be initialized with a null `resultScore`, and the dataSource GUID shall match the Tool Proxy GUID.

- ii. If the Learner already has a Result and the `resultScore` is not null, the TC will not allow the Learner to navigate into the Tool via the LTI Link. Thus, in LTI 1.2, each Learner is permitted at most one attempt on the assignment.
- iii. The TC will include the sourcedid for the Result as a custom parameter in the launch request in accordance with the parameter template defined by the Message Handler.

3.4 Tool Profile

```
{
  "product_instance" : { },
  "service_offered" : [ ],
  "base_url_choice" : [ ],
  "message" : [ ],
  "resource_handler" : [ ]
}
```

Figure 13 ToolProfile synopsis

A Tool Profile contains information about one particular instance of a Tool product. In addition to the common elements discussed in Section 3.1, the Tool Profile defines handlers for receiving messages from the Tool Consumer.

Certain message types can have at most one handler. The handlers for such messages are called *singletons*. Other message types can have multiple handlers which are nested within a so-called *Resource Handler*.

3.4.1 Singleton message handlers

The example in Figure 14 shows a singleton handler for two different types of messages:

- ToolProxyDeploymentRequest
- ToolProxyReregisterRequest

As this example illustrates, a Tool Provider may choose to expose a single endpoint to handle multiple message types. In the example, the Tool Profile declares a single handler for two message types at the same endpoint – <https://acme.example.com/Lti/registration>. This handler can receive either a ToolProxyDeploymentRequest or a ToolProxyReregistrationRequest.

```
{
  "product_instance" : { },
  "service_offered" : [ ],
  "base_url_choice" : [ ],
  "message" : [
    { "message_type" : [ "ToolProxyDeploymentRequest",
                        "ToolProxyReregistrationRequest" ],
      "path" : "https://acme.example.com/Lti/registration",
      "parameter" : [
        {
          "name" : "lis_person_name_given",
          "variable" : "$Person.name.given"
        }
      ]
    }
  ],
  "resource_handler" : [ ]
}
```

Figure 14 Singleton message handlers

A message handler may define a template for the custom parameters that are expected for all messages delivered to that handler. Parameter templates are discussed in Section 3.4.3.

3.4.2 Resource Handlers

A Tool may expose many different kinds of resources that can be launched from the Tool Consumer. For example, a given Tool might expose tutorials, homework assignments, quizzes, simulators, etc. Each type of resource may potentially have a different endpoint in the Tool Provider. Furthermore, each type of resource may require a different set of parameters when it is launched. The Tool Profile introduces the concept of a Resource Handler to capture these variations.

For each type of resource, the Tool Profile may declare a distinct Resource Handler, as illustrated by the example in Figure 15. The Resource Handler has a *code* that identifies the type of resource that is being made available through the LTI integration. This code must be unique within the associated *product_family* (see Section 3.1.2).

The Resource Handler has a *name* that is suitable for display to end users. The Tool Consumer MAY choose to display this name to content builders who are authoring links within the Tool Consumer user interface. The Resource Handler also has an optional description which the Tool Consumer might display as well.

When designing a Tool Profile, the Tool Provider must decide how it wants to expose its resources. It may choose to use a single Resource Handler for all resources. However, if different resources are accessed through different endpoints, then multiple Resource Handlers will be necessary. Each Resource Handler defines its own Message Handler for the *basic-lti-launch-request* message.

Furthermore, if different resources require different sets of parameters, then it may be useful to define separate Resource Handlers – even if they all have the same endpoint – because each Resource Handler can specify its own template for parameters. Parameter templates are discussed in the next section.

Finally, each Resource Handler may enable a different set of capabilities, as discussed in Section 3.4.4.

```
{
  "resource_type" : "urn:lti:ResourceType:acme.example.com/nitrolab/homework",
  "name" : {
    "default_value" : "Acme Homework Assignment"
  },
  "description" : {
    "default_value" : "A homework assignment related to a chapter in your textbook"
  },
  "message" : [
    {
      "message_type" : ["basic-lti-launch-request"],
      "path" : "resource/homework",
      "parameter" : [
        { "name" : "discipline",
          "fixed" : "chemistry"
        },
        { "name" : "lis_person_name_given",
          "variable" : "$Person.name.given"
        },
        { "name" : "textbook_isbn" },
        { "name" : "chapter" }
      ],
      "capability" : ["Resource.autocreate"]
    }
  ]
}
```

Figure 15 Example Resource Handler

3.4.3 Parameter templates

Each Message Handler within a Tool Profile may declare a *parameter template* that specifies the custom parameters that should be included in all messages sent to that handler. Figure 15 shows an example of a template that contains five custom parameters.

Each parameter within the template has a name. When a custom parameter is rendered in a `basic-lti-launch-request`, the HTTP POST binding of the parameter includes the prefix “`custom_`” as discussed in Section 2.2. Thus, the parameters listed in Figure 15 will have the following names in the HTTP POST binding:

- `custom_discipline`
- `custom_lis_person_name_given`
- `custom_textbook_isbn`
- `custom_chapter`
- `custom_result_id`

Each parameter in the template may have a fixed value, a variable value, or no value specified. If the value is fixed, then every message sent to the Message Handler must include the specified parameter with the exact literal value that is defined in the Tool Profile. If the value is a variable, the TC SHOULD substitute the variable with the appropriate value at runtime when it constructs a message. If the TC is unable to expand the variable, then it MUST supply the name of the variable as the parameter value in the message. If no value is specified in the Tool Profile, then a content builder needs to specify a value when authoring the link.

The use of parameter templates is optional. However, if the parameter template is empty, the content builder is responsible for defining all parameters without any guidance.

3.4.4 Enabled Capabilities

A Message Handler may declare a set of capabilities that it wishes to enable for messages sent to the given endpoint. In LTI 1.2, there is only one relevant capability, namely the `Result.autocreate` capability, which was discussed in Section 3.3.3. The Message Handler enables a capability merely by listing it as shown in Figure 15.

3.4.5 Base URLs

Within a Tool Profile, the endpoint for a `basic-lti-launch-request` is defined relative to a certain base URL.

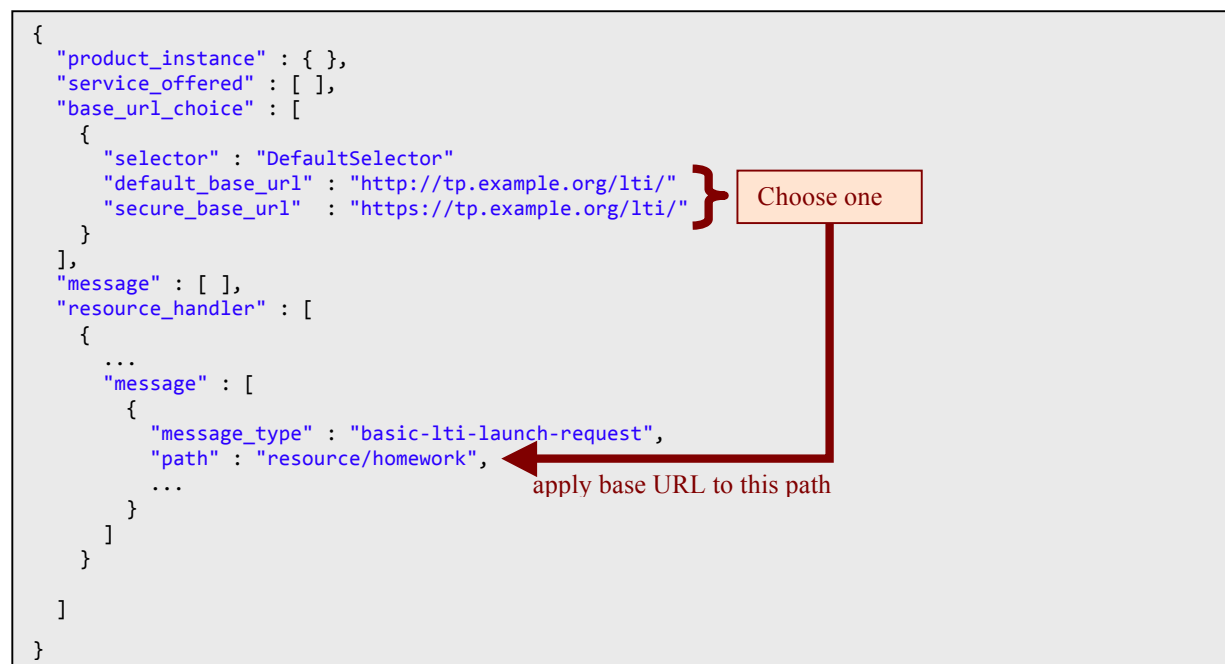


Figure 16 Base URL Choice

The `base_url_choice` object defines two options for the base URL: (1) a default base URL and (2) a secure base URL.

The Tool Consumer applies the following rules to choose the correct base URL:

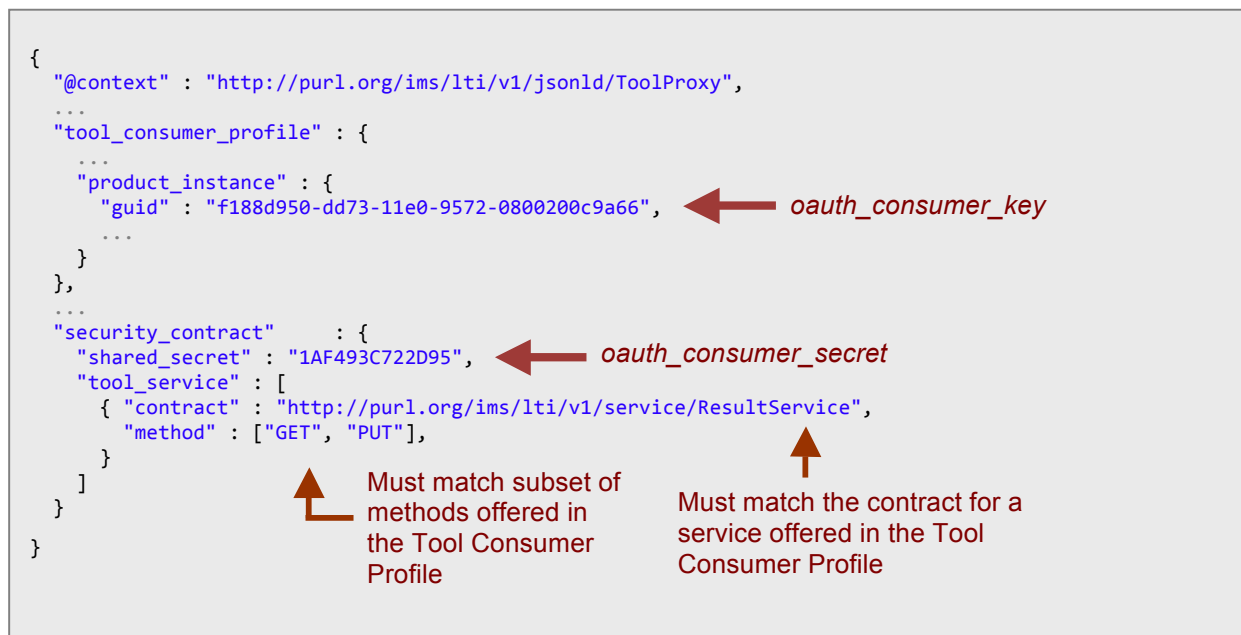
1. If the Tool is being launched from a secure web page in the TC system (i.e. protected by SSL) and the `secure_base_url` is defined, then use that value for the base URL.
2. Otherwise, use the `default_base_url`.

These rules ensure that end users won't see the "mixed content" warning in circumstances where the Tool resource is embedded in an iframe within the Tool Consumer user interface.

The `secure_base_url` is optional; it is omitted if the TP does not support SSL. By contrast, the `default_base_url` is a required element.

If the Tool Provider wants all of its resources to be protected by SSL, it should use the https protocol in the `default_base_url` and leave the `secure_base_url` undefined.

3.5 The Security Contract

**Figure 17 Example Security Contract**

The security contract contains two key pieces of information:

1. A shared secret
2. The subset of services offered by the TC which the TP intends to use.

The example in Figure 13 shows that the TP intends to use just one service offered by the TC, namely the Result Service. In general, the TP may use as many services as the TP offers. The `contract` element within the `tool_service` must match exactly the contract for a service that is offered in the Tool Consumer Profile. The TP also declares exactly which methods on the service it intends to use; the set of methods declared here must be a subset of the method offered in the Tool Consumer Profile.

The shared secret is used to digitally sign launch requests in accordance with the OAuth protocol and to authenticate service calls via HTTP Basic Access Authentication (RFC 1945).

When launching a Tool via the Tool Proxy, the value of the `oauth_consumer_key` is given by the GUID for the Tool Consumer instance, and the `oauth_consumer_secret` is given by the `shared_secret` element within the `security_contract`.

When invoking a service like the Result Service, the Tool must use HTTP Basic Access Authentication to authenticate itself to the TC.

4 Link Authoring

When an LTI link is created it must be associated with one of four possible entities:

1. Link Level Credentials
2. TC Wide Domain Credentials
3. TC Wide URL Credentials
4. An LTI Resource Type

TODO: Insert discussions about the first three approaches here. (Move it out of the Appendix.)

TODO: Should we discuss the concept of a Tool Proxy Binding?

TODO: Add recommendations for associating links with a Resource Type if they were originally created using one of the first three methods (Link Level Credentials, TC Wide Domain Credentials, TC Wide URL Credentials). In principle, it's pretty simple – construct the fully qualified URL for all basic-lti-launch-request handlers defined in the Tool Proxies, and then match the URL in the LTI links against those constructed URLs. If a match is found, then create the association.

4.1 Authoring links associated with a Resource Type

Once a collection of Tool Proxies have been registered, the Tool Consumer must expose an interface for creating links associated with the Resource Handlers defined in those Tool Proxies. The LTI standard does not prescribe a user experience for creating links, but a typical implementation might present the user with a list box like the one illustrated schematically in Figure 17. In this example, the labels in the list box correspond to the *name* attributes of the various Resource Handlers. The TC might display the Resource Handler *description* as a tooltip when the user's mouse hovers over a selection.

When an LTI Link is created in this way, the TC must associate the new link with the selected *Resource Type*.

Each Resource Type is uniquely identified by three values:

1. Vendor Code
2. Product Code
3. Resource Code

The vendor code is globally unique, the product code is unique within the namespace of the vendor, and the resource code is unique within the namespace of the product family.

Thus, each Resource Type can be unambiguously identified by a URN of the form.

Link To
Acme Chemlab Homework
Acme Chemlab Redox Simulator
Beta Quiz
Gamma Lesson Plan
eBook from Delta Publishing

Figure 17 List Box for creating links

`urn:ims:lti/ResourceType/?vendorCode/?productCode/?resourceCode`

(URI 4-1)

A URN of this form should be defined as a property of any LTI Link that points to a resource of the specified type.

Given a URN for a Resource Type, it is possible to discover the corresponding Resource Handler by applying the following algorithm:

1. Parse the *vendorCode*, *productCode* and *resourceCode* from the URN.
2. Find the Tool Proxy whose Tool Profile contains the given *vendorCode* and *productCode*.
3. Within the Tool Profile, find the Resource Handler for the specified Resource Type.

This procedure is illustrated in Figure 18.

When launching a Tool via an LTI Link that is associated with a Resource Type, the Tool Consumer must construct the appropriate endpoint URL from the Resource Handler as discussed in Section 3.4.5.

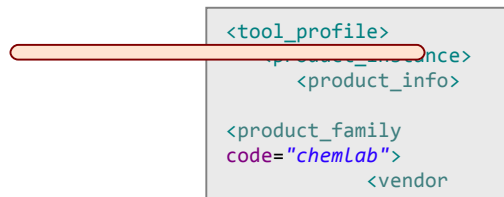


Figure 18 Discovering the Handler for a Resource Type

5 Establishing an Integration Contract

The process of establishing an integration contract involves three key steps.

1. **Request access to a Tool.** A Tool Consumer administrator requests access to a Tool.
2. **Fetch Tool Consumer Profile.** The Tool fetches a Tool Consumer Profile from the Tool Consumer system and confirms that The Tool Consumer satisfies the minimum requirements for a successful integration.
3. **Register a ToolProxy.** The Tool creates a so-called ToolProxy and registers it with the Tool Consumer system.
4. **Make the ToolProxy Available.**

The ToolProxy effectively defines the integration contract between the ToolConsumer and the Tool.

Let's examine the above steps in some detail.

5.1 Request access to a Tool

To embed a new Tool within a Tool Consumer system, the TC administrator submits a so-called Tool Proxy Deployment Request. The TC administrator must know the web address (URL) to which this request should be submitted within the TP system. The LTI specification does not say how the TC administrator discovers this URL. Typically, the URL will be published somewhere, or the administrator might receive it in an email from the Tool's vendor or a service provider.

The TC administrator enters this URL into a user interface exposed by the TC application. In response, the TC constructs a Tool Proxy Deployment Request which is returned to the user's browser as a collection of hidden HTML Form parameters, and the form is auto-submitted to the specified URL.

Figure 19 illustrates a sample Tool Proxy Deployment Request generated by the TC application after the TC administrator enters the URL "https://acme.example.com/lti/deployment".

```

<form
  name="LtiRequest"
  action="https://acme.example.com/lti/deployment"
  method="post"
  enctype="application/x-www-form-urlencoded">

  <input type="hidden" name="lti_message_type" value="ToolProxyDeploymentRequest" />
  <input type="hidden" name="user_id" value="0291739133" />
  <input type="hidden" name="roles" value="SysAdmin" />
  <input type="hidden" name="tc_profile_url" value=
    "http://coyote.example.biz/lti/ToolConsumerProfile/72942860da8611e095720800200c9a66"/>

  <input type="hidden" name="reg_password"
    value="5622cdc0da8211e095720800200c9a66"/>

  <input type="hidden" name="launch_presentation_return_url"
    value="http://coyote.example.biz/managetools" />
</form>
<script>
  document.LtiRequest.submit();
</script>

```

The URL submitted by the TC Administrator.

The URL to which the administrator returns after completing the workflow in the Tool Provider system.

Figure 19 Sample Tool Proxy Deployment Request

See Section 2.4 for a description of the fields in the Tool Proxy Deployment Request.

5.2 Fetch Tool Consumer Profile

TODO: Write this section

5.3 Register a Tool Proxy

TODO: Write this section

5.4 Make the Tool Proxy Available

6 Use Cases

This section is taken verbatim from the LTI Tool Management [LTI, 10, TMT] documentation.

Use Case Title:	Deploying a Tool Proxy via the Tool Proxy Request Form
Use Case Local ID:	LTIv1-02
Brief Description:	This use case extends the Basic Tool Proxy Deployment process by defining one way to initiate the deployment workflow. In this use case, an administrator initiates the deployment of a Tool Proxy by submitting the Tool Proxy Request Form.
Level:	Summary
Actors:	Tool Consumer Administrator, Tool Consumer system
Basic Flow of Events:	<ol style="list-style-type: none"> 1. <u>Obtain Tool Deployment URL</u>. The administrator obtains a Deployment URL from the Tool Provider. This is the URL that will be used to launch the Deployment UI. The LTI standard does not define how the administrator acquires the Deployment URL. It may be delivered via email, or by any other means. 2. <u>Submit Tool Proxy Request Form</u>. The Tool Consumer implements a Tool Proxy Request Form which contains an input field where the administrator can enter the Deployment URL. The administrator navigates to this form within the Tool Consumer, enters the Deployment URL, and submits the form. 3. <u>Generate ToolProxyDeploymentRequest</u>. The Tool Consumer responds to the submission of the Tool Proxy Request Form by generating a ToolProxyDeploymentRequest. In this use case, the <code>vendor_code</code>, <code>tool_code</code>, and <code>tool_version</code> attributes of the request are not defined. In accordance with the LTI Messaging Framework [LTI, 10 MSF], the ToolProxyDeploymentRequest is rendered as an HTML form and auto-submitted to the Tool Provider. The value for the <code>action</code> attribute of the form is given by the Deployment URL that was submitted in Step 2. Thus, when the form is auto-submitted, it will be posted to the Tool Provider at the Deployment URL. 4. <u>Use Basic Tool Proxy Deployment Process</u>. The remainder of this use case follows the Basic Tool Proxy Deployment sequence, see Section Error! Reference source not found. Since the <code>tool_code</code> and <code>tool_version</code> are not defined in the ToolProxyDeploymentRequest, the Tool Provider will need to identify the particular Tool Proxy that is being requested by some other means. The Tool Provider may use either of the following methods: <ul style="list-style-type: none"> • The Tool Provider may infer the Tool Proxy from the Deployment URL. In other words, the Tool Provider might use a different Deployment URL for each version of the Tool Proxy. • Alternatively, the Tool Provider might require the administrator to choose a particular Tool Proxy as one of the activities that occurs within the Deployment UI. (See Step 3 of the Tool Proxy Deployment use case.)
Alternative Flows:	<p>A. Customized Tool Consumer Profile</p> <p>At Step 2, the Tool Proxy Request Form may allow the TC Administrator to limit the set of services and capabilities that will be offered to the Tool Provider. In this case, the Tool Consumer generates a customized Tool Consumer Profile which advertises only the selected services and capabilities. In this case, the</p>

	ToolProxyDeploymentRequest generated at Step 3 will contain the URL for the customized Tool Consumer Profile.
--	---

Use Case Title:	Updating a Tool Proxy within a Tool Consumer
Use Case Local ID:	LTiv1-11
Brief Description:	Tool Consumer Administrator applies updates to a Tool Proxy that is currently registered with the Tool Consumer Runtime. Note that the activities in this use case are similar to the activities that occur in Basic Tool Deployment (Use Case LTiv1-01).
Level:	Summary
Actors:	<ul style="list-style-type: none"> • Tool Consumer Administrator • Tool Consumer Runtime (TC) • Tool Provider (TP)
Preconditions:	<ul style="list-style-type: none"> • Tool Proxy has been registered with the Tool Consumer • Tool Consumer Administrator has been alerted that an update is available.
Basic Flow of Events:	<ol style="list-style-type: none"> 1. <i><u>Initiate the update process.</u></i> TC Administrator performs an action within the Tool Console to initiate the update process for a selected Tool Proxy. 2. <i><u>Send HTML Form.</u></i> The TC responds by sending an HTML form that contains a ToolProxyReregistrationRequest to the administrator's browser (a.k.a. User Agent). 3. <i><u>Post ToolProxyReregistrationRequest.</u></i> The User Agent auto-submits the HTML form and thereby posts the ToolProxyReregistrationRequest to the Tool Provider. 4. <i><u>Get Tool Consumer Profile.</u></i> The TP retrieves the Tool Consumer Profile from the URL specified by the <code>tc_profile_url</code> parameter. 5. <i><u>Interact with Administrator.</u></i> The TP may optionally interact with the TC Administrator. The LTI standard does not require any particular interactions, but typical activities may include: <ol style="list-style-type: none"> a. Choosing (or confirming) which version of the tool should be used. b. Changing the set of tool features that should be enabled. c. Collecting payment (if the license has expired). 6. <i><u>Submit.</u></i> When all interactions are complete, the Administrator makes a final submission to the TP. 2. <i><u>Validate Tool Consumer Profile.</u></i> The TP validates the Tool Consumer profile that was obtained in Step 4 to ensure that the services and capabilities offered satisfy the requirements of the tool. 3. <i><u>Invoke Tool Registration Service.</u></i> TP invokes the <code>reregisterTool</code> operation on the Tool Registration Service implemented by the TC. As a best practice, the <code>ToolRegistrationRequest</code> submitted to the TC should contain a new shared secret. <p>The TC persists the changes to the Tool Proxy in accordance with the <code>ToolRegistrationRequest</code>. In particular, any new Menu Links declared in the Tool Profile should appear in all learning contexts that are bound to the Tool Proxy. Obsolete Menu Links dropped from the Tool Profile should disappear from those learning contexts. However, if the security contract has changed, then the Tool Proxy should be put into the <i>unavailable</i> state immediately after reregistration. In this case, changes to the Menu Links will appear only after the Tool Proxy is made</p>

	<p>available again (see Step 10).</p> <p>4. <u>Redirect back to Tool Consumer</u>. TP redirects the Administrator's browser back to the TC at the URL specified by the <code>launch_presentation_return_url</code> parameter from the <code>ToolProxyReregistrationRequest</code> posted at Step 3. To this URL, the Tool Provider appends the following HTTP query parameters:</p> <pre>status = success tool_guid = <globally unique identifier for the Tool Proxy></pre> <p>Typically, this redirect action returns the user to the Tool Console within the TC system where the updated Tool Proxy can be made available (see the next step).</p> <p>5. <u>Make Tool Proxy Available</u>. If the security contract has changed, the TC displays the new set of entitlements that will be granted to the Tool Provider, and it presents the administrator with an option to make the Tool Proxy available again.</p>
Alternative Flows:	<p>A. No state change</p> <p>If the security contract has not changed, then the state of the Tool Proxy should not change at Step 8. In particular, if the Tool Proxy was originally available, it will still be available after reregistration, and in this case Step 10 can be skipped.</p>

Title:	Setting TP Domain Credentials (Basic LTI)
Use Case Local ID:	LTIv1-12
Description:	The TC administrator configures TP domain credentials for a particular TP.
Actors:	<ul style="list-style-type: none"> • TC Administrator • TP Administrator
Preconditions:	None
Basic Flow of Events:	<ol style="list-style-type: none"> 1. The TP Administrator creates the key and secret combination for the TC (where the TC administrator may request a particular key, often the TC domain name). 2. The TC Administrator obtains the key and secret from the TP administrator. 3. The TC Administrator installs the key and secret and associates them with launches destined for the TP's domain using a TC-provided dialog or configuration mechanism.
Alternate Path	A. In step 3, the key and secret combination may be designated to apply to a particular TP URL instead of all URLs destined for the TP's domain.

Title:	Setting Link Level Credentials (Basic LTI)
Use Case Local ID:	LTIv1-13
Description:	An instructor authors a Basic LTI link and sets the key/password for that link.
Actors:	<ul style="list-style-type: none"> • Instructor • Tool Provider (TP)
Preconditions:	None
Basic Flow of Events:	<ol style="list-style-type: none"> 1. The Instructor contacts the TP and gains access to a provider tool or content. 2. The TP provides the Instructor with (1) a Basic LTI launch URL or XML snippet for the content or tool, (2) a key used to access this content/tool, and (3) a secret

	associated with the key.
	3. The Instructor enters the three values (URL or XML snippet, Key, Secret) into a Basic LTI authoring dialog in the TC system.

Title:	Launching an Authored Basic LTI Link from a Context
Use Case Local ID:	LTIv1-14
Description:	A non-Instructor user selects a Basic LTI link from a context in the TC.
Actors:	<ul style="list-style-type: none"> • TC User
Preconditions:	An Instructor has properly authored or imported a Basic LTI Link and there are appropriate credentials in place.
Basic Flow of Events:	<ol style="list-style-type: none"> 1. The TC User clicks on the link in the TC UI. 2. The tool or content from the TP appears in the TC UI. If JavaScript is turned off – the TC User will need to click on a "continue" button to send the POST data to the TP.

Title:	Launching Basic LTI Imported from a Cartridge (with secret)
Use Case Local ID:	LTIv1-15
Description:	An Instructor imports a Common Cartridge containing Basic LTI link descriptors into their context and users use the content.
Actors:	<ul style="list-style-type: none"> • Cartridge Creator • Instructor • TC User
Preconditions:	The TC Administrator has received and installed the appropriate credentials for the particular TP(s) that are referenced in the cartridge.
Basic Flow of Events:	<ol style="list-style-type: none"> 1. The Cartridge Creator authors a cartridge and includes one or more Basic LTI link descriptors in the cartridge. The Basic LTI link descriptors in the cartridge contain a launch URL(s) and other data but do not contain keys or secrets. 2. The Instructor obtains the Common Cartridge and imports it into their context in the TC system. 3. When a TC User launches the Basic LTI link imported from the Common Cartridge, the TC uses the pre-configured credentials associated with the TP domain or URLs. 4. In particular, as long as the TC-wide credentials are already installed, the Instructor does not need to take any further action to launch the content beyond importing a cartridge.
Alternate Path	A. If the preconditions are not satisfied, it is also possible to set the credentials after the cartridge import has taken place. If a launch occurs before credentials have been defined, it is the responsibility of the TP to notify the user that credentials are required.

Title:	Launching Basic LTI Imported from a Cartridge (no secret)
Use Case Local ID:	LTIv1-16
Description:	An Instructor imports a Common Cartridge containing Basic LTI link descriptors into their context and users use the content. Note that this scenario is optional – the TC and/or TP may decide that Basic LTI launches without secrets are treated as an error.

Title:	Returning a decimal score (0.1-1.0) from the TP to the TC
Use Case Local ID:	LTI-IGv1-01
Description:	An instructor creates a resource link in a course and indicates that it is to receive grades from the TP and sets up any necessary routing between the LTI link and the grade book. Whether or not the TC accepts grades for a particular user/context/link is up to some combination of the TC Admin and TC Instructor.
Actors:	<ul style="list-style-type: none">• TC Admin• Instructor• TC User
Preconditions:	None.
Basic Flow of Events:	<ol style="list-style-type: none">1. The TC admin enables support for grade routing in the TC2. The Instructor authors a link and indicates that the link will accept grades and sets up any needed routing for the received grades within the TC.3. The TC user launches the link and the TC includes necessary service endpoints and callback data so as to allow the TP to make service calls to set, read, and delete grades.4. The TP system makes calls to the TC outcomes services to set, read, and/or delete grades as needed. The setting of grades can be done at any time as it is a server to server to trust (i.e. not just during the launch period).

7 LTI Message Items

This section describes the data items that are passed as part of the POST data when a message is sent from the TC to the TP. Currently there are two messages sent from the TC to the TP currently defined in this implementation guide: ToolProxyDeploymentRequest and Basic LTI Launch Request.

First we will define the common fields used across all messages, and then look at the required and optional elements in each of the two messages.

If a TC wants to send any additional or proprietary fields, they should prefix all fields not described herein with "ext_".

lti_message_type=basic-lti-launch-request | ToolProxyDeploymentRequest | ToolProxyReregistrationRequest

This indicates the type of the message. This allows a TP to accept a number of different LTI message types at the same launch URL. This parameter is required.

lti_version=LTI-1p0

This indicates which version of the specification is being used for this particular message. This parameter is required.

user_id=0ae836b9-7fc9-4060-006f-27b2066ac545

Uniquely identifies the user. This should not contain any identifying information for the user. Best practice is that this field should be a TC-generated long-term "primary key" to the user record – not the "logical key". This parameter is recommended

roles=Instructor

A comma-separated list of URN values for roles. If this list is non-empty, it should contain at least one role from the LIS System Role, LIS Institution Role, or LIS Context Role vocabularies (See Appendix A). The assumed namespace of these URNs is the LIS vocabulary of LIS Context Roles so TCs can use the handles when the intent is to refer to an LIS context role. If the TC wants to include a role from another namespace, a fully-qualified URN should be used. Usage of roles from non-LIS vocabularies is discouraged as it may limit interoperability. This parameter is recommended.

launch_presentation_locale=en-US

Language, country and variant as represented using the IETF Best Practices for Tags for Identifying Languages (BCP-47) available at <http://www.rfc-editor.org/rfc/bcp/bcp47.txt>

launch_presentation_document_target=iframe

The value should be either 'frame', 'iframe' or 'window'. This field communicates the kind of browser window/frame where the TC has launched the tool. This parameter is recommended.

launch_presentation_css_url=

This is a URL to an LMS-specific CSS URL. There are no standards that describe exactly what CSS classes, etc. should be in this CSS. The TP should send its standard CSS URL that it would apply to its local tools. The TC should include styling for HTML tags to set font, color, etc and also include its proprietary tags used to style its internal tools.

Someday perhaps we will come up with a cross-LMS standard for CSS classes to allow a tool to look "built-in" with only one set of markup, but until that happens, the **launch_presentation_css_url** allows tools a chance to adapt their look and feel across LMS systems to some degree.

launch_presentation_width=320

The width of the window or frame where the content from the tool will be displayed. This parameter is recommended.

launch_presentation_height=240

The height of the window or frame where the content from the tool will be displayed. This parameter is recommended.

launch_presentation_return_url=http://lmsng.school.edu/portal/123/page/988/

Fully qualified URL where the TP can redirect the user back to the TC interface. This URL can be used once the TP is finished or if the TP cannot start or has some technical difficulty. In the case of an error, the TP may add a parameter called **lti_errormsg** that includes some detail as to the nature of the error. The **lti_errormsg** value should make sense if displayed to the user. If the tool has displayed a message to the end user and only wants to give the TC a message to log, use the parameter **lti_errorlog** instead of **lti_errormsg**. If the tool is terminating normally, and wants a message displayed to the user it can include a text message as the **lti_msg** parameter to the return URL. If the tool is terminating normally and wants to give the TC a message to log, use the parameter **lti_log**. This data should be sent on the URL as a GET – so the TP should take care to keep the overall length of the parameters small enough to fit within the limitations of a GET request. This parameter is recommended.

7.1 LTI ToolProxyDeploymentRequest and ToolProxyReregisterRequest Message

These requests start with the Tool Consumer administrator navigating in the TC user interface to an "Add New Proxy Tool" or "Reregister Proxy Tool" user interface. The TC admin enters the URL of the Tool Provider's provisioning service into that user interface and then the Tool Consumer launches the Tool provider via a signed message (1). Then the Tool Provider retrieves the Tool Consumer Profile (2) with a GET request/. Then the Tool Provider optionally (3,4) interacts with the TC Administrator in the iframe/window possibly prompting for unlock codes or even performing some sort of eCommerce. At the point where the Tool Provider (5,6) decides to install the Tool Proxy in the Tool Consumer, it calls the registerTool operation in the Tool consumer using the secret provided in the original ToolProxyDeploymentRequest. When the registerTool operations completes successfully, the TP (7) redirects the TC Admin browser back to a URL of the TC's choosing where the TC typically will complete the process of (8) activating the Proxy tool.

There are many different paths through the logic depending on whether errors are encountered or the Tool Producer chooses not to call the registerTool operation, but the common flow is shown below.

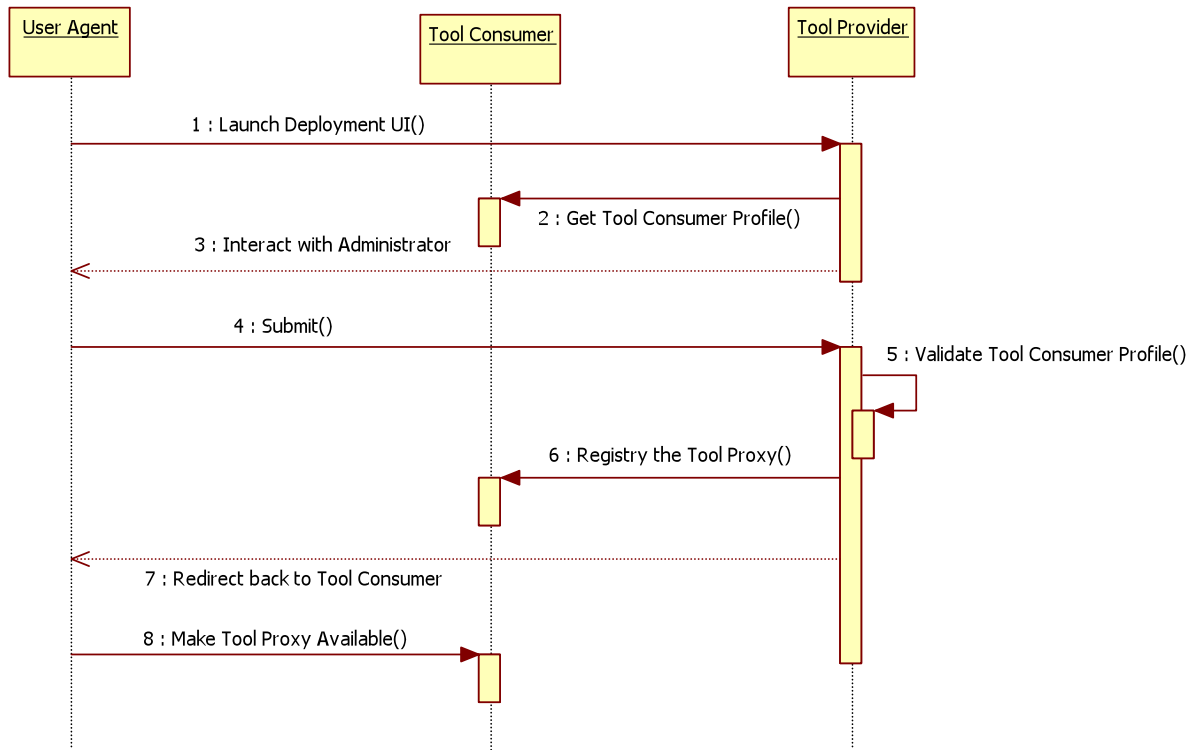


Figure 20 Flow chart for basic tool proxy deployment.

lti_message_type=ToolProxyDeploymentRequest | ToolProxyReregisterRequest

This indicates the type of the message. This parameter is required.

Required Message Data Items: **user_id, roles**

Optional Message Data Items: **launch_presentation_locale, launch_presentation_css_url, launch_presentation_document_target, launch_presentation_window_name, launch_presentation_width, launch_presentation_height**

reg_password=918283799228bbejsdh999288

This is the secret used to sign the RegisterToolRequest when the TP performs the operation. This will typically be a single use secret and only can be used in the context of this particular ToolProxyDeploymentRequest message. This parameter is required.

tc_profile_url=http://lmsng.school.edu/ims/lti/tool_consumer_profile.xml

This URL contains the URL where the TP can retrieve the TC tool profile. This URL must be retrievable by a GET request by the TP. If the URL is protected from retrieval in general, the TC must append necessary parameters to allow the TP to retrieve the URL with nothing more than a GET request. It is legal for this URL to contain a security token that is changed for each ToolProxyDeploymentRequest so the TP must retrieve the tc_profile on each request. This parameter is required.

launch_presentation_return_url=http://lmsng.school.edu/admin/continue_proxy.php

The Tool Provider redirects the user's browser back to this URL when the TP has completed the tool deployment process. The TP should redirect to this URL regardless of whether the deployment process succeeds or fails. In addition to the log and message parameters, described above, the Tool Provider appends the following HTTP query parameters

status=success | failure

tool_guid=<globally unique identifier for the Tool Proxy>

where the value for the tool_guid parameter is given by the return value from the registerTool operation if the operation was successful. Typically, this action redirects the administrator's browser to the Tool Console within the Tool Consumer system where the Tool Proxy can be made available. This parameter is required.

7.2 LTI Launch Message

Very few of the fields are technically required as each Tool Provider may have different requirements. Some TPs may see the fields in the launch as information to be gathered for tracking and others may need highly detailed and precise information to perform high-stakes activities and reliably and securely return high-stakes results from those activities.

TC systems should provide as much data as possible in each launch to maximize the chance that the TP will have the data it needs to function properly. TC systems may have sandboxing features that limit the sending of certain Basic LTI data elements only to "approved" TPs. It is outside the scope of the document to define the nature of the TC sandboxing of Basic LTI launches. TPs should be prepared to work with partial information – either because the TC does not have the information or the TC has been configured not to share the information with the TP.

Required Message Data Items: **user_id, roles**

Optional Message Data Items: **launch_presentation_locale, launch_presentation_css_url, launch_presentation_document_target, launch_presentation_window_name, launch_presentation_width, launch_presentation_height**

lti_message_type=basic-lti-launch-request

This indicates that this is a Basic LTI Launch Message. This parameter is required.

context_id=8213060-006f-27b2066ac545

This is an opaque identifier that uniquely identifies the context that contains the link being launched. This parameter is recommended.

context_type=CourseSection

This string is a comma-separated list of URN values that identify the type of context. At a minimum, the list MUST include a URN value drawn from the LIS vocabulary (see Appendix A). The assumed namespace of these URNs is the LIS vocabulary so TCs can use the handles when the intent is to refer to an LIS context type. If the TC wants to include a context type from another namespace, a fully qualified URN should be used. This parameter is optional.

context_title=Design of Personal Environments

A title of the context – it should be about the length of a line. This parameter is recommended.

context_label=SI182

A label for the context – intended to fit in a column. This parameter is recommended.

resource_link_id=88391-e1919-bb3456

This is an opaque unique identifier that the TC guarantees will be unique within the TC for every placement of the link. If the tool / activity is placed multiple times in the same context, each of those placements will be distinct. This value will also change if the item is exported from one system or context and imported into another system or context. This parameter is required.

resource_link_title=My Weekly Wiki

A title for the resource. This is the clickable text that appears in the link. This parameter is recommended.

resource_link_description=...

A plain text description of the link's destination, suitable for display alongside the link. Typically no more than several lines long. This parameter is optional.

lis_person_name_given=Jane**lis_person_name_family=Public****lis_person_name_full=Jane Q. Public****lis_person_contact_email_primary=user@school.edu**

These fields contain information about the user account that is performing this launch. The names of these data items are taken from LIS. The precise meaning of the content in these fields is defined by LIS. These parameters are recommended unless they are suppressed because of privacy settings.

user_image=http://...

This attribute specifies the URI for an image of the user who launched this request. This image is suitable for use as a "profile picture" or an avatar representing the user. This parameter is optional.

lis_person_sourcedid=school.edu:user

This field contains the LIS identifier for the user account that is performing this launch. The example syntax of "school:user" is not the required format – **lis_person_sourcedid** is simply a globally unique identifier (i.e., a normalized string). This field is optional and its content and meaning are defined by LIS.

lis_course_offering_sourcedid=school.edu:SI182-F08**lis_course_section_sourcedid=school.edu:SI182-001-F08**

These fields contain LIS course identifiers associated with the context of this launch. These fields are optional and their content and meaning are defined by LIS.

lis_result_sourcedid=83873872987329873264783687634

This field contains an identifier that indicates the LIS Result Identifier (if any) associated with this launch. This field is optional and its content and meaning is defined by LIS.

custom_keyname=value

The creator of a LTI link can add custom key/value parameters to a launch which are to be included with the launch of the LTI link. The Common Cartridge section below describes how these parameters are represented when storing custom parameters in a Common Cartridge.

When there are **custom** name / value parameters in the launch, a POST parameter is included for each custom parameter. The parameter names are mapped to lower case and any character that is neither a number nor letter in a parameter name is replaced with an "underscore". So if a **custom** entry was as follows:

Review:Chapter=1.2.56

Would map to:

```
custom_review_chapter=1.2.56
```

Creators of Basic LTI links would be well served to limit their parameter names to lower case and to use no punctuation other than underscores.

If these custom parameters are included in the Basic LTI link, the TC must include them in the launch data or the TP may fail to function.

TC implementations may have the ability to make value substitutions for custom parameters as described in Appendix C. For example if a custom parameter was:

```
xstart=$CourseSection.timeFrame.begin
```

The parameter would be

```
custom_xstart=2012-04-21T01:00:00Z
```

Note that a DateTime data type in IMS LIS represents a combined date and time in the format of ISO 8601 i.e. 'YYYY-MM-DDThh:mm:ssTZD'. The time is denoted in Coordinated Universal Time (UTC) with TZD denoting the time zone offset in hours and minutes with respect to UTC.

It is important for a TP not to depend on the TC supporting any particular parameter substitution. If a TC that did not support parameter substitution were to see the above custom parameter, it would simply send

```
custom_xstart=$CourseSection.timeFrame.begin
```

as the parameter (i.e. send the parameter unsubstituted). It the responsibility of the TP to deal with both kinds of launched from TCs (i.e. with and without substitution available).

tool_consumer_instance_guid=lmsng.school.edu

This is a unique identifier for the TC. A common practice is to use the DNS of the organization or the DNS of the TC instance. If the organization has multiple TC instances, then the best practice is to prefix the domain name with a locally unique identifier for the TC instance. This parameter is recommended.

tool_consumer_instance_name=SchoolU

This is a user visible field – it should be about the length of a column. This parameter is recommended.

tool_consumer_instance_description=University of School (LMSng)

This is a user visible field – it should be about the length of a line. This parameter is optional.

tool_consumer_instance_url=http://lmsng.school.edu

This is the URL of the consumer instance. This parameter is optional.

tool_consumer_instance_contact_email=System.Admin@school.edu

An email contact for the TC instance. This parameter is recommended.

8 LTI Tool Consumer Profile

This is an example of the Tool Consumer's profile that is provided to the Tool Provider as part of the ToolProxyDeploymentRequest via the **tc_profile_url** parameter.

```
<?xml version="1.0" encoding="UTF-8"?>
<im:tool_consumer_profile
xmlns:imsltipc="http://www.imsglobal.org/xsd/imsltiPC_v1p0"
xmlns:imsltisec="http://www.imsglobal.org/xsd/imsltiSEC_v1p0"
xmlns:im="http://www.imsglobal.org/xsd/imsltiTCP_v1p0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imsltiTCP_v1p0
file:/Users/csev/Desktop/ims/full-
lti/LTI_LocalWSDLsXSDsvd5/xsd/imsltiTCP_v1p0.xsd" lti_version="lti_version0">
  <im:vendor>
    <imsltipc:code>bb-9</imsltipc:code>
    <imsltipc:name>Blackboard Learn 9</imsltipc:name>
    <imsltipc:url>http://www.blackboard.com/</imsltipc:url>
  </im:vendor>
  <im:tool_consumer_info>
    <imsltipc:code>school.edu</imsltipc:code>
    <imsltipc:name>LMSng</imsltipc:name>
    <imsltipc:version>BB9.1SP6</imsltipc:version>
  </im:tool_consumer_info>
  <im:tool_consumer_instance>
    <im:guid>lmsng.school.edu</im:guid>
    <im:name>SchoolU</im:name>
    <im:description>University of School (LMSng)</im:description>
    <im:contact>
      <imsltipc:email>System.Admin@school.edu</imsltipc:email>
    </im:contact>
  </im:tool_consumer_instance>
  <im:services_offered>
    <imsltipc:service_profile>
      <imsltipc:service id="00001" namespace="???" xsd="???"
name="ToolManagement" version="1.0" url="http://lmsng.school.edu/imslti/">
        <imsltipc:operation>registerTool</imsltipc:operation>
      </imsltipc:service_profile>
      <imsltipc:service_profile>
        <imsltipc:service id="00002" namespace="???" xsd="???"
name="IMSLIS" version="1.0" url="http://lmsng.school.edu/imslti/">
          <imsltipc:operation>replaceResult</imsltipc:operation>
          <imsltipc:operation>readResult</imsltipc:operation>
          <imsltipc:operation>deleteResult</imsltipc:operation>
        </imsltipc:service_profile>
      </im:services_offered>
    <im:security_profiles>
      <imsltisec:oauth_hash_message_security_profile>
        <imsltisec:applies_to>
          <imsltisec:message_ref type="basic-lti-launch-request"/>
          <imsltisec:message_ref type="ToolProxyDeploymentRequest"/>
        </imsltisec:applies_to>
        <imsltisec:algorithm>HMAC-SHA1</imsltisec:algorithm>
      </imsltisec:oauth_hash_message_security_profile>
      <imsltisec:hash_pox_oauth_security_profile>
        <imsltisec:applies_to>
```

```
<imsltipc:service_profile>
  <imsltipc:service id="00001"/>
  <imsltipc:operation>registerTool</imsltipc:operation>
</imsltipc:service_profile>
<imsltipc:service_profile>
  <imsltipc:service id="00002"/>
  <imsltipc:operation>replaceResult</imsltipc:operation>
  <imsltipc:operation>readResult</imsltipc:operation>
  <imsltipc:operation>deleteResult</imsltipc:operation>
</imsltipc:service_profile>
</imsltisec:applies_to>
<imsltisec:algorithm>HMAC-SHA1</imsltisec:algorithm>
</imsltisec:hash_pox_oauth_security_profile>
</im:security_profiles>
</im:tool_consumer_profile>
```

9 LTI Security Model

9.1 LTI Credential Management

This section is taken from the LTI Tool Management [LTI, 10 TMT] documentation.

For an LTI link that is set up via the tool registration request, the launch url, key, and secret are set in the registration request sent from the TP to the TC. The security environment for Basic LTI launches must be set up using out-of-band interactions between the TP administrator and either the TC administrator or an instructor who will be authoring a Basic LTI link.

There are three possible credentials (keys and secrets) associated with a particular LTI launch, listed in precedence order where TP domain credentials have the highest precedence.

- **Credentials associated with a TP domain.** These credentials authorize access to all TP URLs from the TC. Once the TP domain credentials are established for a TP, all LTI tool launches to the TP will use this same secret. Using TP domain credentials gives TPs the option of trusting user information and context information across multiple contexts within a particular TC instance as being maintained properly by the TC.

In order to select which TP domain credentials are used for a particular LTI link, the TC examines the domain name in the launch URL for the LTI link. The TP domain credentials are looked up after scanning the domain name of the launch URL. So for example, if the launch URL was:

```
http://launch.math.vendor.com/launch.php
```

The TC would prefer the following TP domain credentials in order from specific to general:

`launch.math.vendor.com`, `math.vendor.com`, and then `vendor.com`. So when TPs are generating link URLs and giving them to an instructor or embedding those links in a cartridge, it is important to use consistent domain names in those launch URLs so as to be able to match a TP domain credentials for a particular TP with the appropriate launches.

- **Credentials associated with a TP URL.** These credentials authorize access to a particular TP URL from the TC. These are typically used when the administrator is enabling a remote tool within the TC with a preset configuration that can be added to a context by the instructor with little or no further configuration.
- **Credentials associated with a particular link.** These credentials authorize access to the resource at the URL specified by the link. The instructor typically enters these credentials at the moment that the link is created in the context.

Basic LTI launches can happen from the TC with any combination of credentials. When more than one is present, the TC uses the highest precedent credentials to sign the request.

If there are no credentials available for this launch and the TC wants to perform the launch, the TC should not sign the launch data using OAuth. The TC can decide if it wants to send unsigned requests and the TP can decide if it wants to accept unsigned requests. A TC may also choose to treat the lack of credentials as an error and refuse to perform the launch.

9.2 OAuth Message Signing for x-www-form-encoded Messages

Note: This section is taken verbatim from the LTI Messages documentation.

OAuth is a security mechanism designed to protect POST and GET requests. This section only applies to protecting launch and other LTI messages that are being serialized and sent using POST and a content type of **x-www-form-encoded**.

The site <http://www.oauth.net> contains the specification for OAuth 1.0 and sample source code for implementing OAuth security. OAuth 1.0 specifies how to construct a base message string and then sign that string using the secret. The signature is then sent as part of the POST request and is validated by the TP using OAuth.

Per the OAuth specification, the signing process produces a number of values that are to be added to the launch request:

```
oauth_consumer_key=b289378-f88d-2929-lmsng.school.edu
oauth_signature_method=HMAC-SHA1
oauth_timestamp=1244834250
oauth_nonce=1244834250435893000
oauth_version=1.0
oauth_signature=Xddn2A%2BjzwjgBIVYkvigaKxCdcc%3D
```

The important values for signing a message using OAuth are the **oauth_consumer_key** and **oauth_consumer_secret**. The value of the **oauth_consumer_key** depends on which credentials are being used.

The **oauth_consumer_key** is passed in the message as plain text and identifies which TC is sending the message allowing the TP to look up the appropriate secret for validation. The **oauth_consumer_secret** is used to sign the message.

TC and TP must support and use the **HMAC-SHA1** signing method with OAuth fields coming from POST parameters for launch requests.

Since we are using OAuth in a signing-only scenario (i.e., we are not using OAuth to transfer third-party identity), there is no need for an **oauth_token** as per OAuth 1.0 documentation section 6.2.3. Since most of the OAuth implementations in the marketplace have upgraded to OAuth 1.0A, Tool Consumers should include **oauth_callback** and set it to a value such as "about:blank". Note that Basic LTI's **launch_presentation_return_url** serves a very different purpose than OAuth's **oauth_callback**.

Upon receipt of the POST, the TP will perform the OAuth validation utilizing the shared secret it has stored for the **oauth_consumer_key**. The timestamp should also be validated to be within a specific time interval. This time interval can be TP defined, but should be small (on the order of a few minutes if you do not record nonces or a few hours if you do).

The TP should keep a record of nonces received and only allow the use of any nonce a single time. Combined with the timestamp, this means that they only have to keep track of nonces for a period of time equal to their acceptable time interval. Recommended practice would be to have a time interval of 90 minutes so that you keep a record of nonces for 90 minutes.

NOTE that this security profile requires the TC and TP to have synchronized clocks. The use of a configurable time interval can adjust for slightly-off clocks, but setting the interval too large is discouraged.

9.3 OAuth Message Body Signing for application/xml Messages

These services follow a "Plain Old XML" (POX) pattern and the messages are signed using OAuth body signing to insure message integrity and establish the identity of the calling system (ie. the TP).

The body of the message is XML that follows the schema for the particular requested service operation requested and the message is signed using the **oauth_consumer_key** and **oauth_consumer_secret** that was used to do the launch of tool for the particular user/course/resource.

The procedure for signing a body using OAuth is described on this web site:

http://oauth.googlecode.com/svn/spec/ext/body_hash/1.0/oauth-bodyhash.html

It is important that all messages using these services must use a content type of **application/xml**. The services will legitimately reject any other content type. In particular, the OAuth body signing specification specifically prohibits the combination of **oauth_body_hash** and **x-www-form-encoded** data in any request.

Also these services will insist that all of the OAuth parameters are sent as part of the **Authorization** header. In particular, OAuth parameters from the request URL and POST body will not be processed.

The **oauth_body_hash** is computed using a SHA-1 hash of the body contents and added to the **Authorization** header. All of the OAuth parameters, HTTP method, and URL are signed like any other OAuth signed request. Other than in constructing the body hash value, the actual POST data is not involved in the computation of the **oauth_signature**.

Most OAuth libraries can produce and verify the signatures for these messages as most libraries already support sending OAuth parameters in the **Authorization** header.

A sample signed request is shown below. The line-breaks in the **Authorization** header are there to make it easier to read the values. The **oauth_signature** is not valid for the data below - it is just an example signature.

```
POST http://www.imsglobal.org/developers/BLTI/service_handle.php HTTP/1.0
Host: 127.0.0.1:80
Content-Length: 757
Authorization: OAuth realm="",oauth_version="1.0",
  oauth_nonce="29f90c047a44b2ece73d00a09364d49b",
  oauth_timestamp="1313350943",oauth_consumer_key="lmsng.school.edu",
  oauth_body_hash="v%2BxFnmDSHV%2Fj29qhXLwkFILrtPo%3D",
  oauth_signature_method="HMAC-SHA1",
  oauth_signature="8auRpRdPY2KRXUrOyz3HKCs92y8%3D"
Content-type: application/xml

<?xml version = "1.0" encoding = "UTF-8"?>
<imsx_POXEnvelopeRequest xmlns = "http://www.imsglobal.org/lis/oms1p0/pox">
  <imsx_POXHeader>
    <imsx_POXRequestHeaderInfo>
      <imsx_version>V1.0</imsx_version>
      <imsx_messageIdentifier>999999123</imsx_messageIdentifier>
    </imsx_POXRequestHeaderInfo>
  </imsx_POXHeader>
  <imsx_POXBody>
    <readResultRequest>
      <resultRecord>
        <sourcedGUID>
          <sourcedId>3124567</sourcedId>
        </sourcedGUID>
      </resultRecord>
    </readResultRequest>
  </imsx_POXBody>
</imsx_POXEnvelopeRequest>
```

Please consult the IMS General Web Services **[TODO: NEED-REFERENCES]** for the full detail on the **ims_POXHeader** fields. These definition and values for the header items are taken directly from IMS General Web Services. See "Table A1.2 Interpretation of the 'CodeMajor/severity' matrix" from IMS General Web Services WSDL Binding Guidelines **[NEED REF]** for further details on header values for 'unsupported' responses.

Each service will define its own XML Schema for the **imsx_POXBody** Request and Response content for a particular operation within a particular service.

10 Representing Basic LTI Links in a Cartridge

Note: This section is taken from the LTI Common Cartridge Interaction [LTI, 10 CCI] documentation.

A Basic LTI link is a simplified and self-contained LTI link. The Basic LTI link is defined in the resource section of an IMS Common Cartridge as follows:

```
<resource identifier="I_00010_R" type="imsbasiclti_xmlvlp0">
  <file href="I_00001_R/BasicLTI.xml"/>
</resource>
```

The **href** in the resource entry refers to a file path in the cartridge that contains an XML description of the Basic LTI link.

```
<?xml version="1.0" encoding="UTF-8"?>
<cartridge_basiclti_link xmlns="http://www.imsglobal.org/xsd/imslticc_vlp0"
  xmlns:blti = "http://www.imsglobal.org/xsd/imsbasiclti_vlp0"
  xmlns:lticm = "http://www.imsglobal.org/xsd/imslticm_vlp0"
  xmlns:lticp = "http://www.imsglobal.org/xsd/imslticp_vlp0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.imsglobal.org/xsd/imslticc_vlp0
http://www.imsglobal.org/xsd/lti/ltivlp0/imslticc_vlp0.xsd
http://www.imsglobal.org/xsd/imsbasiclti_vlp0
http://www.imsglobal.org/xsd/lti/ltivlp0/imsbasiclti_vlp0.xsd
http://www.imsglobal.org/xsd/imslticm_vlp0
http://www.imsglobal.org/xsd/lti/ltivlp0/imslticm_vlp0.xsd
http://www.imsglobal.org/xsd/imslticp_vlp0
http://www.imsglobal.org/xsd/lti/ltivlp0/imslticp_vlp0.xsd">
  <blti:title>Grade Book</blti:title>
  <blti:description>Grade Book with many column types</blti:description>
  <blti:custom>
    <lticm:property name="keyname">value</lticm:property>
  </blti:custom>
  <blti:extensions platform="my.lms.com">
    <lticm:property name="keyname">value</lticm:property>
  </blti:extensions>
  <blti:launch_url>url to the basiclti launch URL</blti:launch_url>
  <blti:secure_launch_url>secure url to the basiclti launch URL</blti:secure_launch_url>
  <blti:icon>url to an icon for this tool (optional)</blti:icon>
  <blti:secure_icon>secure url to an icon for this tool (optional)</blti:secure_icon>
  <blti:vendor>
    <lticp:code>vendor.com</lticp:code>
    <lticp:name>vendor.name</lticp:name>
    <lticp:description>This is a vendor of learning tools.</lticp:description>
    <lticp:url>http://www.vendor.com/</lticp:url>
    <lticp:contact>
      <lticp:email>support@vendor.com</lticp:email>
    </lticp:contact>
  </blti:vendor>
  <cartridge_bundle identifierref="BLTI001_Bundle"/>
  <cartridge_icon identifierref="BLTI001_Icon"/>
</cartridge_basiclti_link>
```

The **launch_url** contains the URL to which the LTI Launch is to be sent. The **secure_launch_url** is the URL to use if secure http is required. One of either the **launch_url** or the **secure_launch_url** must be specified. It is acceptable to specify both and if both are specified, the TC decides which to use. Typically, the TC will use a **secure_launch_url** when embedding the Tool in a secure page and the **launch_url** when embedding the tool in a non-secure page. So, it's important that the TP provides the same functionality whether the **launch_url** or **secure_launch_url** is used.

The **icon** and **secure_icon** are both optional and indicate a URL to be used for an icon to the tool.

Once the Basic LTI link is defined in the resources section of the cartridge manifest, it can be referenced in the organization section of the manifest as needed:

```
<item identifier="BasicLTI1" identifierref="I_00010_R">
  <title>Homework Problems</title>
```

```
</item>
```

The TC will generally display the **title** in the **item** entry in the user interface rather than **title** in the **basic_lti_link** entry.

The optional **custom** section can contain a set of key value pairs that were placed in the link in the system that originally authored the link. For example if the link were a section in an eTextbook, there might be a setting like:

```
<parameter key="section">1.2.7</parameter>
```

These parameters are sent back to the external tool when the tool is launched. If Basic LTI link is imported and then exported the **custom** should be maintained across the import/export process unless the intent is to re-author the link.

The **extensions** section allows the hosting TC to add its own key/value pairs to the link. The TC may use extensions to store information that the TC or authoring environment might use across an export-import cycle. In order to allow multiple sets of extensions to be contained in the same Basic LTI descriptor, authoring environments should add the **platform** attribute and include an identifier that identifies the authoring environment.

It is possible to include the icon for the link in the cartridge instead of including it as a URL using the **cartridge_icon** entry in the descriptor. The **identifierref** attribute points to a link that includes the icon image and a dependency is added to the resource section of the Basic LTI resource entry in the manifest as shown below.

```
<resource identifier="I_00010_R" type="imsbasiclti_xmlv1p0">
  <file href="I_00001_R/BasicLTI.xml"/>
  <dependency identifierref="BLTI001_Icon"/>
</resource>

<resource identifier="BLTI001_Icon"
type="associatedcontent/imscc_xmlv1p0/learning-application-resource">
  <file href="BLTI001_Media/learning_icon.gif"/>
</resource>
```

11 Using Services

A Tool Consumer can offer various services to the Tool Provider through the Tool Consumer Profile. This chapter discusses the services defined in LTI 1.2.

11.1 Tool Proxy Service

In LTI 1.2 it is possible to create and update a Tool Proxy via a REST endpoint. Methods to read or delete a Tool Proxy are not supported in LTI 1.2.

Each Tool Proxy has a URL which defines the physical address of that object within the Tool Consumer system. Every Tool Proxy URL has the following form:

[?baseURL/ims/lti/ToolProxy/?toolProxyGUID](#) (URI 11-1)

where *?baseURL* is the base URL for REST services within the Tool Consumer system, and *?toolProxyGUID* is the globally unique identifier for the Tool Proxy.

A Tool Provider can discover the base URL value for the Tool Consumer system by inspecting the appropriate `service_offered` element within the Tool Consumer Profile. The base URL value corresponds to the `endpoint_url` as shown in **Error! Reference source not found.**



```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "@subject" : "urn:ims:lti/ToolConsumerProfile/alpha.university.edu",
  "product_instance" : { },
  "service_offered" : [
    {
      "contract" : "http://purl.org/ims/lti/v1/service/ToolProxy",
      "method" : ["POST", "PUT"],
      "endpoint_url" : "http://lms.example.com/services/"
    }
  ],
  "capability" : [ ]
}
```

Figure 21 Discovering the Base URL for REST Services

Following traditional REST practices, one can create a new Tool Proxy by issuing an HTTP POST request to the endpoint

[?baseURL/lti/ToolProxy](#) (URI 11-2)

which represents the collection of all Tool Proxy objects. One can update a given Tool Proxy by issuing a PUT request to the URL for the particular object as specified by (URI 11-1).

When creating or updating a Tool Proxy, the JSON-LD representation of the object is submitted to the target system as the body of the HTTP request. The JSON-LD representation of a Tool Proxy is described in Chapter 3. As a reminder, the top-level structure of a Tool Proxy has the form shown in Figure 22.

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "tool_consumer_profile" : { "@iri", "urn:ims:lti/ToolConsumerProfile/alpha.university.edu" },
  "tool_profile" : { },
}
```

```
"security_contract" : { }
}
```

Figure 22 Top-Level Structure of Tool Proxy

When submitting a Tool Proxy to the Tool Consumer, the details of the `tool_consumer_profile` are omitted and instead, a reference to the TC Profile is provided, as shown in Figure 22.

11.1.1 POST Method of ToolProxy

To create a new ToolProxy within the TC, the TP issues an HTTP POST request to a REST endpoint conforming to (URI 11-2). The request body must contain the JSON-LD representation for a ToolProxy without the `guid` attribute. The GUID is assigned by the Tool Consumer and returned in the response. In particular, the HTTP response from the POST method is a degenerate `tool_proxy` element containing the `guid` attribute but no other content, as illustrated in Figure 23.

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/ToolProxy",
  "guid" : "c0913320-ded0-11e0-9572-0800200c9a66"
}
```

Figure 23 Response from ToolProxy POST method

The REST endpoint must be protected by SSL, and the TP must authenticate using OAuth Body Hash with `oauth_consumer_key` = "lti-tool-registration" and the `oauth_consumer_secret` corresponds to the value of the `reg_password` parameter from some ToolProxyDeploymentRequest (see Section 2.4).

Table 11-1 Status Codes for Tool Proxy POST Method

HTTP Status	Description	Response Body
201 Created	The request was successful.	JSON-LD (See Figure 23)
400 Bad Request	The supplied Tool Proxy is missing a required element or the Tool Proxy is ill-formed. For example, this status code applies if the security contract references service methods that were not offered in the Tool Consumer Profile.	Empty
401 Unauthorized	The client did not authenticate properly.	Empty
500 Internal Server Error	A generic error message, given when no more specific message is suitable.	Empty

11.1.2 PUT Method of ToolProxy

To update an existing ToolProxy within the TC, the TP issues an HTTP PUT request at the REST endpoint for the specific Tool Proxy (URI 11-1). The body of the request must contain the complete JSON-LD representation of the Tool Proxy. It is not possible to perform an incremental update of a Tool Proxy by just transmitted the portions that have changed. If the Tool Profile within the Tool Proxy is changed in any way, the `version` of the corresponding `product_info` must also change.

The REST endpoint must be secured by SSL, and the client must authenticate with OAuth Body Hash where the `oauth_consumer_key` is given by the Tool Proxy GUID and the `auth_consumer_secret` is given by the `shared_secret` from the original Security Contract. It is possible for the supplied Tool Proxy object to define a new `shared_secret` that will be used to secure future interactions between the TC and TP.

As a best practice, the TC should not immediately update the Tool Proxy instance. Instead, the TC administrator should have an opportunity to review and approve the changes before they become effective. However, in all cases, if the `shared_secret` is modified the new secret must take effect immediately.

Table 11-2 Status Codes for ToolProxy PUT Method

HTTP Status	Description	Response Body
200 OK	The request was successful and the Tool Proxy has been updated.	Empty
400 Bad Request	The supplied Tool Proxy is missing some required element or attribute.	Empty
401 Unauthorized	The client did not authenticate properly.	Empty
403 Forbidden	The request was a legal request and the client has authenticated properly, but the client does not have sufficient permissions to perform the requested action.	Empty
404 Not Found	The specified GUID does not correspond to any known Tool Proxy in the target system.	Empty
500 Internal Server Error	A generic error message, given when no more specific message is suitable.	Empty

11.2 LIS Result Service

This RESTful service supports reading and updating individual LIS Result objects [LIS, 10a]. The LTI 1.2 contract for this service presents a narrow “view” of the underlying Result object. This view consists of nothing but the score, which is constrained to be a decimal value in the range [0.0, 1.0]. The LTI 1.2 contract does not support creating or deleting Result objects.

Each Result Object has a URL which defines the physical address of that object within the Tool Consumer system.

The URL has the following form:

[?baseURI/lis/Result/?sourcedGUID](#) (URI 11-3)

where *?baseURI* is the base URL for REST services within the Tool Consumer system, and *?sourcedGUID* is the globally unique identifier for the Result object in question. A Tool Provider can discover the base URL value for the Tool Consumer system by inspecting the appropriate `service_offered` element within the Tool Consumer Profile. The base URL value corresponds to the `endpoint_url` as shown in

**Figure 24 Discovering the Base URL for REST Services**

Following traditional REST practices, one can read or update a Result object by issuing an HTTP GET or PUT request respectively at the object's URL.

For example, a Tool would read the current score in a certain Result object by issuing an HTTP GET request to an address like:

<http://lms/example.com/services/ims/Lis/Result/5b485a90-de60-11e0-9572-0800200c9a66>

The Tool may update the Result by issuing a PUT request to the same address.

6.1.1 The JSON-LD Representation of a Result object

The JSON-LD representation of an LIS Result object (in LTI 1.2) is illustrated in Figure 25

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/Outcomes",
  "@subject" : "urn:ims:lms/Result/5b485a90-de60-11e0-9572-0800200c9a66",
  "resultScore" : {
    "@datatype" : "decimal"
    "@literal" : "0.83"
  }
}
```

Figure 25 Example of Result object with a score

The `@context` property gives the URI for the JSON-LD context which maps terms in the JSON document to IRIs for unambiguous identification.

The `@subject` property gives the URN for the Result object. This URN must conform to the pattern defined by (URI 11-3), where `?baseURI` is replaced by “`urn:ims:`”. The service contract requires that the `resultScore` has a decimal value in the range [0.0, 1.0].

If the Result object does not yet have a score defined, then the `resultScore` property is omitted, as shown in Figure 26.

```
{
  "@context" : "http://purl.org/ims/lti/v1/jsonld/Outcomes",
  "@subject" : "urn:ims:lms/Result/ad9da880-018e-11e1-be50-0800200c9a66",
}
```

Figure 26 Example of Result object without a score

6.1.2 GET Method of Result

The GET method for a Result object returns a JSON-LD representation of the object, as described in Section 6.1.1.

The HTTP status codes for the GET method are listed in Table 11-3.

Table 11-3 Status Codes for GET Method of a Result Object

HTTP Status	Description	Response Body
200 OK	The request was successful.	JSON-LD
401 Unauthorized	The client did not authenticate properly.	Empty
403 Forbidden	The request was a legal request and the client has authenticated properly, but the client does not have sufficient permissions to perform the requested action.	Empty
404 Not Found	The specified sourcedGUID does not correspond to any known Result object in the target system.	Empty

500 Internal Server Error	A generic error message, given when no more specific message is suitable.	Empty
----------------------------------	---	-------

6.1.3 PUT Method of Result

The PUT method for a Result object is used to set, unset or modify the score within the Result. The HTTP request contains a JSON-LD representation of the object in the request body. To set or modify the score, the caller submits a Result object with a well-defined score, as illustrated in the example from Figure 25. To unset the score, the caller submits a Result object that has no score, as illustrated in Figure 26.

Unsetting the score is useful because it gives the Learner an opportunity to take the same assignment again. As discussed in Section 3.3.3, a Learner is allowed to launch an assignment only if the score is null.

The LTI 1.2 profile of the LIS Outcomes model stipulates that each Learner has at most one Result object for a given LineItem. If the Tool modifies a previously defined score in a Result, the TC may record a history of the previous scores. However, the TC must always present the same sourcedGUID for the Result when launching the assignment in the Tool. Subsequent versions of LTI will allow fine-grained manipulation of multiple attempts.

The HTTP status codes for the PUT method of a Result object are listed in Table 11-4.

Table 11-4 Status Codes for PUT Method of a Result Object

HTTP Status	Description	Response Body
200 OK	The request was successful.	Empty
401 Unauthorized	The client did not authenticate properly.	Empty
403 Forbidden	The request was a legal request and the client has authenticated properly, but the client does not have sufficient permissions to perform the requested action.	Empty
404 Not Found	The specified sourcedGUID does not correspond to any known Result object in the target system.	Empty
500 Internal Server Error	A generic error message, given when no more specific message is suitable.	Empty

Appendix A – LTI Standard Vocabularies

This Appendix is taken verbatim from the LTI Messages [LTI, 10 MSS] documentation.

The LTI v1.0 specification uses URN values to identify certain entities. This section contains URN vocabularies for ContextType and Role values.

A.1 ContextType Vocabularies

A.1.1 LIS vocabulary for ContextType

Handle	Full URN
CourseTemplate	urn:lti:context-type:ims/lis/CourseTemplate
CourseOffering	urn:lti:context-type:ims/lis/CourseOffering
CourseSection	urn:lti:context-type:ims/lis/CourseSection
Group	urn:lti:context-type:ims/lis/Group

A.2 Role Vocabularies

A.2.1 LIS vocabulary for System Role

The following table lists URN values for system role as defined by the LIS standard.

Handle	Full URN
SysAdmin	urn:lti:sysrole:ims/lis/SysAdmin
SysSupport	urn:lti:sysrole:ims/lis/SysSupport
Creator	urn:lti:sysrole:ims/lis/Creator
AccountAdmin	urn:lti:sysrole:ims/lis/AccountAdmin
User	urn:lti:sysrole:ims/lis/User
Administrator	urn:lti:sysrole:ims/lis/Administrator
None	urn:lti:sysrole:ims/lis/None

A.2.2 LIS vocabulary for Institution Role

The following table lists URN values for institution roles as defined by the LIS standard

Handle	Full URN
Student	urn:lti:instrole:ims/lis/Student
Faculty	urn:lti:instrole:ims/lis/Faculty
Member	urn:lti:instrole:ims/lis/Member
Learner	urn:lti:instrole:ims/lis/Learner
Instructor	urn:lti:instrole:ims/lis/Instructor
Mentor	urn:lti:instrole:ims/lis/Mentor
Staff	urn:lti:instrole:ims/lis/Staff
Alumni	urn:lti:instrole:ims/lis/Alumni
ProspectiveStudent	urn:lti:instrole:ims/lis/ProspectiveStudent
Guest	urn:lti:instrole:ims/lis/Guest
Other	urn:lti:instrole:ims/lis/Other
Administrator	urn:lti:instrole:ims/lis/Administrator

Observer	urn:lti:instrole:ims/lis/Observer
None	urn:lti:instrole:ims/lis/None

A.2.3 LIS vocabulary for Context Role

Roles within the LIS standard [LIS, 10a] consist of a RoleType and an optional SubRoleType. The handle for the corresponding URN value contains both elements, separated by a slash.

Handle	Full URN
Learner	urn:lti:role:ims/lis/Learner
Learner/Learner	urn:lti:role:ims/lis/Learner/Learner
Learner/NonCreditLearner	urn:lti:role:ims/lis/Learner/ NonCreditLearner
Learner/GuestLearner	urn:lti:role:ims/lis/Learner/ GuestLearner
Learner/ExternalLearner	urn:lti:role:ims/lis/Learner/ ExternalLearner
Learner/Instructor	urn:lti:role:ims/lis/Learner/Instructor
Instructor	urn:lti:role:ims/lis/Instructor
Instructor/PrimaryInstructor	urn:lti:role:ims/lis/Instructor/PrimaryInstructor
Instructor/Lecturer	urn:lti:role:ims/lis/Instructor/Lecturer
Instructor/GuestInstructor	urn:lti:role:ims/lis/Instructor/ GuestInstructor
Instructor/ExternalInstructor	urn:lti:role:ims/lis/Instructor/ ExternalInstructor
ContentDeveloper	urn:lti:role:ims/lis/ContentDeveloper
ContentDeveloper/ContentDeveloper	urn:lti:role:ims/lis/ContentDeveloper/ ContentDeveloper
ContentDeveloper/Librarian	urn:lti:role:ims/lis/ContentDeveloper/Librarian
ContentDeveloper/ContentExpert	urn:lti:role:ims/lis/ContentDeveloper/ContentExpert
ContentDeveloper/ExternalContentExpert	urn:lti:role:ims/lis/ContentDeveloper/ ExternalContentExpert
Member	urn:lti:role:ims/lis/Member
Member/Member	urn:lti:role:ims/lis/Member/Member
Manager	urn:lti:role:ims/lis/Manager
Manager/AreaManager	urn:lti:role:ims/lis/Manager/AreaManager
Manager/CourseCoordinator	urn:lti:role:ims/lis/Manager/CourseCoordinator
Manager/Observer	urn:lti:role:ims/lis/Manager/Observer
Manager/ExternalObserver	urn:lti:role:ims/lis/Manager/ExternalObserver
Mentor	urn:lti:role:ims/lis/Mentor
Mentor/Mentor	urn:lti:role:ims/lis/Mentor/Mentor
Mentor/Reviewer	urn:lti:role:ims/lis/Mentor/Reviewer
Mentor/Advisor	urn:lti:role:ims/lis/Mentor/Advisor
Mentor/Auditor	urn:lti:role:ims/lis/Mentor/Auditor
Mentor/Tutor	urn:lti:role:ims/lis/Mentor/Tutor
Mentor/LearningFacilitator	urn:lti:role:ims/lis/Mentor/LearningFacilitator

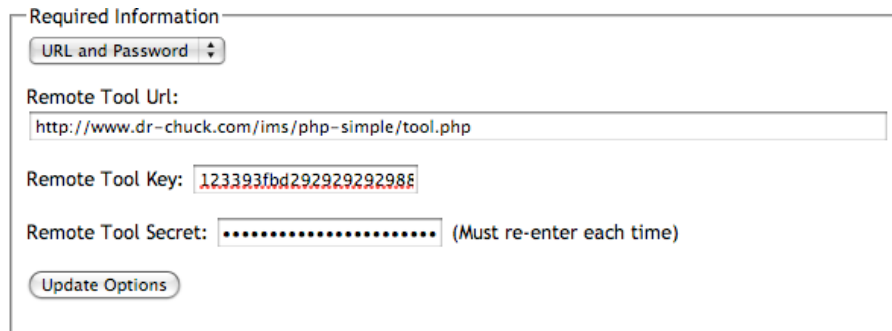
Mentor/ExternalMentor	urn:lti:role:ims/lis/Mentor/ExternalMentor
Mentor/ExternalReviewer	urn:lti:role:ims/lis/Mentor/ExternalReviewer
Mentor/ExternalAdvisor	urn:lti:role:ims/lis/Mentor/ExternalAdvisor
Mentor/ExternalAuditor	urn:lti:role:ims/lis/Mentor/ExternalAuditor
Mentor/ExternalTutor	urn:lti:role:ims/lis/Mentor/ExternalTutor
Mentor/ExternalLearningFacilitator	urn:lti:role:ims/lis/Mentor/ ExternalLearningFacilitator
Administrator	urn:lti:role:ims/lis/Administrator
Administrator/Administrator	urn:lti:role:ims/lis/Administrator/Administrator
Administrator/Support	urn:lti:role:ims/lis/Administrator/Support
Administrator/Developer	urn:lti:role:ims/lis/Administrator/Developer
Administrator/SystemAdministrator	urn:lti:role:ims/lis/Administrator/SystemAdministrator
Administrator/ExternalSystemAdministrator	urn:lti:role:ims/lis/Administrator/ ExternalSystemAdministrator
Administrator/ExternalDeveloper	urn:lti:role:ims/lis/Administrator/ExternalDeveloper
Administrator/ExternalSupport	urn:lti:role:ims/lis/Administrator/ExternalSupport
TeachingAssistant	urn:lti:role:ims/lis/TeachingAssistant
TeachingAssistant/TeachingAssistant	urn:lti:role:ims/lis/TeachingAssistant/ TeachingAssistant
TeachingAssistant/TeachingAssistantSection	urn:lti:role:ims/lis/TeachingAssistant/ TeachingAssistantSection
TeachingAssistant/ TeachingAssistantSectionAssociation	urn:lti:role:ims/lis/TeachingAssistant/ TeachingAssistantSectionAssociation
TeachingAssistant/ TeachingAssistantOffering	urn:lti:role:ims/lis/TeachingAssistant/ TeachingAssistantOffering
TeachingAssistant/ TeachingAssistantTemplate	urn:lti:role:ims/lis/TeachingAssistant/ TeachingAssistantTemplate
TeachingAssistant/TeachingAssistantGroup	urn:lti:role:ims/lis/TeachingAssistant/ TeachingAssistantGroup
TeachingAssistant/Grader	urn:lti:role:ims/lis/TeachingAssistant/Grader

Appendix B - Implementation Practice

This section includes *non-normative* discussion and recommendations to help guide implementations.

B.1 Authoring Links with Link-Level Credentials

If the TC chooses to support link-level credentials, they are supporting the ability for the Instructor to author Basic LTI links inside of the TC. The minimal authoring screen is very simple.

The image shows a web-based form titled "Required Information". At the top, there is a dropdown menu with "URL and Password" selected. Below this, the form has three main sections: "Remote Tool Url:" with a text input field containing "http://www.dr-chuck.com/ims/php-simple/tool.php"; "Remote Tool Key:" with a text input field containing "123393fbd29292929298f"; and "Remote Tool Secret:" with a masked text input field (dots) and a note "(Must re-enter each time)". At the bottom of the form is a button labeled "Update Options".

Required Information

URL and Password

Remote Tool Url:

http://www.dr-chuck.com/ims/php-simple/tool.php

Remote Tool Key: 123393fbd29292929298f

Remote Tool Secret: (Must re-enter each time)

Update Options

Figure B.2 Authoring screen for Basic LTI links inside of the TC.

Another possible authoring interface might be to allow the pasting of the XML **basic_lti_link** descriptor into an input field.

```
<?xml version="1.0" encoding="UTF-8"?>
<basic_lti_link xmlns="http://www.imsglobal.org/xsd/imsbasiclti_v1p0"
  xmlns:lticm="http://www.imsglobal.org/xsd/imslticm_v1p0"
  xmlns:lticp="http://www.imsglobal.org/xsd/imslticp_v1p0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imsglobal.org/xsd/imsbasiclti_v1p0
    http://www.imsglobal.org/xsd/lti/ltiv1p0/imsbasiclti_v1p0.xsd
    http://www.imsglobal.org/xsd/imslticm_v1p0
    http://www.imsglobal.org/xsd/lti/ltiv1p0/imslticm_v1p0.xsd
    http://www.imsglobal.org/xsd/imslticp_v1p0
    http://www.imsglobal.org/xsd/lti/ltiv1p0/imslticp_v1p0.xsd">
  <title>Grade Book</title>
  <description>Grade Book with many column types</description>
  <custom>
    <lticm:property name="keyname">value</lticm:property>
  </custom>
  <extensions platform="my.lms.com">
    <lticm:property name="keyname">value</lticm:property>
  </extensions>
  <launch_url>url to the basiclti launch URL</launch_url>
  <secure_launch_url>secure url to the basiclti launch URL</secure_launch_url>
  <icon>url to an icon for this tool (optional)</icon>
  <secure_icon>secure url to an icon for this tool (optional)</secure_icon>
  <vendor>
    <lticp:code>vendor.com</lticp:code>
    <lticp:name>vendor.name</lticp:name>
    <lticp:description>This is a vendor of learning tools.</lticp:description>
    <lticp:url>http://www.vendor.com/</lticp:url>
    <lticp:contact>
      <lticp:email>support@vendor.com</lticp:email>
    </lticp:contact>
  </vendor>
</basic_lti_link>
```

Figure B.3 Sample pasting of a Basic LTI link in XML.

As a best practice, TC systems should support both the URL/Key/Secret and XML/Key/Secret of authoring a Basic LTI link. The user interface for these options and how and where these options are shown to the user is up to the TC.

The TC might add other features like frame height, "open in new window" or add a title field to the link entry.

Display Information

Set Button Text:

Set Tool Title: (Above the tool)

Optional Launch Information

iFrame Height:

Debug Launch: ☐

When Debug Launch is selected, the tool pauses before launching and displays launch data.

Figure B.4 Sample interface for including various options inside the TC.

These screens will be available in the TC where the Instructor is creating the course organization and adding a new link. A typical approach is to make creating a Basic LTI launch just one more type of TC link in the course structure.

B.2 Security Policy / SandBoxing Launch Requests

TC systems will likely implement a number of security policy related features that can be controlled by both the TC administrator and the Instructor. These are some considerations:

- TC systems will likely limit the transmission of identifying information for users such as name and E-Mail to a trusted set of TPs.
- TC Administrators may want to allow only certain approved/trusted Instructors to be allowed to author their own Basic LTI links.
- The TC system may want the ability of an Instructor to further reduce/sandbox the data items transmitted to a TP.

It is out of the scope of this document to specify how the TC system controls which instructors can author LTI links or which URLs can be launched using LTI or which data is shared with particular TPs.

B.3 Roles

Some of the commonly used roles from LIS include **Learner**, **Instructor**, **Administrator**, **TeachingAssistant**, **ContentDeveloper**, and **Mentor**. Multiple roles can be included, separated by commas. TC systems should include as many roles as appropriate for the user (i.e., more roles are better). TC systems should be aware that simple TPs will key off the presence or absence of the **Instructor** role and group users into those with the **Instructor** role (read-write-configure) and those without the **Instructor** role (read).

B.4 Non-Context LTI Launches

While the typical use of a LTI link is in a context, it is also possible to use LTI to launch a link that is not part of a context. One example of a non-context launch might be a menu item that is part of the portal or part of a global menu in the TC.

Supporting non-context launches is optional for both the TP and TC.

If a LTI launch is coming from a non-context placement, the context information is simply omitted and the launch will contain the user and organization information but no context information.

B.5 LTI Sample Launch

The LTI launch protocol is a POST to the launch URL with the LTI parameters described above, properly signed using OAuth.

The most common launch approach will be for the TC to emit a form to the browser and then include code to automatically submit the form to the launch URL. The TP will assume that it is in a browser, process the input parameters, setting session information if necessary and optionally redirecting.

Here is a sample of an HTML form using a password of "secret" and **oauth_consumer_key** of "12345".

```
<div id="ltiLaunchFormSubmitArea">
<form action="http://dr-chuck.com/ims/php-simple/tool.php"
  name="ltiLaunchForm" id="ltiLaunchForm" method="post"
  enctype="application/x-www-form-urlencoded">
<input type="hidden" name="oauth_version" value="1.0"/>
<input type="hidden" name="oauth_nonce" value="c8350c0e47782d16d2fa48b2090c1d8f"/>
<input type="hidden" name="oauth_timestamp" value="1251600739"/>
<input type="hidden" name="oauth_consumer_key" value="12345"/>
<input type="hidden" name="resource_link_id" value="120988f929-274612"/>
<input type="hidden" name="user_id" value="292832126"/>
<input type="hidden" name="roles" value="Instructor"/>
<input type="hidden" name="lis_person_name_full" value="Jane Q. Public"/>
<input type="hidden" name="lis_person_contact_email_primary" value="user@school.edu"/>
<input type="hidden" name="lis_person_sourced_id" value="school.edu:user"/>
```

```

<input type="hidden" name="context_id" value="456434513"/>
<input type="hidden" name="context_title" value="Design of Personal Environments"/>
<input type="hidden" name="context_label" value="SI182"/>
<input type="hidden" name="lti_version" value="LTI-1p0"/>
<input type="hidden" name="lti_message_type" value="basic-lti-launch-request"/>
<input type="hidden" name="tool_consumer_instance_guid" value="lmsng.school.edu"/>
<input type="hidden" name="tool_consumer_instance_description"
    value="University of School (LMSng)"/>
<input type="submit" name="basiclti_submit" value="Launch Endpoint with BasicLTI Data"/>
<input type="hidden" name="oauth_signature_method" value="HMAC-SHA1"/>
<input type="hidden" name="oauth_callback" value="about:blank"/>
<input type="hidden" name="oauth_signature" value="TPFPK4u3NwmtLt0nDMP1G1zG30U="/>
</form>
</div>
<script language="javascript">
    document.getElementById("ltiLaunchFormSubmitArea").style.display = "none";
    nei = document.createElement('input');
    nei.setAttribute('type', 'hidden');
    nei.setAttribute('name', 'basiclti_submit');
    nei.setAttribute('value', 'Press to continue to external tool.');
```

This form is designed to work even if JavaScript is turned off in the browser – the user simply presses the submit button. If JavaScript is on, the button is quickly hidden and the form is automatically submitted.

The JavaScript must add a hidden field to the form when it is auto-submitting the form because the submit value was included in the OAuth base string and signature computation.

The following is the base string prior to the OAuth signature computation:

```

POST&http%3A%2F%2Fdr-chuck.com%2Ffims%2Fphp-
simple%2Ftool.php&basiclti_submit%3DLaunch%2520Endpoint%2520with%2520BasicLTI%2520Data%26context_
id%3D456434513%26context_label%3DSI182%26context_title%3DDesign%2520of%2520Personal%2520Environme
nts%26lis_person_contact_email_primary%3Duser%2540school.edu%26lis_person_name_full%3DJane%2520Q.
%2520Public%26lis_person_sourced_id%3Dschoo1.edu%253Auser%26lti_message_type%3Dbasic-lti-launch-
request%26lti_version%3DLTI-
1p0%26oauth_consumer_key%3D12345%26oauth_nonce%3Dc8350c0e47782d16d2fa48b2090c1d8f%26oauth_signatu
re_method%3DHMAC-
SHA1%26oauth_timestamp%3D1251600739%26oauth_version%3D1.0%26resource_link_id%3D120988f929-
274612%26roles%3DInstructor%26tool_consumer_instance_description%3DUniversity%2520of%2520School%2
520%2528LMSng%2529%26tool_consumer_instance_guid%3DLmsng.school.edu%26user_id%3D292832126
```

In the above string, all line wrapping needs to be removed. Notice that all of the POST values, including the submit button, are included in the base string (i.e., the string signed by OAuth).

B.6 Conformance

Conformance for Basic LTI is granted through the IMS CC-LTI Alliance and consists of certification testing for TC and TP implementations. For additional information about conformance, visit the CC-LTI Alliance here: <http://www.imsglobal.org/cc/alliance.html>.

B.7 Administrator / Instructor User Interfaces / Custom Parameters

While the user interface is completely up to the TC, there are several user interface patterns that have evolved to be quite effective in practice. There are two primary use case patterns:

B.7.1 Instructor Creates New Tools

If the TC decides to allow the instructor to place tools without administrator action by getting a URL, key, and secret from a TP and plugging them into a course structure. In this case, it is a good practice to allow the instructor

to enter custom parameters without requiring administrator assistance. Some TPs will need custom parameters to function properly. Also if the instructor is using a TC to produce an IMS Common Cartridge with Basic LTI links in the cartridge, often setting custom parameters for a tool placement is an essential part of authoring a cartridge.

B.7.2 Admin Creates New Tools, Instructor Only Places Tools

Another common case is to only allow the administrator to create new tools (i.e. key/secret/url) and then let the instructor place those pre-configured tools in their courses. In this use case, instructors never handle url/key/secret values. Even in this use case it is important to allow the instructor to be able to set or augment custom parameters for each placement. These parameters may be necessary for the TP to function and/or may be necessary if the instructor is building a course in the TC to be exported into an IMS Common Cartridge. It is not necessary to always give the instructor the option to configure custom parameters but it should be possible for the administrator to make a choice to reveal a user interface to set custom parameters.

B.8 TP URL Remapping in the TC

In order to allow a cartridge to be generated and useful worldwide, it is suggested that TC systems provide a URL remapping capability. The use case for this is a large publisher wants to create a cartridge and distribute it around the world, but enable the cartridge to be used in a situation where there is a local copy of the content. Perhaps the reason for the local copy of the content is poor or nonexistent connectivity to the central copy of the content or perhaps the course is being taught behind a firewall that does not allow outbound connections to the global Internet.

For these and other use cases, it is valuable for a TC administrator to be able to dynamically and easily remap all of a set of LTI launch URLs across an entire TC instance.

The remapping should at a minimum allow an exact match of an initial substring of a launch URL. An example mapping entry might be as follows:

```
http://global.oerhost.org/content/ -> http://info.ulcc.ac.uk/oercontent/
```

It is important that this mapping be easily changed and resolved at launch time rather than import time to give administrators the option to redirect launches as needed and as conditions change. For example, a school might start out importing cartridges that point to some central location far away, and as long as only a few courses use that content, there is no reason to remap launch URLs. However after the content becomes more popular, the administrators may want to make arrangements to have a local copy of the materials and once the local copy is available switch all launches to use that local copy.

Appendix C - Custom Parameter Substitution

Support for substitutable custom parameters is optional and the TP should anticipate that these parameters may come from the TC in their unsubstituted form.

Note: This material is copied from the IMS GLC Learning Tools Interoperability Vocabularies document section 2.4.

C.1 LTI User Variables

Message Variable Name	Corresponding LTI value
\$User.id	LaunchMixin.user_id (This is the local identifier for the user within the TC.)
\$User.image	The URL that contains an image of the user suitable for use as a profile picture or avatar.

C.2 LIS Person Variables

Message Variable Name	XPath for value from LIS Database
\$Person.sourcedId	personRecord/sourcedid
\$Person.name.full	personRecord/person/formname/[formnameType/instanceValue/text="Full"]/formattedName/text
\$Person.name.family	personRecord/person/name/partName[instanceName/text="Family"]/instanceValue/text
\$Person.name.given	personRecord/person/name/partName[instanceName/text="Given"]/instanceValue/text
\$Person.name.middle	personRecord/person/name/partName[instanceName/text="Middle"]/instanceValue/text
\$Person.name.prefix	personRecord/person/name/partName[instanceName/text="Prefix"]/instanceValue/text
\$Person.name.suffix	personRecord/person/name/partName[instanceName/text="Suffix"]/instanceValue/text
\$Person.address.street1	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart/nameValuePair/[instanceName/text="NonFieldedStreetAddress1"]/instanceValue/text ¹
\$Person.address.street2	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart/nameValuePair[instanceName/text="NonFieldedStreetAddress2"]/instanceValue/text
\$Person.address.street3	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart/nameValuePair/[instanceName/text="NonFieldedStreetAddress3"]/instanceValue/text
\$Person.address.street4	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart/nameValuePair/[instanceName/text="NonFieldedStreetAddress3"]/instanceValue/text
\$Person.address.locality	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart

¹ The "Preferred" instanceName is not part of the default LIS vocabulary. We are proposing to add this term in the LTI Profile of LIS so that we can support a single address instead of dealing with multiple address types as prescribed by the full LIS standard.

	/nameValuePair /instanceName/text="Locality"/instanceValue/text
\$Person.address.statepr	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart /nameValuePair /instanceName/text="Statepr"/instanceValue/text
\$Person.address.country	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart /nameValuePair /instanceName/text="Country"/instanceValue/text
\$Person.address.postcode	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart /nameValuePair /instanceName/text="Postcode"/instanceValue/text
\$Person.address.timezone	personRecord/person/address/[addressType/instanceValue/text="Preferred"]addressPart /nameValuePair /instanceName/text="Timezone"/instanceValue/text
\$Person.phone.mobile	personRecord/person/contactinfo[contactinfoType/instanceValue/text="Mobile"] /contactInfoValue/text
\$Person.phone.primary	personRecord/person/contactinfo [contactinfoType/instanceValue/text="Telephone_Primary"] /contactinfoValue /text
\$Person.phone.home	personRecord/person/contactinfo [contactinfoType/instanceValue/text="Telephone_Home"] /contactinfoValue /text
\$Person.phone.work	personRecord/person/contactinfo [contactinfoType/instanceValue/text="Telephone_Work"] /contactinfoValue /text
\$Person.email.primary	personRecord/person/contactinfo [contactinfoType/instanceValue/text="Email_Primary"] /contactinfoValue /text
\$Person.email.personal	person/contactinfo[contactinfoType/instanceValue/text="Email_Personal"] /contactinfoValue /text
\$Person.webaddress	personRecord/person/contactinfo[contactinfoType/instanceValue/text="Web-Address"] /contactinfoValue/text
\$Person.sms	personRecord/person/contactinfo[contactinfoType/instanceValue/text="SMS"] /contactinfoValue/text

C.3 LIS Course Template Variables

Message Variable Name	XPath for value from LIS Database
\$CourseTemplate.sourcedId	courseTemplateRecord/sourcedId
\$CourseTemplate.label	courseTemplateRecord/courseTemplate/label/textString
\$CourseTemplate.title	courseTemplateRecord/courseTemplate/title/textString
\$CourseTemplate.shortDescription	courseTemplateRecord/courseTemplate/catalogDescription/shortDescription
\$CourseTemplate.longDescription	courseTemplateRecord/courseTemplate/catalogDescription/longDescription
\$CourseTemplate.courseNumber	courseTemplateRecord/courseTemplate/courseNumber/textString
\$CourseTemplate.credits	courseTemplateRecord/courseTemplate/defaultCredits/textString

C.4 LIS Course Offering Variables

Message Variable Name	XPath for value from LIS Database
\$CourseOffering.sourcedId	courseOfferingRecord/sourcedId
\$CourseOffering.label	courseOfferingRecord/courseOffering/label

\$CourseOffering.title	courseOfferingRecord/courseOffering/title
\$CourseOffering.shortDescription	courseOfferingRecord/courseOffering/catalogDescription/shortDescription
\$CourseOffering.longDescription	courseOfferingRecord/courseOffering/catalogDescription/longDescription
\$CourseOffering.courseNumber	courseOfferingRecord/courseOffering/courseNumber/textString
\$CourseOffering.credits	courseOfferingRecord/courseOffering/defaultCredits/textString
\$CourseOffering.academicSession	courseOfferingRecord/courseOffering/defaultCredits/textString

C.5 LIS Course Section Variables

Message Variable Name	XPath for value from LIS Database
\$CourseSection.sourcedId	courseSection/sourcedId
\$CourseSection.label	courseSectionRecord/courseSection/label
\$CourseSection.title	courseSectionRecord/courseSection/title
\$CourseSection.shortDescription	courseSectionRecord/courseSection/catalogDescription/shortDescription
\$CourseSection.longDescription	courseSectionRecord/courseSection/catalogDescription/longDescription
\$CourseSection.courseNumber	courseSectionRecord/courseSection/courseNumber/textString
\$CourseSection.credits	courseSectionRecord/courseSection/defaultCredits/textString
\$CourseSection.maxNumberOfStudents	courseSectionRecord/courseSection/maxNumberOfStudents
\$CourseSection.numberOfStudents	courseSectionRecord/courseSection/numberOfStudents
\$CourseSection.dept	courseSectionRecord/courseSection/org[type/textString="Dept"] /orgName/textString
\$CourseSection.timeFrame.begin	courseSectionRecord/courseSection/timeFrame/begin
\$CourseSection.timeFrame.end	courseSectionRecord/courseSection/timeFrame/end
\$CourseSection.enrollControl.accept	courseSectionRecord/courseSection/enrollControl/enrollAccept
\$CourseSection.enrollControl.allowed	courseSectionRecord/courseSection/enrollControl/enrollAllowed
\$CourseSection.dataSource	courseSectionRecord/courseSection/dataSource
\$CourseSection.sourceSectionId	createCourseSectionFromCourseSectionRequest/sourcedId

C.6 LIS Group Variables

Message Variable Name	XPath for value from LIS Database
\$Group.sourcedId	groupRecord/sourcedId
\$Group.grouptype.scheme	groupRecord/group/groupType/scheme/textString
\$Group.grouptype.typevalue	groupRecord/group/groupType/typevalue/textString
\$Group.grouptype.level	groupRecord/group/groupType/typevalue/level/textString
\$Group.email	groupRecord/group/email
\$Group.url	groupRecord/group/url

\$Group.timeFrame.begin	groupRecord/group/timeframe/begin
\$Group.timeFrame.end	groupRecord/group/timeframe/end
\$Group.enrollControl.accept	groupRecord/group/enrollControl/enrollAccept
\$Group.enrollControl.allowed	groupRecord/group/enrollControl/enrollAllowed
\$Group.shortDescription	groupRecord/group/description/shortDescription
\$Group.longDescription	groupRecord/group/description/longDescription
\$Group.parentId	groupRecord/group/relationship[relation="Parent"]/sourcedId

C.7 LIS Membership Variables

Message Variable Name	XPath for value from LIS Database
\$Membership.sourcedId	membershipRecord/sourcedId
\$Membership.collectionSourcedId	membershipRecord/membership/collectionSourcedId
\$Membership.personSourcedId	membershipRecord/membership/member/personSourcedId
\$Membership.status	membershipRecord/membership/member/role/status
\$Membership.role	membershipRecord/membership/member/role/roleType
\$Membership.createdTimestamp	membershipRecord/membership/member/role/dataTime
\$Membership.dataSource	membershipRecord/membership/member/role/dataSource

C.8 LTI LineItem Variables

\$LineItem.sourcedId	lineItemRecord/sourcedId
\$LineItem.type	lineItemRecord/lineItem/lineItemType
\$LineItem.type.displayName	lineItemTypeRecord/lineItemType/displayName
\$LineItem.resultValue.max	resultValueRecord/resultValue/valueRange/max <i>where</i> resultValueRecord.sourcedId = lineItemRecord/lineItem/resultValueSourcedId
\$LineItem.resultValue.list	resultValueRecord/resultValue/valueList/orderValue <i>where</i> resultValueRecord.sourcedId = lineItemRecord/lineItem/resultValueSourcedId
\$LineItem.dataSource	lineItemRecord/lineItem/dataSource

C.9 LIS Result Variables

\$Result.sourcedId	resultRecord/sourcedId
\$Result.createdTimestamp	resultRecord/result/date
\$Result.status	resultRecord/result/statusofResult
\$Result.resultScore	resultRecord/result/resultScore/textString

\$Result.dataSource	resultRecord/result/dataSource
---------------------	--------------------------------

Appendix D – Deprecated Parameter Names

LTI 1.0 defined a long list of standard parameters for an LTI launch request. All of these parameter names are still supported in LTI 1.2, but many of them are now deprecated. The preferred method for handling these parameters in LTI 1.2 is to use custom parameters with variable substitution as described in Section 2.2.

If an LTI Link is associated with a Tool Proxy, then it is clear that the link was constructed in accordance with the rules for LTI 1.2 (with “custom_” as a prefix). In this case, the TC should not use any of the deprecated parameter names. On the other hand, if an LTI link is NOT associated with a Tool Proxy, then during a transitional period, the TC should list the parameter in the launch request twice – once using the deprecated name (without the “custom_” prefix) and again with the prefix.

The following list enumerates the parameters from LTI 1.0 that are deprecated in LTI 1.2.

context_title=Design of Personal Environments *(Deprecated. Use custom_context_title)*
A title of the context – it should be about the length of a line.

context_label=SI182 *(Deprecated. Use custom_context_label)*
A label for the context – intended to fit in a column. This parameter is recommended.

resource_link_title=My Weekly Wiki *(Deprecated. Use custom_resource_link_title)*
A title for the resource. This is the clickable text that appears in the link.

resource_link_description=... *(Deprecated. Use custom_resource_link_description)*
A plain text description of the link’s destination, suitable for display alongside the link. Typically no more than several lines long.

lis_person_name_given=Jane *(Deprecated. Use custom_lis_person_name_given)*
lis_person_name_family=Public *(Deprecated. Use custom_lis_person_name_family)*
lis_person_name_full=Jane Q. Public *(Deprecated. Use custom_lis_person_name_full)*
lis_person_contact_email_primary=user@school.edu *(Deprecated. Use custom_lis_person_contact_email_primary)*

These fields contain information about the user account that is performing this launch. The names of these data items are taken from LIS. The precise meaning of the content in these fields is defined by LIS.

user_image=http://... *(Deprecated. Use custom_user_image)*
This attribute specifies the URI for an image of the user who launched this request. This image is suitable for use as a "profile picture" or an avatar representing the user.

lis_person_sourcedid=school.edu:user *(Deprecated. Use custom_list_person_sourcedid)*
This field contains the LIS identifier for the user account that is performing this launch. The example syntax of "school:user" is not the required format – **lis_person_sourcedid** is simply a globally unique identifier (i.e., a normalized string). This field is optional and its content and meaning are defined by LIS.

lis_course_offering_sourcedid=school.edu:SI182-F08 *(Deprecated. Use custom_lis_course_offering_sourcedid)*

lis_course_section_sourcedid=school.edu:SI182-001-F08 *(Deprecated. Use custom_lis_course_section_sourcedid)*

These fields contain LIS course identifiers associated with the context of this launch. These fields are optional and their content and meaning are defined by LIS.

tool_consumer_instance_name=SchoolU *(Deprecated. Use custom_tool_consumer_instance_name)*
This is a user visible field – it should be about the length of a column.

tool_consumer_instance_description=University of School *(Deprecated. Use custom_tool_consumer_instance_description)*
This is a user visible field – it should be about the length of a line.

tool_consumer_instance_url=http://lmsng.school.edu *(Deprecated. Use custom_tool_consumer_instance_url)*
This is the URL of the consumer instance.

tool_consumer_instance_contact_email=System.Admin@school.edu *(Deprecated. Use custom_tool_consumer_instance_contact_email)*
An email contact for the TC's service provider.

Appendix E – Developer Cookbooks

E.1 Outcomes Reporting Cookbook

E.1.1 Tool Consumer Requirements for Outcomes Reporting

Requirement	Reference
1. Publish a Tool Consumer Profile at some URL. See Figure 27 for an example of a Tool Consumer Profile that supports outcomes reporting.	Section 3.2
2. User Interface to Request Access to a Tool. Implement a user interface which allows the TC Administrator to request access to an LTI Tool. This interface accepts a URL from the Administrator and then posts a ToolProxyDeploymentRequest to the specified URL.	Section 5.1
3. Tool Proxy Service. Implement the POST method of the Tool Proxy Service. This method allows the Tool Provider to register a new Tool Proxy with the Tool Consumer.	Section 11.1.1
4. User Interface to Make Tool Proxy Available. Implement a user interface which allows the TC Administrator to make a newly registered Tool Proxy available.	Section 5.4
5. Link Authoring. Implement a user interface which allows Content Builders to create LTI Links that are associated with a Resource Type.	Section 4.1
6. Tool Launch Service. Implement a service that can launch a Tool from an LTI Link.	Section 2.6
7. Result Auto-create Capability. The Tool Launch Service must include support for the Result.autocreate capability.	Section 3.3.3
8. LIS Result Service. Implement PUT method of the LIS Result Service	Section 6.1.3

```
<?xml version="1.0" encoding="UTF-8"?>
<tool_consumer_profile lti_version="LTI-1p1"
  xmlns="http://www.imsglobal.org/Lti/v1/schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <product_instance guid="LMS3.someuniversity.edu">
    <product_info>
      <product_name>Alpha LMS</product_name>
      <version>1.0</version>
      <product_family code="alpha">
        <vendor code="LMSVendor.com">
          <timestamp>2011-10-01T12:00:00</timestamp>
          <name>LMS Vendor</name>
        </vendor>
      </product_family>
    </product_info>
  </product_instance>
  <service_offered>
    <contract rdf:resource="http://purl.org/ims/Lti/v1/service#ToolProxy"/>
    <method name="POST"/>
    <endpoint_url>http://lms3.someuniversity.com/services/</endpoint_url>
  </service_offered>
  <service_offered>
    <contract rdf:resource="http://purl.org/ims/Lis/v1/service#ResultService"/>
    <method name="PUT"/>
    <endpoint_url>http://lms3.someuniversity.com/services/</endpoint_url>
  </service_offered>
  <capability rdf:resource="http://purl.org/ims/Lti/v1/message#basic-Lti-launch-request"/>
  <capability rdf:resource="http://purl.org/ims/Lti/v1/variable#Result.sourcedGUID"/>
  <capability rdf:resource="http://purl.org/ims/Lti/v1/capability#Result.autocreate"/>
</tool_consumer_profile>
```

Figure 27 Profile for a Tool Consumer that support Outcomes Reporting

E.1.2 Tool Provider Requirements for Outcomes Reporting

Requirement	Reference
1. Message Handler for Tool Proxy Deployment Request. Implement a message handler that can receive a Tool Proxy Deployment Request. The handler should perform the following actions.	Section 2.4
a. Tool Profile. Construct a Tool Profile that describes the Tool that will be reporting assessment results to the Tool Consumer.	Section 3.4
i. Enable Result autocreate capability. The Tool Profile must contain at least one Resource Handler that enables the <code>Result.autocreate</code> capability.	Section 3.4.4
ii. Custom Parameter for Result GUID. The Resource Handler(s) must declare a custom parameter whose value is given by the <code>\$Result.sourcedGUID</code> variable.	Section 3.4.3
b. Security Contract. Define the Security Contract for the integration with the Tool Consumer. This contract enumerates the service methods that the Tool intends to use and it also encapsulates a shared secret that will be used for future interactions. The Security Contract must include the PUT method of Result Service.	Section 3.5
c. Tool Proxy. Construct a Tool Proxy that aggregates the Tool Profile and Security Contract. See Figure 28 for an example of a Tool Proxy that supports outcomes reporting.	Chapter 3
d. Register the Tool Proxy. Register the Tool Proxy with the Tool Consumer by issuing a POST request to Tool Consumer's Tool Proxy Service.	Section 11.1.1
e. Return control to the Tool Consumer. Redirect the user's browser back to the Tool Consumer at the <code>launch_presentation_return_url</code> specified in the <code>ToolProxyDeploymentRequest</code> .	Section 5.4
2. Message Handler for Tool Launch Request. Implement a message handler that receives a basic-lti-launch-request. This handler must be prepared to receive the GUID for a Result object so that the Tool can later report a score using this GUID. The parameter that holds the Result GUID is defined in the Tool Profile (see Requirement 1.a.ii above.)	Section 2.3
3. Report Score to the Tool Consumer. After the Learner has completed the assignment, the Tool should report a score by issuing a PUT request to the Tool Consumer's Result Service.	Section 6.1.3

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ToolProfile [
  <!ENTITY msg "http://purl.org/ims/lti/v1/message#">
  <!ENTITY baseURL "http://purl.org/ims/lti/v1/baseURL#">
  <!ENTITY cap "http://purl.org/ims/lti/v1/capability#">
]>

<tool_proxy lti_version="LTI-1p1"
  xmlns="http://www.imsglobal.org/Lti/v1/schema#"
  xmlns:lti-type="http://www.imsglobal.org/Lti/v1/type#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <tool_profile>
    <product_instance>
      <product_info>
        <product_name>Acme Chemistry Tutor</product_name>
        <version>2.3</version>
        <product_family code="chemtutor">
          <vendor code="acme.example.com">
            <timestamp>2011-10-01T12:00:00</timestamp>
            <name>Acme Corporation</name>
          </vendor>
        </product_family>
      </product_info>
    </product_instance>
    <base_url_choice>
      <selector rdf:resource="&baseURL;DefaultSelector"/>
      <default_base_url>http://tp.example.com/lti</default_base_url>
    </base_url_choice>
    <resource_handler>
      <resource_type code="assessment"/>
      <name>Acme Assessment</name>
      <message path="assessment/launch">
        <message_type rdf:resource="&msg;basic-lti-launch-request"/>
        <parameter>
          <name>result_id</name>
          <variable>$Result.sourcedGUID</variable>
        </parameter>
        <capability rdf:resource="&cap;Result.autocreate"/>
      </message>
    </resource_handler>
  </tool_profile>

  <security_contract>
    <shared_secret>39B4911F632D1</shared_secret>
    <tool_service>
      <contract rdf:resource="http://purl.org/ims/lis/v1/service#ResultService"/>
      <method name="PUT"/>
    </tool_service>
  </security_contract>

</tool_proxy>

```

Figure 28 Tool Proxy that supports Outcomes Reporting

About This Document

Title:	IMS Learning Tools Interoperability Basic LTI Implementation Guide
Co-chairs:	Greg McFall (Pearson), Lance Neumann (Blackboard)
Editor:	Charles Severance (IMS GLC)
Version:	v1.0
Version Date:	17 May 2010
Release:	v1.0
Status:	Final
Revision Information:	Original Release
Purpose:	This document is made available for adoption by the public community at large.
Document Location:	Join the discussion and post comments on the LTI Public Forum: http://www.imsglobal.org/community/forum/categories.cfm?catid=44

List of Contributors

The following individuals contributed to the authoring of this document:

Hal Herzog	Learning Objects	Charles Severance	IMS GLC
Greg McFall	Pearson	Colin Smythe	IMS GLC
Mark McKell	IMS GLC	Bruno Van Haetsdaele	Wimba
Lance Neumann	Blackboard		

Revision History

Version No.	Release Date	Comments
Base Document v1.0	27 July 2009	The first formal release of the Base Document. This document is released for review by the IMS GLC LTI Project Group.
Internal Draft Final v1.0	30 October 2009	The first formal release of the Internal Draft document. This document is released for interoperability implementation by the IMS GLC Members and Affiliates.
Public Draft v1.0	15 February 2010	The first formal release of the Public Draft. This document is released for interoperability adoption by the community at large.
Final v1.0	17 May 2010	The first formal release of the Final specification. This document is released for public adoption.
Draft 1.1	08 Sep 2011	Added the tool registration and grade return use cases.

IMS Global Learning Consortium, Inc. ("IMS GLC") is publishing the information contained in this IMS GLC Learning Tools Interoperability Basic LTI Implementation Guide ("Specification") for purposes of scientific, experimental, and scholarly collaboration only.

IMS GLC makes no warranty or representation regarding the accuracy or completeness of the Specification. This material is provided on an "As Is" and "As Available" basis.

The Specification 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 Specification 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 GLC Learning Tools Interoperability Basic LTI Implementation Guide v1.0
Revision: 17 May 2010*