Guide to the Tutor Message format (v4)

a standard XML vocabulary for logging student and tutor actions

Version 1.0 for tutor_message_v4.xsd, tutor_message_v4.dtd

2011-01-10

2008-12-08

2007-09-28

Table of Contents

1. About this document

1.1. Who should read this document?

This guide is intended for a software developer who wants to evaluate, implement, or update logging in an educational tutoring application, or convert existing logs created by a tutoring application. In addition, a developer working on a learning environment could describe the application's events in the format described in this document. By doing so, that application would satisfy the "recordability" metric for a *tool* (Ritter and Blessing 1998), describing events in a format that the Cognitive Tutor Authoring Tools [http:// ctat.pact.cs.cmu.edu] (CTAT) software could read. CTAT could then record those application events (for playback later) or provide tutoring for a student using that application.

This document may also be of interest to an educator or researcher who wants to better understand the tutor message logging format, the standard logging format espoused by the Pittsburgh Science of Learning Center [http://www.learnlab.org] (PSLC).

1.2. How to use this document

Start at the beginning if you are unfamiliar with logging from tutoring applications, or are not sure you should implement logging to the format described in this document.

If you are familiar with logging but not the Tutor Message format or the Online Learning Initiative logging format, start with Section 2, "The Tutor Message format".

If you are in the process of implementing logging, consider using this guide as a reference for the Tutor Message format. It details the requirements of the format, and describes scenarios for logging the various message types. In addition, examples of tutor interfaces and XML logging sequences may make the logging specification more concrete.

Lastly, this guide describes DataShop processing expectations, which are not expressed in the XML Schema or DTD. While validating is an important step in ensuring your XML conforms to this format, it is only part of correctness.

Conventions used in this document

`<element>` represents an XML element. `attribute` represents an XML attribute of an element.

`<extended_code_examples>` appear in this typeface. Blue text represents new code added to an example.

In the diagrams of XML content models in this document, a box represents an XML element. In a box, the first line of text denotes the name of the element, while indented text within a box represents an attribute. Numbers in parentheses describe the number of times that element can or should appear. For example, the number "1" means that one instance of this element must be included in your XML to be valid, while the number "0" means that the element is optional. Colors in the diagrams are used to show elements with identical content models appearing in different message types. The `<meta>` element, for example, appears in all message types.

1.3. What is a tutoring application?

A tutoring application tracks the student as he or she works, and provides hints and feedback in response to student actions.

The tutor message format is designed to capture the details of student sessions with tutoring applications. If an application provides no tutoring—if it is primarily for assessment, or provides a simulation only—the tutor message format may be a useful logging format: you can describe audio and video actions and user interface events using the tutor message format. Contact the DataShop team if you're uncertain whether or not your application should log to this specification. See Section 1.6, "Where to go for more information" for contact information.

1.4. What is the difference between a tool and a tutor?

This tutor message format describes student and tutor actions in terms of the "tool" and the "tutor". This distinction is useful as deciding which message type to record or send depends on knowing what the source of that action is (tool or tutor).

A *tool* is an interactive application with which a user interacts. A *tutor* is the component of an application that provides tutoring to the user. In general, a student's action with the application is captured in a tool message, while the tutor's response to the student's action(s) is captured in a tutor message.

Where a single application may be a tutoring application, it can be considered to have both of these components. This distinction is made in the 1996 paper "An architecture for plug-in tutor agents" by Steve Ritter and Kenneth Koedinger, in which the authors propose architectural principals for adding tutoring to an application (Ritter and Koedinger 1996).

1.5. What is media logging?

Media logging, described in this document in Section 3, "Media Logging", is an extension of the Tutor Message format to log student interactions with media. It was designed with video and audio in mind, but attempts to remain applicable to any media type. It captures when the media was initially presented to the student, and how the student interacts with that media. For example, it captures when the student stops, starts, cues, or mutes any media. It provides support for both tutored and untutored media.

1.6. Where to go for more information

Both the Tutor Message format and Java Logging Library are maintained by the Pittsburgh Science of Learning Center (PSLC) DataShop team. They can be reached by email at datashop-help@lists.andrew.cmu.edu [mailto:datashop-help@lists.andrew.cmu.edu].

The Flash Logging Library is developed and maintained by the Cognitive Tutor Authoring Tools [http:// ctat.pact.cs.cmu.edu] (CTAT) team. This logging library is bundled with CTAT, and is also available as a standalone download [http://ctat.pact.cs.cmu.edu/index.php?id=logging-flash]. The CTAT team can be contacted at ctat-support@lists.andrew.cmu.edu [mailto:ctat-support@lists.andrew.cmu.edu].

2. The Tutor Message format

The tutor message format describes an XML hierarchy in which various message elements are contained within a root element. The format is described normatively in an XML Schema Definition (XSD), and informally in a DTD.

2.1. Validating your XML

The normative schema for the Tutor Message format is defined using the XML Schema language in an XML Schema Definition (XSD). You should use this XSD for the purposes of validating your XML to determine how well it conforms to the Tutor Message format.

The XSD file can be downloaded from the DataShop web site:

http://learnlab.web.cmu.edu/dtd

You may also validate against the DTD; however, we recommend you validate with the XSD, which expresses the constraints of the Tutor Message format more accurately than the older DTD representation.

Validation tools

DataShop provides a free validation tool, the XML Validator Tool [http://learnlab.web.cmu.edu/about/xmlvalidator.html], for validating Tutor Message XML. We strongly encourage you to use this tool for validation.

2.2. Root element

The root element of an XML file or message that conforms to the tutor message format is the <tutor_related_message_sequence>. The XML Schema Definition defines the possible contents of this element.

Here's how your XML should start and end:

```
<?xml version="1.0" encoding="UTF-8"?>
<tutor_related_message_sequence
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='http://learnlab.web.cmu.edu/dtd/tutor_message_v4.xsd'
    version_number="4"
>
  <!-- Tutor message elements go here -->

</tutor_related_message_sequence>
```

As in the above code, start your XML with:

• an XML prolog (XML version, encoding);

- the opening tag for the `tutor_related_message_sequence` root node;

- a qualifier for the schema instance namespace (`xmlns:xsi`);

- a pointer to the schema itself (`xsi:noNamespaceSchemaLocation`), which references the schema file; and

- a `version_number` attribute with value 4—this is the version of the Tutor Message format described in this document.

Note

The DataShop will continue to accept logs that conform to version 2 of the Tutor Message format, as well as logs conforming to version 4.

Be sure the data in your XML complies with the encoding specified in the XML prolog. UTF-8 is not required; other ISO encodings are permitted.

End your XML by closing the root element, `</tutor_related_message_sequence>`.

If you are using the older DTD version of the schema, your XML should start and end like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tutor_related_message_sequence SYSTEM
  "http://learnlab.web.cmu.edu/dtd/tutor_message_v4.dtd">
<tutor_related_message_sequence version_number="4">

  <!-- Tutor message elements go here -->

</tutor_related_message_sequence>
```

A note about sending XML to an OLI logging database

When logging to an Online Learning Initiative (OLI) database, your XML must be transported as the text value of OLI's `log_action` element. In this way, the XML you create can be stored in an OLI logging database such as that used by PSLC DataShop. This property of nesting XML is commonly referred to as "XML within XML".

Send your XML in the OLI log_action container when:

- **Logging directly to an OLI log database.** The DataShop team, for example, hosts such a database on their server. The CTAT software, for example, provides this functionality.

- **Logging from an OLI course delivery system.** This behavior is somewhat automatic, however, so contact the OLI group for details.

- **Logging to files that will be loaded directly into an OLI log database.** The CTAT software, for example, also provides this functionality.

The diagram below shows the position of the `<tutor_related_message_sequence>` when sent in its OLI container:
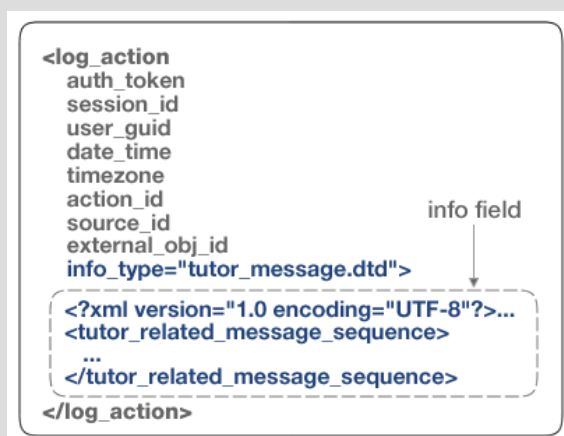
Figure 1. The OLI log_action element with embedded XML (tutor_related_message_sequence).

Note

- If logging to an OLI logging database (instead of to file), `session_id` must be a globally unique identifier (GUID).

- When embedding a tutor_related_message_sequence as XML within the OLI log_action element, the `info_type` attribute's value must be set to `tutor_message.dtd`. This tells the system interpreting the log that the contents conform to this schema. Use this value (`tutor_message.dtd`) even if you are writing XML that will conform to the XSD (XML Schema).

- If your tutor application is implemented in Java or Flash, use the Java or Flash logging library to wrap and send the tutor_related_message_sequence XML. These libraries are available from the DataShop [http://learnlab.web.cmu.edu/about/libraries.html].

2.3. Message types

The Tutor Message format describes four message types used to describe a tutoring session. These are:

- context_message: establishes the context for the following sequence of messages

- tool_message: describes a student's interaction with the "tool"

- tutor_message: describes the tutor's response to the student's action

- message: describes other information not adequately captured by the other message types; CTAT, for example, uses this message type for replay.

These messages must appear within a `<tutor_related_message_sequence>` element, as shown below.
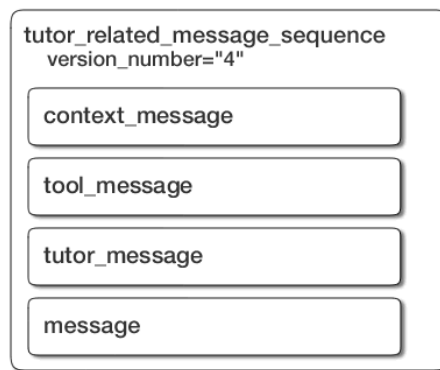
Figure 2. Message types, which appear within the tutor_related_message_sequence

Student action, tutor response: tool-tutor message pairs

A common unit of interaction is a student action with the tool and a tutor response for that action. This sequence is represented by the tool and tutor message pair.

Note

As of September 2007 (DataShop v2.3), the DataShop will import unpaired tool messages (a tool message without a tutor message), but not unpaired tutor messages (a tutor message without a tool message). Previously (DataShop v1.0–2.2), DataShop did not import unpaired tutor or tool messages.

2.4. <context_message>

Establishing context with the opening context message

The first element contained in the `<tutor_related_message_sequence>` should be a `<context_message>`. This single message establishes the context for the following sequence of elements (`<tool_message>`, `<tutor_message>`, or `<message>`).

For a researcher, a `<context_message>` can answer questions such as:

- Were these data collected in a classroom? If so, what were the details?

- Is there a hierarchy surrounding this data sequence?

- Were these data from a particular research condition?

- With which problem or activity do these log messages correspond?

Tip

You only need to log a single `<context_message>` element per student per single problem per session (a student can have multiple sessions). Logging once instead of with every tool and tutor message pair saves space. If the context changes—if the student logs out or in, or if the problem, session, or activity changes—then a new context message must be logged.
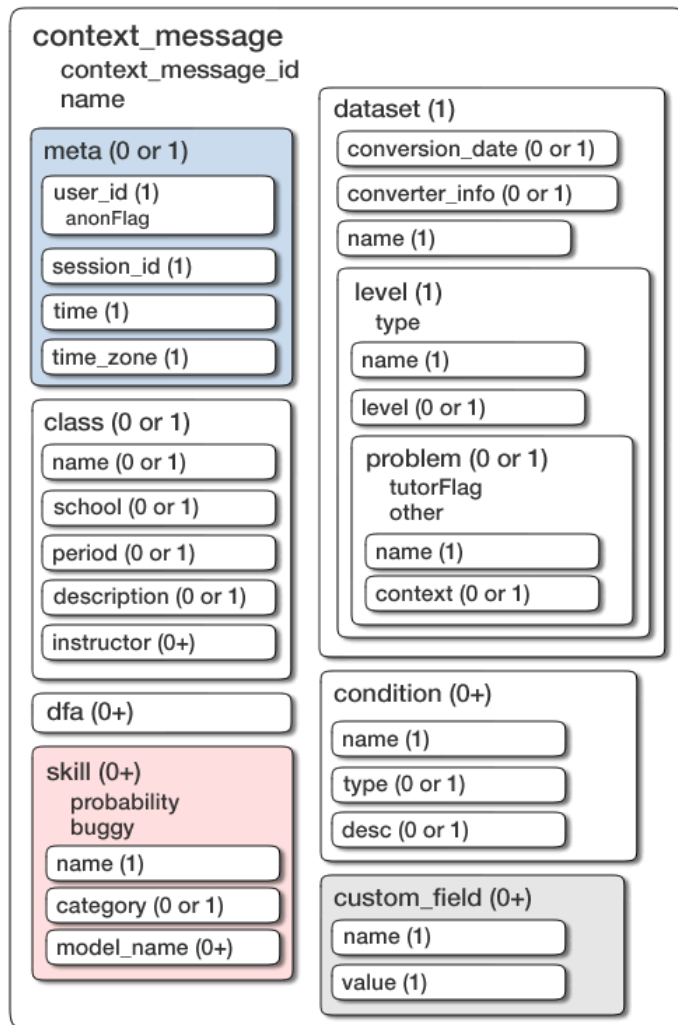

Figure 3. Structure of a <context_message> element

2.4.1. Attributes of <context_message>

A `<context_message>` has two required attributes:

- `name`

- `context_message_id`

The `name` attribute is used to indicate where the student is in the process of working on a tutor or problem. The PSLC DataShop team has established some canonical values for this attribute that should be used, displayed in Table 1, "Recommended values for the <context_message> name attribute".

Description	Preferred value	Other possible values
Starting a problem or activity	START_PROBLEM	PROBLEM_START START QUESTION LOAD_PROBLEM LOAD_TUTOR LOAD_AUDIO LOAD_VIDEO LOAD_MEDIA START_TUTOR START_AUDIO START_VIDEO START_MEDIA
Skipping a problem or activity	SKIP_PROBLEM	SKIP_TUTOR SKIP
Resuming a problem or activity	RESUME_PROBLEM	RESUME CONTINUE RESUME_TUTOR CONTINUE_PROBLEM
Finishing a problem or activity	DONE_PROBLEM	DONE FINISHED
Quitting a problem or activity	QUIT_PROBLEM	QUIT
Resetting a problem or activity	RESET_PROBLEM	RESET
Reading (REAP [http://reap.cs.cmu.edu/])	READING	

Table 1. Recommended values for the <context_message> name attribute

The `context_message_id` attribute ties a sequence of tutor-related messages together; when referenced in the other messages, it identifies this message as providing the relevant context.

Important

The value of the `context_message_id` attribute must be a globally unique identifier (GUID). In DataShop, messages will be grouped by user_id, session_id, and context_message_id.

2.4.2. Child elements of <context_message>

- meta

- dataset (required)

- class

- condition

- skill (unused by DataShop in the context message)

- custom_field

- [dfa](#) (unused by DataShop)

\<meta\>

The \<meta\> element supplies the same information as the OLI \<log_action\> element (i.e., user ID, session ID, time, and time zone); it is redundant to supply this information twice. So if logging from inside the OLI course delivery system, you may omit the \<meta\> element; otherwise, you must include a \<meta\> element in the context message and all other messages you send.

Note

If your tutor appears within an OLI course, don't provide a \<meta\> element or a \<log_action\> wrapper; the OLI software supplies the relevant session information.

```
meta (0 or 1)
  user_id (1)
    anonFlag
  session_id (1)
  time (1)
  time_zone (1)
```
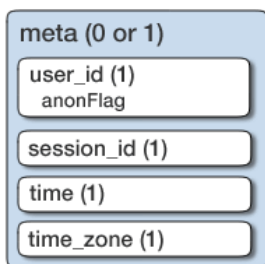
Figure 4. Structure of a \<meta\> element

Attributes of \<meta\>

None.

Child elements of \<meta\>

If you are supplying a meta element, you must supply its four child elements:

\<meta\>\<user_id\>
Required.

The user ID of the current user. A single Boolean attribute, anonFlag, specifies whether or not the supplied user ID has been anonymized. Its default value is false. If set to true, then DataShop will not re-anonymize the user ID.

\<meta\>\<session_id\>
Required.

A dataset-unique string that identifies the user's session with the tutor.

\<meta\>\<time\>
Required.

Local time; can be of the following formats:

- yyyy-MM-dd HH:mm:ss.SSS

- yyyy-MM-dd HH:mm:ss

- yyyy-MM-dd HH:mm

- MMMMM dd, yyyy hh:mm:ss a

- MM/dd/yy HH:mm:ss:SSS

- MM/dd/yy HH:mm:ss

\<meta\>\<time_zone\>
Required.

The local time zone name, such as one from the "TZ" column in this List of zoneinfo time zones [http://en.wikipedia.org/wiki/List_of_zoneinfo_time_zones]. Three-letter time zone abbreviations such as "EST" and "PST" are still valid, but are deprecated.

Example of \<meta\>

Here is our XML so far, including the meta element:

```
<?xml version="1.0" encoding="UTF-8"?>
<tutor_related_message_sequence
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='http://learnlab.web.cmu.edu/dtd/tutor_message_v4.xsd'
  version_number="4"
>
  <context_message
    context_message_id="0CEF2E07-24DE-BFDA-9BAB-957C3AE236CE"
    name="START_PROBLEM">
    <meta>
      <user_id>student01</user_id>
      <session_id></session_id>
      <time>2005/02/22 06:43:47.002</time>
      <time_zone>US/Eastern</time_zone>
    </meta>
  </context_message>
</tutor_related_message_sequence>
```

\<dataset\>
Required

The \<dataset\> element and its descendant elements describe the DataShop dataset to which the data should belong; the hierarchy of "levels" in which the current problem exists; and the current problem.
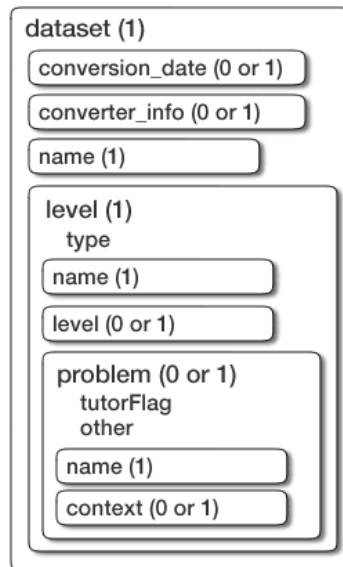

Figure 5. Structure of a \<dataset\> element

Attributes of <dataset>

None.

Child elements of <dataset>

- [conversion_date](#)

- [converter_info](#)

- [name](#) (required)

- [level](#) (required)

<dataset><conversion_date>

Optional.

If you are converting these data to the Tutor Message format from another format, include the date and, optionally, the time, that these data were converted.

Attributes of <conversion_date>

None.

<dataset><converter_info>

Optional.

If you are converting these data to the Tutor Message format from another format, include the name of the tool used to perform the conversion, the tool's version, and the date the tool was created or released.

Attributes of <converter_info>

None.

Example

```
<context_message ... >
  <dataset>
    <conversion_date>2007/06/29</conversion_date>
    <converter_info>Geometry Protocol Translator 4.11 June 27, 2007
    </converter_info>
    ...
  </dataset>
```

<dataset><name>

Required.

A <dataset> requires a single <name> element, the value of which will appear in the DataShop as the title of the dataset.

Note

Dataset <name> must be kept consistent—it determines the "bucket" in DataShop where your data are stored and displayed. If no name is provided, your data will fall into the default bucket, making it difficult to find later.

<dataset><level>

Required.

Nested <level> elements describe the curriculum hierarchy, a positioning of the problem within an organization you define. While nested <level> elements are common, only a single <level> is required.

Important

DataShop expects consistent levels—consistent both in terms of depth and type—throughout the dataset. For example, if a context message describes a unit that contains sections, other context messages in the dataset should also describe a unit-section hierarchy.
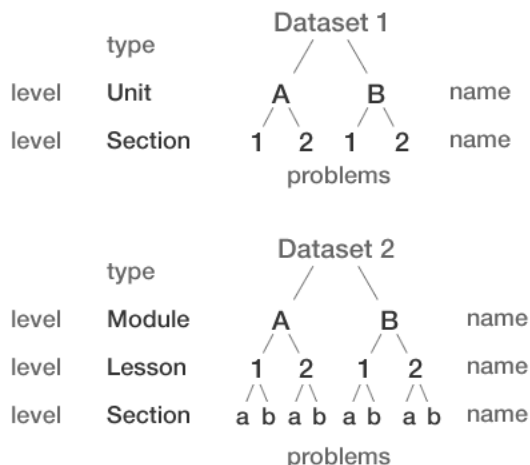
Figure 6. Unit-section hierarchy (top), and module-lesson-section hierarchy (bottom), each consistent across context messages for a dataset

Attributes of <level>

This element has a single attribute:

- `type`: the type of level, usually something like "unit", "section", "chapter", etc.

Child elements of <level>

A <level> is composed of a <name>, and either another <level> or a <problem>. This structure can either repeat —a nested `<level>` element—or a `<problem>` element can be included. There is no limit on the depth of the level hierarchy.

<dataset><level><name>
Required.

Provide a name for your level. For example, a name of "Understanding Fractions", when combined with a level type of "Section", produces "Section Understanding Fractions".

<dataset><level><problem>
Required at the bottom-most level.

The <problem> element contains both a <name> and a <context>.

Note

Only one <problem> element is expected within the entire <dataset> element, and it is expected to be within the deepest <level> element. Similarly, every <problem> must be placed within at least one <level>.

Attributes of <problem>

The <problem> element has two attributes: The tutorFlag attribute can be any of the following values:

- tutorFlag: can be any of the values specified in Table 2, "Recommended values for the tutorFlag attribute"

- other: a value not listed in Table 2, "Recommended values for the tutorFlag attribute"; stored only if tutorFlag is other.

Value of tutorFlag attribute	Description
tutor	Denotes a tutored problem.
test	Denotes a test problem.
pre-test	Denotes a pre-test problem.
post-test	Denotes a post-test problem.
other	Used to describe a problem that does not fit any of the above categories. If tutorFlag is other, then the other attribute must be filled in with application-specific data indicating where the problem falls in the spectrum between tutor and test. If tutorFlag is one of the other values in this table, then the other attribute is ignored.

Table 2. Recommended values for the tutorFlag attribute

Note

If tutorFlag is not set or empty, the default value will be tutor.

How DataShop considers problems to be the same or unique

For two given problem instances to be considered the same "problem" by DataShop, they must have the following identical elements and attributes:

- <name>

- <context>

- tutorFlag (attribute)

- other (attribute)

If any of these elements or attributes are different between the two instances, DataShop will not increment the "problem view" field, which tracks the number of instances of the problem the student saw.

Child elements of <problem>

<dataset><level><problem><name>
Required.

The name of the problem (e.g., "FractionAddition-1-2plus1-3").

<dataset><level><problem><context>
Optional.

The `<context>` element's value can be any string that provides more information—a text prompt to the student, a scenario, or other descriptor—but should not be HTML: HTML is not validated as part of the schema or by DataShop, and it has the potential to break the DataShop interface when viewed.

Example of <dataset>

The following code snippet defines a problem with a "unit-section" curriculum hierarchy. The problem "ChemPT1" falls within "Unit A", "Section One".

```
<?xml version="1.0" encoding="UTF-8"?>
<tutor_related_message_sequence
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='http://learnlab.web.cmu.edu/dtd/tutor_message_v4.xsd'
  version_number="4"
>
  <context_message context_message_id="0CEF2E07-24DE-BFDA-9BAB-957C3AE236CE"
    name="START_PROBLEM">
    <dataset>
      <name>Stoichiometry Study 1 - Spring 2005</name>
      <level type="Unit">
        <name>A</name>
        <level type="Section">
          <name>One</name>
          <problem>
            <name>ChemPT1</name>
            <context>Chemistry Problem One</context>
          </problem>
        </level>
      </level>
    </dataset>
  </context_message>
</tutor_related_message_sequence>
```

<class>
Optional.

The `<class>` element is used to describe the context of tutor usage in a school.

Of its child elements, only `<instructor>` can occur more than once.

Note

If class name is not filled in, but period is, then period will be used as the class name. If class name and period name are not filled in, but description is, then the description will be used as the class name.
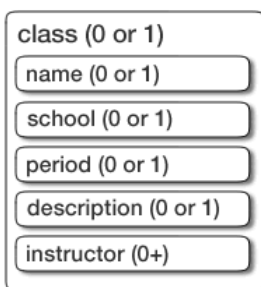
Figure 7. Structure of a <class> element

Child elements of <class>

<class><name>
Optional.

The name of the class in which the tutor is used (eg, "Intro to Chemistry").

\<class\>\<school\>
      Optional.

The school in which the tutor is used (eg, "Perrysville Elementary").

\<class\>\<period\>
      Optional.

The class period in which the data are collected (eg, "5").

\<class\>\<description\>
      Optional.

A description of the class in which the tutor is used.

\<class\>\<instructor\>
      Optional.

The class instructor(s). If multiple, use an `<instructor>` element to name each instructor.

\<condition\>
      Optional.

A `<condition>` describes a study condition, in the case that these data are being collected in the context of a research study.

Zero or more `<condition>` elements are allowed.

condition (0+)
name (1)
type (0 or 1)
desc (0 or 1)
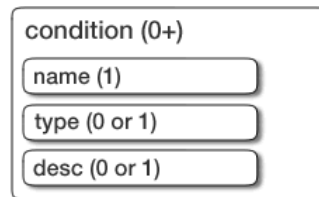
Figure 8. Structure of a \<condition\> element

Child elements of \<condition\>

If a `<condition>` is included, it must have a `<name>`, which cannot be an empty element. Optionally, `<condition>` can include a `<type>` and `<desc>`, or description.

\<condition\>\<name\>
      Required.

The name of the condition (eg, "Unworked").

Important

In DataShop, condition name must be unique within a dataset, so do not rely on `<type>` or `<desc>` elements to distinguish conditions.

\<condition\>\<type\>
      Optional.

A condition classification (eg, "Experimental", "Control").

\<condition\>\<desc\>
Optional.

A description of the condition.

\<skill\>
Optional.

A `<skill>` in a context message is a description of a knowledge component that may be exercised by the student in the course of solving the problem or activity. It's primary purpose is to describe the probability of the student knowing the skill at the *beginning* of the problem. This probability is commonly used to track student mastery of knowledge components ("mastery learning") through a knowledge-tracing algorithm.

Note

A `<skill>` element contained in the context message is currently not imported by DataShop.

For a description of the skill content model, see the section called "\<skill\>" in tutor_message.

\<custom_field\>
Optional.

Use this element to describe other contextual information not adequately captured by the other context message elements.

Zero or more `<custom_field>` elements are permissible.

Note

While DataShop does not use or show this information in its reports, the content of a `<custom_field>` element is both stored and included in data exports.

```
custom_field (0+)
  name (1)
  value (1)
```
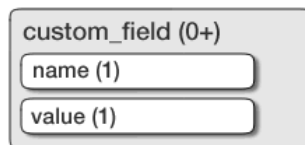
Figure 9. Structure of a \<custom_field\> element

Child elements of \<custom_field\>

\<custom_field\>\<name\>
Required.

A name for the custom field.

\<custom_field\>\<value\>
Required.

A value for the custom field.

Example of \<custom_field\>

```
<context_message ... >
 ...
  <custom_field>
    <name>System-Equation</name>
```

```
    <value>y=6x-8</value>
  </custom_field>
 ...
</context_message>
```

<dfa>

Optional.

DFA is an element that describes the results of a difficulty factors analysis. It has no attributes and contains no child elements. This element is not used by DataShop.

2.5. <tool_message>

Representing student action

While the <context_message> places the actions of the student in context of the curriculum, the <tool_message> captures the actions of the student. Create a <tool_message> whenever the student interacts with the tool or the tool itself performs some action.
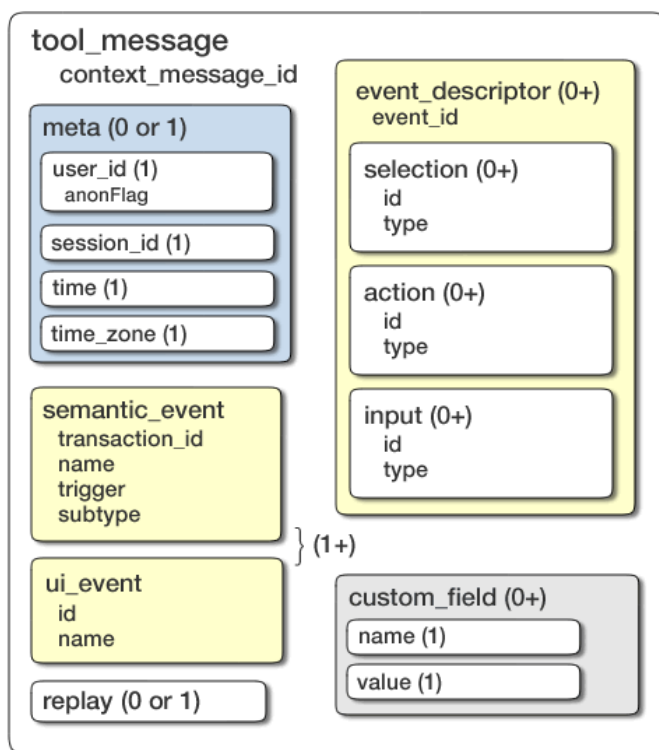
Figure 10. Structure of a <tool_message> element

2.5.1. Attributes of <tool_message>

<tool_message> has a single, required attribute: context_message_id. The value of this attribute must be the same for all tool and tutor messages described by the opening <context_message>, and it should be identical to the one specified in the <context_message>.

2.5.2. Child elements of <tool_message>

• meta

- event_descriptor

- semantic_event (required)

- ui_event

- custom_field

- replay (unused by DataShop)

\<meta\>

The content model of the `<meta>` element is identical to the element of the same name contained in the `<context_message>`. It is described in the section called "\<meta\>".

\<event_descriptor\>
Optional.

This element describes the specifics of the student's interaction with the tool, primarily the selection, action, and input. It contains the details of a single observable change in the state of the user interface. This change in state is usually the result of a student action, but a tutor could cause a change in tool state as well. In either case, the tool message is used.
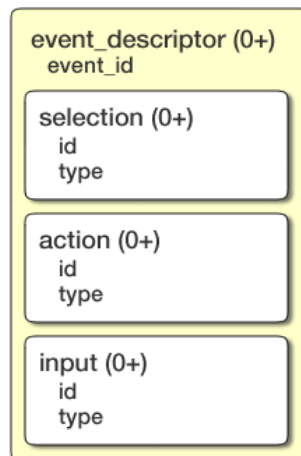
Figure 11. Structure of an \<event_descriptor\> element

Note

The selection, action, and input (SAI) used in an `<event_descriptor>` element should depend on whether the `<event_descriptor>` is appearing in a tool message or tutor message, and what the value of the `<semantic_event>` name attribute is. For example:

- the SAI for a `HINT_REQUEST` semantic event in a tool message specifies the hint button

- the SAI for a `HINT_MSG` semantic event in a tutor message specifies what the student should have done

- the SAI for a `ATTEMPT` semantic event in a tool message specifies what the student actually did

- the SAI for a `RESULT` semantic event in a tutor message specifies what the student should have done

Attributes of <event_descriptor>

- `event_id`: an identifier for the event. This attribute is not used by DataShop and should not be used.

Child elements of <event_descriptor>

<event_descriptor><selection>
Optional.

A description of the interface element that the student selected or interacted with (for example, "LowestCommonDenominatorCell"). It should be unique within the user interface so that the student's action is identifiable. Some actions will involve multiple selections (see the section called "Example of <event_descriptor>" for an example of this usage).

Attributes of <selection>
Optional.

- `id`: unique identifier; can be used when multiple selection-action-inputs (SAIs) are present to tie them together.

- `type`: the role this particular selection plays with respect to the other SAIs.

<event_descriptor><action>
Optional.

A description of the manipulation applied to the selection. Many interface widgets have a single action, but multiple actions are also possible.

Attributes of <action>
Optional.

- `id`: unique identifier; can be used when multiple selection-action-inputs (SAIs) are present to tie them together.

- `type`: the role this particular action plays with respect to the other SAIs.

<event_descriptor><input>
Optional.

The input the student submitted (e.g., the text entered, the text of a menu item or a combobox entry).

Attributes of <input>
Optional.

- `id`: unique identifier; can be used when multiple selection-action-inputs (SAIs) are present to tie them together.

- `type`: the role this particular input plays with respect to the other SAIs.

Example of <event_descriptor>

The following is an example of a tool message from ChemCollective's Virtual Lab (VLab).

```
<tool_message context_message_id="0CEF2E07">
   <meta>
      <user_id>A1371</user_id>
      <session_id>148852539062A137120</session_id>
      <time>2005-17-9 5:16:42</time>
```

```
        <time_zone>US/Eastern</time_zone>
    </meta>
    <semantic_event transaction_id="B503948-9164-DD83"
        name="SOLUTION_SET_THERMAL" />
    <event_descriptor>
        <selection type="flaskID">2500mL Bottle (ID2)</selection>
        <selection type="flaskName">2500mL Bottle</selection>
        <selection type="flaskTemp">303.15K</selection>
        <selection type="flaskInsulation">false</selection>
        <action>SOLUTION_SET_THERMAL</action>
        <input>303.15</input>
    </event_descriptor>
</tool_message>
```

\<semantic_event>

One <semantic_event> or one or more <ui_event> elements required.

The `<semantic_event>` element describes a high-level, meaningful event, as opposed to a low-level, non-semantic event, which should be captured in the `<ui_event>` element. This notion of separating UI events and semantic events is taken from the proposed IEEE LTSC specification (Ritter).

This element also occurs in the tutor_message element.

Note

`<semantic_event>` is a *content-less* element: all information is provided in the attributes—not the content—of the element.

Note

DataShop expects either a single `<semantic_event>` with a single `<event_descriptor>`, or one or more `<ui_event>` elements, but not both.

```
semantic_event
  transaction_id
  name
  trigger
  subtype
```
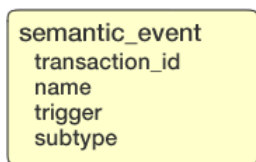
Figure 12. Structure of a \<semantic_event> element

Attributes of \<semantic_event>

- `transaction_id`: a string that uniquely identifies the event within the session. If this tool message is intended to be paired with a tutor message, the `transaction_id` value should correspond with the one defined in the tutor message.

- `name`: a semantic description of the event. When exported by DataShop, the value is shown in "Student Response Type" column.

Note

DataShop expects a variety of semantic event names in tool messages and correspondingly paired event names in tutor messages. Though you can enter any event name, conforming to one of the existing pairs may be more useful for analysis in DataShop.

Current pairs include:

- `ATTEMPT` and `RESULT`

- `HINT_REQUEST` and `HINT_MSG`

In these pairs, the first value would appear in the `name` attribute of the `<semantic_event>` in the *tool* message, while the second value would appear in the same location in the *tutor* message.

For an example of this usage, see the section called "Examples of <semantic_event>" below.

- `trigger`: The agent that caused the change in state within the tool. Should be `USER` if the user caused (or performed) the action; `DATA` if the tool caused (or performed) the action. This attribute is not imported by DataShop.

- `subtype`: A further classification for this semantic event. For example, the CTAT software describes tool-performed actions as having `subtype tutor-performed` and `trigger data` (for an XML example, see the section called "Examples of <semantic_event>"). When exported by DataShop, this attribute value is shown in the "Student Response Subtype" column.

Examples of <semantic_event>

Corresponding semantic event elements in tool and tutor messages.

```
<tool_message context_message_id="C2badc36e:113e3ba9c5c:-7fe5">
  <problem_name>kl</problem_name>
  <semantic_event transaction_id="T2badc36e:113e3ba9c5c:-7fe7"
    name="ATTEMPT" />
  <event_descriptor>
    <selection>dorminMultipleChoice1</selection>
    <action>UpdateMultipleChoice</action>
    <input>Option0</input>
  </event_descriptor>
</tool_message>

<tutor_message context_message_id="C2badc36e:113e3ba9c5c:-7fe5">
  <problem_name>kl</problem_name>
  <semantic_event transaction_id="T2badc36e:113e3ba9c5c:-7fe7"
    name="RESULT" />
  <event_descriptor>
    <selection>dorminMultipleChoice1</selection>
    <action>UpdateMultipleChoice</action>
    <input>Option2</input>
  </event_descriptor>
  <action_evaluation>INCORRECT</action_evaluation>
</tutor_message>
```

Tutor-performed action as logged by CTAT. In this example, the tutor performs two steps for the student.

```
<tool_message context_message_id="C-793fdeed:11489b4f0f8:-7f9e">
  <problem_name>13-26</problem_name>
  <semantic_event transaction_id="T-793fdeed:11489b4f0f8:-7f6e"
    name="ATTEMPT" trigger="DATA" subtype="tutor-performed" />
  <event_descriptor>
    <selection>convertDenom2</selection>
    <action>UpdateTextField</action>
    <input>6</input>
  </event_descriptor>
</tool_message>

<tool_message context_message_id="C-793fdeed:11489b4f0f8:-7f9e">
  <problem_name>13-26</problem_name>
  <semantic_event transaction_id="T-793fdeed:11489b4f0f8:-7f68"
```

```
         name="ATTEMPT" trigger="DATA" subtype="tutor-performed" />
    <event_descriptor>
      <selection>unreducedDenom</selection>
      <action>UpdateTextField</action>
      <input>6</input>
    </event_descriptor>
</tool_message>
```

<ui_event>

One <semantic_event> or one or more <ui_event> elements required.

Use this element to log low-level user interface events that are non-semantic. For example, you could use a <ui_event> to describe a mouse click; a special format for machine processing (aimed at fine-grained reproduction during playback); or a description intended for human readers.

Note

DataShop imports only the first <ui_event> element if more than one are contained in a single tool message. DataShop uses the name attribute of the <ui_event> as the "Student Response Type" and sets the "Student Response Subtype" to "UI Event".

```
ui_event
  id
  name
```
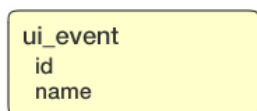
Figure 13. Structure of a <ui_event> element

Attributes of <ui_event>

- id: an identifier for the event. This attribute is not used by DataShop and should not be used.

- name: a classification for the event

Example of <ui_event>

```
<tool_message context_message_id="0CEF2E07">
   <ui_event name="VLAB_RESIZE ">The virtual Lab is resized to a width
     of 780 and a height of 550</ui_event>
</tool_message>
```

<custom_field>

Optional.

A tool message may include zero or more <custom_field> elements. See the section called "<custom_field>" for details on the content model of this element.

<replay>

Optional.

The <replay> element's content model allows for any content (character data or other elements). It is intended as a free-form area where a tool (such as the Chemistry VLab) can put extra information so that these logs can be fed back into the tool for replay.

Note

DataShop does not import the content of the <replay> element.

2.6. <tutor_message>

Representing tutor response

A <tutor_message> commonly describes an evaluation of a student action. It is similar to a <tool_message>, but contains additional information about the correctness of the student action.

Construct a <tutor_message> whenever the application evaluates a student action.
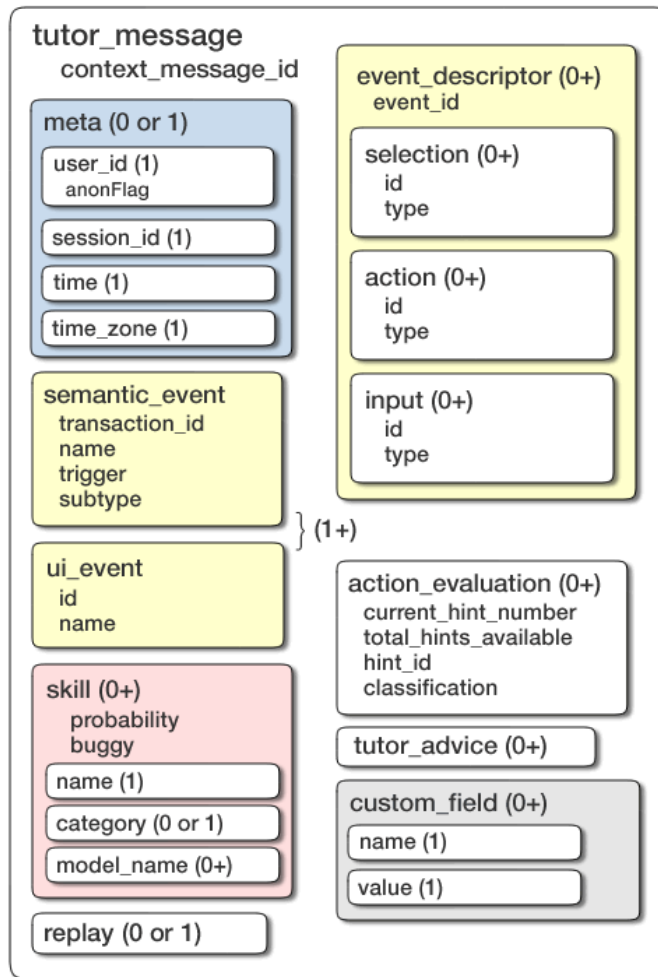
Figure 14. Structure of a <tutor_message> element

2.6.1. Attributes of <tutor_message>

<tutor_message> has a single, required attribute: context_message_id. The value of this attribute must be the same for all tool and tutor messages described by the opening <context_message>, and it should be identical to the one specified in the <context_message>.

2.6.2. Child elements

Many child elements in a <tutor_message> are structurally identical to the same elements in a <tool_message>, but serve different purposes. For example, <event_descriptor> in a

`<tutor_message>` should describe the *desired action on the step*. Similarly, `<semantic_event>` should describe the purpose of the tutor's response.

The elements described in `<tool_message>` that also appear in `<tutor_message>` are:

- event_descriptor

- semantic_event (required)

- ui_event (unused by DataShop)

- custom_field

- replay (unused by DataShop)

In addition, the following elements can appear in a tutor message:

- action_evaluation

- skill

- tutor_advice

- interpretation (do not use)

<action_evaluation>

Optional.

An action evaluation is the result of the tutor's evaluation of a student's action. The evaluation value is expected as the text content of this element.

Zero or more `<action_evaluation>` elements are permissible; however, only one is expected and used/imported by DataShop.
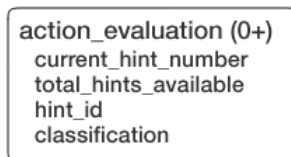
Figure 15. Structure of an <action_evaluation> element

DataShop expects the content of the `<action_evaluation>` element to be one of the values defined in Table 3, "Recommended values for the <action_evaluation> element". Any other value will become `"unknown"` in the DataShop Error Report.

Description	Preferred	Other common values
Student's action was correct	CORRECT	OK
Student's action was incorrect	INCORRECT	BUG ERROR
Student requested a hint	HINT	HELP GLOSSARY_ITEM

Table 3. Recommended values for the <action_evaluation> element

Attributes of <action_evaluation>

<action_evaluation> has four attributes:

- classification: the type of error (e.g., "sign error") or type of hint.

- current_hint_number: only used if the <action_evaluation> is HINT

- total_hints_available: only used if the <action_evaluation> is HINT

- hint_id: a unique ID of the hint; only used if <action_evaluation> is HINT. Can be used to identify hints that are all the same but have different variables filled in depending on the problem.

<skill>
Optional.

The <skill> element is used to associate a knowledge component with a tutor response (and optionally, to provide updates to a knowledge-tracing system that tracks the probability of a student mastering a skill.)

Zero or more <skill> elements are allowed.
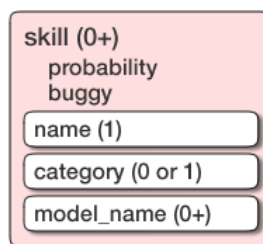
Figure 16. Structure of a <skill> element

Attributes of <skill>

The <skill> element has two attributes:

- probability: the probability that the student knows the skill

- buggy: whether or not the skill associated with this tutor response is a "buggy" skill—this is a student misconception modeled or identified by the tutoring system. Can be true or false.

> **Caution**
>
> DataShop does not import either of these attributes.

Child elements of <skill>

<skill><name>
Required.

The name of the skill (eg, "write-density").

> **Note**
>
> DataShop expects each skill name to be unique within a skill model (<model_name>).

\<skill>\<category>
> Optional.

> A category that contains the skill (eg, "fraction-addition").

\<skill>\<model_name>
> Optional.

> The name of a [knowledge component model](https://pslcdatashop.web.cmu.edu/help?page=terms#kc_model) [https://pslcdatashop.web.cmu.edu/help?page=terms#kc_model], which maps steps to knowledge components.

> You can specify more than one model (multiple `<model_name>` elements), as the specified skill can exist in more than one model. Alternatively, you can create separate skill elements with the same skill name that specify different model names.

> If `<model_name>` is empty or not defined, DataShop will use the value `"Default"`.

\<tutor_advice>
> Optional.

> This element captures the body of a hint, success, or error message shown to the student. It is generally a text value.

tutor_advice (0+)

Figure 17. Structure of a \<tutor_advice> element

Example of \<tutor_advice>

```
<tutor_message ...>
  ...
  <action_evaluation>INCORRECT</action_evaluation>
  <tutor_advice>Please work on the highlighted step.</tutor_advice>
</tutor_message>
```

\<interpretation>

An interpretation is a tutor's plausible explanation for a student's action. It is an optional element. With some tutors, multiple interpretations exist for a single student action. These are competing explanations, and there should be only one "chosen" interpretation, signified via the `chosen` attribute.

Caution

> This element is likely to change in future versions of the Tutor Message format and should not be used.

Child elements

Its child elements are `<correct_step_sequence>` and `<incorrect_step_sequence>`, both of which have an attribute `ordered`, which can be either "true" or "false". When a tutor that provides interpretations evaluates a student action as correct, there is most likely a single interpretation with a `<correct_step_sequence>`. When that evaluation is of an incorrect action, there is likely to be both an `<incorrect_step_sequence>` and a `<correct_step_sequence>`.

2.7. \<message>

The `<message>` element should only be used for capturing arbitrary communications content not adequately captured by the tutor, tool, or context message types. It is currently used by CTAT for the purposes of playback.
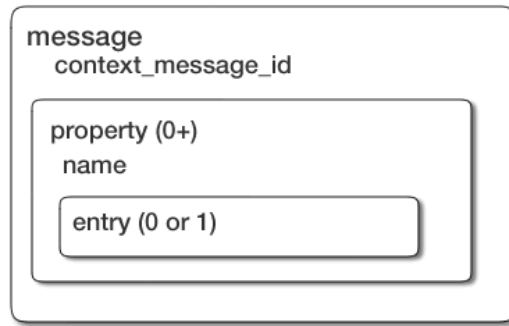
Figure 18. Structure of a <message> element

3. Media Logging

Media logging is the process of describing student interactions with media, both within and outside of tutored activities, using the Tutor Message format.

This section of the guide describes this standard form for logging media events using the Tutor Message framework. It assumes you are familiar with the tutor message format, described in Section 2, "The Tutor Message format".

3.1. Similarity between media logging and tutor logging

In media logging, the existing message types—the context, tool, and tutor messages—are used to describe user interactions with media objects.

Similar to logging in which no media objects are present, tool and tutor messages are always preceded by a context message. The entire series of messages is also contained in a single element, the `<tutor_related_message_sequence>`. But unlike logging in which no media is present, media logs are composed primarily of tool messages; tutor messages are only required when a media event is explicitly tutored.

3.2. <context_message> (media logging)

The context message establishes the context for a following sequence of tool and tutor messages. Please refer to Section 2.4, "<context_message>" for primary details on the context message if you are not familiar with it.

An example of a complete context message for video logging is included at the end of this section.

3.2.1. Media embedded within a tutor

If the media is embedded within a tutor, the context message should already be logged for the problem (see Section 2.4, "<context_message>") and the `<context_message>` name attribute will be `START_PROBLEM`.

3.2.2. Media not embedded within a tutor

If the media is not embedded within a tutor, then a context message should be logged per media object and the `name` attribute of a `<context_message>` element should be one of the values from the following table:

Value	Description
`LOAD_VIDEO`	Should be logged when the video is actually loaded on the page
`LOAD_AUDIO`	Should be logged when the audio is actually loaded on the page
`LOAD_MEDIA`	Use if neither video or audio (e.g., vector-based animation). Should be logged when the media is actually loaded on the page

Value	Description
START_VIDEO	Should be logged when the user first interacts with the video
START_AUDIO	Should be logged when the user first interacts with the audio
START_MEDIA	Use if neither video or audio (e.g., vector-based animation). Should be logged when the user first interacts with the media

Table 4. Recommended values for the <context_message> name attribute for a media object

Note

We recommend that you log a context message describing the load event (when the media object finishes loading) and, later, a tool message when the user first interacts with the media object. Alternatively, you may choose to omit logging the load event: in this case, you should still log a context message before the tool message, but its name attribute value will start with "START". For XML examples of these approaches see Example A.2, "Media logging: Logging the load video event (media not in a tutor)" and Example A.3, "Media logging: Logging the start video event (no load event) (media not in a tutor)".

In addition, the problem's <name> element (contained in the dataset <level> hierarchy) should contain a unique identifier for that media object. DataShop recommends that the file name of the media be used as the problem name.

Note

This use of the <problem> element may change in future versions of the Tutor Message format to more accurately indicate the type of information being presented to the student.

3.2.3. XML examples of <context_message>

Example 1. Context message for a video clip not embedded in a problem/tutor

The following is an example of a context message for a video clip *not embedded* in a problem/tutor.

```
<context_message
   context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
   name="LOAD_VIDEO">
   <dataset>
      <name>Example Media Dataset</name>
      <level type="unit">
         <name>Stoichiometry</name>
         <level type="section">
            <name>What are moles?</name>
            <problem>
               <name>mymovie.flv</name>
            </problem>
         </level>
      </level>
   </dataset>
</context_message>
```

Example 2. Context message for a video clip embedded in a problem/tutor

The following is an example of a context message for a video clip *embedded* in a problem/tutor.

```
<context_message
   context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
   name="START_PROBLEM">
   <dataset>
      <name>Learn a Language Fall 2007</name>
      <level type="unit">
         <name>Learning Logging</name>
         <problem><name>Translating Tech Talk</name></problem>
      </level>
   </dataset>
</context_message>
```

3.3. <tool_message> (media logging)

While the context message places the actions of the student in context of the curriculum, the tool message captures the actions of the student. Generate a `<tool_message>` whenever the student interacts with the media or the tutor itself performs some action on the media. This will capture whenever media is stopped, started, muted, etc.

A tool message in media logging is structurally the same as a tool message for tutor logging. See Section 2.5, "<tool_message>" for a description of the tool message content model. In addition, the following constraints exist.

3.3.1. <semantic_event> (media logging)

Each tool message must include a `<semantic_event>` element. This element contains the `transaction_id`, which must be unique within the context message for each tool message. If this is a tutored event, a subsequent `<tutor_message>` will be paired with the tool message by this `transaction_id`.

Note

We decided that the media event is semantically important and as such we use the `<semantic_event>` element instead of the `<ui_event>` element. The <ui_event> element is for capturing events that may not be important such as mouse movements and key strokes.

The `name` attribute of the `<semantic_event>` is dependent on whether or not the event is tutored. A tutored event uses the standard `ATTEMPT` value; an untutored event should use a value from the following table:

name attribute value	Description
VIDEO_ACTION	Used for an action on a video object.
AUDIO_ACTION	Used for an action on an audio object.
MEDIA_ACTION	Used for an action on another type of media object.

Table 5. Recommended values for the <semantic_event> name attribute for a media object

The `<semantic_event>` element has two attributes which are worth noting here: the `trigger` and the `subtype`. The trigger should be set to either of the following:

trigger attribute value	Description
USER	If the user initiates the event.

trigger attribute value	**Description**
DATA	If the tool or tutor initiates the event.

Table 6. Recommended values forthea <semantic_event> trigger attribute for a media object

The subtype attribute is more of a free-form field that further qualifies the type of event. It can be set to something like tutor-performed. The value is stored in the DataShop database and exported, but no reasoning is dependent on this value.

3.3.2. <event_descriptor> (media logging)

Included in each tool message is the <event_descriptor> element which provides the details of the semantic event: it describes the specifics of the student's interaction with the tool. This should be a measurable change of state in the tool. It can be either a student action or an action taken for the student by the tool (see Section 3.3.1, "<semantic_event> (media logging)" for how to signify events initiated by the tutor).

For media logging, the event descriptor will include the action performed, the time(s) it occurred within the clip, and which interface elements were part of the interaction. Place these in the <selection>, <action>, <input> elements, each with a type attribute to further clarify the information those elements contain. The recommended list of media actions along with their expected selections and inputs are as follows:

Selections	Action	Input(s)	Notes
selection media_file clip_length	play	time	start/resume playback of clip; time is location in clip where playback was started
selection media_file clip_length	pause	time	time is location in clip where playback was paused
selection media_file clip_length	stop	time	time is location in clip where playback was stopped
selection media_file clip_length	close	time	time is location in current clip when media was closed
selection media_file clip_length	end	time	the end of the clip has been reached and playback has stopped; time should be the same as clip length
selection media_file clip_length	cue	start_cue end_cue	start_cue is the time in the clip when the starting cue point was added; end_cue is the time in the clip when the ending cue point was added
selection media_file clip_length	mute	time	time is location in clip when this action occurred
selection media_file	unmute	time	time is location in clip when this action occurred

Selections	Action	Input(s)	Notes
clip_length			
selection media_file clip_length	volume_change	time start_level final_level	time is location in clip when this action occurred. start_level is the volume level before the change, and final_level is the resulting volume level.
selection media_file clip_length	next_clip	time	move to the next media clip; time is location in current clip when this action occurred
selection media_file clip_length	prev_clip	time	move to the previous media clip; time is location in current clip when this action occurred

Table 7. Format of an <event_descriptor> selections, action, and input values for a media object

Every action is expected to have three selections. The first selection is the widget, button, or component that the student actually clicked on. The second is the name of the media file, and the third is the length of that file.

Example 3. Three selections for every media action

```
<selection>_level0.VideoPlayerInstance1.sliderButtonName</selection>
<selection type="media_file">mymovie.flv</selection>
<selection type="clip_length">00:08:34.823</selection>
```

The `<input>` elements will be used to indicate the media timing. Similar to `<selection>` elements the `type` attribute will be used to differentiate the different inputs. The inputs will vary by the action as noted in the table above. All time values should be in the form `HH:MM:SS.SSS`

Example 4. Example input

```
<input type="time">00:02:34.243</input>
```

3.3.3. XML Examples of tool messages for logging media events

The following are examples of typical media event log messages.

Example 5. Tool message for "play"

```
<tool_message
   context_message_id ="02CE3AE5-F6D5-9177-913F-C34730F1096C">
   <semantic_event
      transaction_id="1F3A9B23-9164-DD83-EBB2-1589FD38D4B3"
      name="VIDEO_ACTION" />
   <event_descriptor>
        <selection>_level0.VideoPlayerInstance1.sliderButtonName</selection>
        <selection type="media_file">mymovie.flv</selection>
        <selection type="clip_length">00:08:00.0</input>
        <action>play</action>
        <input type="time">00:02:34.2</input>
   </event_descriptor>
</tool_message>
```

Example 6. Tool message for "stop"

```
<tool_message
   context_message_id ="02CE3AE5-F6D5-9177-913F-C34730F1096C">
   <semantic_event
      transaction_id="A43B003-9164-DD83-EBB2-1589FD38D435"
      name="VIDEO_ACTION" />
   <event_descriptor>
      <selection>_level0.VideoPlayerInstance1.sliderButtonName</selection>
      <selection type="media_file">mymovie.flv</selection>
      <selection type="clip_length">00:08:00.0</input>
      <action>stop</action>
      <input type="time">00:05:32.2</input>
   </event_descriptor>
</tool_message>
```

Example 7. Tool message for "cue"

```
<tool_message
   context_message_id ="02CE3AE5-F6D5-9177-913F-C34730F1096C">
   <semantic_event
      transaction_id="B503948-9164-DD83-EBB2-1589FD38D435"
      name="VIDEO_ACTION" />
   <event_descriptor>
      <selection>_level0.VideoPlayerInstance1.sliderButtonName</selection>
      <selection type="media_file">mymovie.flv</selection>
      <selection type="clip_length">00:08:00.0</input>
      <action>cue</action>
      <input type="start_cue">00:04:34.8</input>
      <input type="stop_cue">00:05:42.2</input>
   </event_descriptor>
</tool_message>
```

Bibliography

Ritter, S. & Blessing, S.B. (1998). Authoring Tools for Component-Based Learning Environments. *The Journal of Learning Sciences, 7*(1), 107-132.

Ritter, S. & Koedinger, K. R. (1996). An architecture for plug-in tutor agents. *Journal of Artificial Intelligence in Education, 7*(3-4), 315-347.

Glossary

CTAT	Cognitive Tutor Authoring Tools [http://ctat.pact.cs.cmu.edu] software. As of version 2.1, CTAT logs to the format described in this document.
DTD	Document Type Definition. The rules describing the structure of a class of XML markup. In the PSLC community, the "tutor message DTD" is a reference to the logging format described in this document. Going forward, "XML Schema", a successfor to DTD, will be used by the PSLC to define the tutor message format.
OLI	Carnegie Mellon's Online Learning Initiative [http://www.cmu.edu/oli/], a project that specializes in creating online education and courses based on "cognitive theory, formative evaluation for students and faculty, and iterative course improvement based on empirical evidence" (http://

www.cmu.edu/oli/overview/index.html). Its software is the deployment platform for PSLC courses and studies.

s-a-i   Selection, action, input triple. Format for referring to a component of a user interface (selection), the action performed on it (action), and the student input into that component (input).

session   An instance of student use of the tutor, initiated by student log-in into a system.

XML Schema   An XML language used to describe the rules governing the structure and content of an XML document. The tutor message format is described by an XML Schema document (an XSD), which replaces the DTD.

A. XML Examples

Example A.1. Tool message

```
<tool_message context_message_id="C1f3b4f3a:111c34b00ed:-7fff">
   <meta>
      <user_id>joe_cool</user_id>
      <session_id>JUNK1f3b4f3a:111c34b00ed:-8000</session_id>
      <time>2006-08-30 13:22:33</time>
      <time_zone>EST</time_zone>
   </meta>
   <problem_name>ChemPT1</problem_name>
   <semantic_event transaction_id="T1f3b4f3a:111c34b00ed:-7ffe" name="ATTEMPT"/>
   <event_descriptor>
      <selection>ButtonOne</selection>
      <action>PressButton</action>
      <input>box</input>
   </event_descriptor>
   <custom_field>
      <name>Equation</name>
      <value>y=x+ab</value>
   </custom_field>
</tool_message>
```

Example A.2. Media logging: Logging the load video event (media not in a tutor)

In this example, a context message is logged when the video media is first loaded (signified with `name="LOAD_VIDEO"`). Then when the user first interacts with the video, a tool message is logged. For comparison, see the alternative logging approach in the next example.

```
<context_message
   context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
   name="LOAD_VIDEO">
   <dataset>
      <name>Example Media Dataset</name>
      <level type="unit">
         <name>Stoichiometry</name>
         <level type="section">
            <name>What are moles?</name>
            <problem>
               <name>mymovie.flv</name>
            </problem>
         </level>
      </level>
   </dataset>
</context_message>

<tool_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
    <semantic_event
        transaction_id="1F3A9B23-9164-DD83-EBB2-1589FD38D4B3"
        name="VIDEO_ACTION" />
    <event_descriptor>
        <selection>
        _level0.VideoPlayerInstance1.sliderButtonName
        </selection>
        <selection type="media_file">mymovie.flv</selection>
        <selection type="clip_length">00:08:00.0</input>
        <action>play</action>
        <input type="time">00:02:34.2</input>
    </event_descriptor>
</tool_message>
```

Example A.3. Media logging: Logging the start video event (no load event) (media not in a tutor)

In this example, a context message is logged when the the user first interacts with the video (signified with name="START_VIDEO"). At approximately the same time, a tool message is also logged to capture the details of the interaction. For comparison, see the alternative logging approach in the preceding example.

```
<context_message
   context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
   name="START_VIDEO">
   <dataset>
      <name>Example Media Dataset</name>
      <level type="unit">
         <name>Stoichiometry</name>
         <level type="section">
            <name>What are moles?</name>
            <problem>
               <name>mymovie.flv</name>
            </problem>
         </level>
      </level>
   </dataset>
</context_message>

<tool_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
    <semantic_event
        transaction_id="1F3A9B23-9164-DD83-EBB2-1589FD38D4B3"
        name="VIDEO_ACTION" />
    <event_descriptor>
        <selection>
        _level0.VideoPlayerInstance1.sliderButtonName
        </selection>
        <selection type="media_file">mymovie.flv</selection>
        <selection type="clip_length">00:08:00.0</input>
        <action>play</action>
        <input type="time">00:02:34.2</input>
    </event_descriptor>
</tool_message>
```

Example A.4. Media logging: tutored event

The following message sequence is an example of tutored media event.

```
<context_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
    name="START_PROBLEM">
    <dataset>
        <name>Learn a Language Fall 2007</name>
        <level type="unit">
            <name>Learning Logging</name>
            <problem><name>Translating Tech Talk</name></problem>
        </level>
    </dataset>
</context_message>

<tool_message
    context_message_id ="02CE3AE5-F6D5-9177-913F-C34730F1096C">
    <semantic_event
        transaction_id="B503948-9164-DD83-EBB2-1589FD38D435"
        name="ATTEMPT" />
    <event_descriptor>
        <selection>_level0.VideoPlayerInstance1.sliderButtonName</selection>
        <selection type="media_file">mymovie.flv</selection>
        <selection type="clip_length">00:08:00.0</input>
        <action>cue</action>
        <input type="start_cue">00:04:34.8</input>
        <input type="stop_cue">00:05:42.2</input>
    </event_descriptor>
</tool_message>

<tutor_message
    context_message_id ="02CE3AE5-F6D5-9177-913F-C34730F1096C">
    <semantic_event
        transaction_id="B503948-9164-DD83-EBB2-1589FD38D435"
        name="RESULT" />
    <event_descriptor>
        <selection>_level0.VideoPlayerInstance1.sliderButtonName</selection>
        <selection type="media_file">mymovie.flv</selection>
        <selection type="clip_length">00:08:00.0</input>
        <action>cue</action>
        <input type="start_cue">00:04:34.8</input>
        <input type="stop_cue">00:05:42.2</input>
    </event_descriptor>
    <action_evaluation>INCORRECT</action_evaluation>
    <tutor_advice>Your answer is not correct.  Select only the portion of the
        video where the man it talking about his family.</tutor_advice>
    <skill>
        <name>family_words</name>
        <category>video_portion_selection</category>
    </skill>
</tutor_message>
```

Example A.5. Media logging: two untutored events followed by a single tutored event

The following message sequence is an example of two untutored media events followed by a single tutored media event.

```
<context_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
    name="START_PROBLEM">
    <dataset>
        <name>Learn a Language Fall 2007</name>
        <level type="unit">
            <name>Learning Logging</name>
            <problem><name>Translating Tech Talk</name></problem>
```

```
         </level>
      </dataset>
</context_message>


<tool_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
    <semantic_event
        transaction_id="1F3A9B23-9164-DD83-EBB2-1589FD38D4B3"
        name="AUDIO_ACTION" />
    <event_descriptor>
        <selection>_level0.MS_component.item4</selection>
        <selection type="media_file">u1_m1_s5_ss3_nasvow_0268.mp3</selection>
        <selection type="clip_length">0.758</selection>
        <action>play</action>
        <input type="time">0</input>
    </event_descriptor>
</tool_message>


<tool_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C"
    <semantic_event
        transaction_id="1F3A9B23-9164-DD83-EBB2-1589FD38D4B3"
        name="AUDIO_ACTION" />
    <event_descriptor>
        <selection>_level0.MS_component.item4</selection>
        <selection type="media_file">u1_m1_s5_ss3_nasvow_0268.mp3</selection>
        <selection type="clip_length">0.758</selection>
        <action>end</action>
        <input type="time">0.758</input>
    </event_descriptor>
</tool_message>


<tool_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C">
    <semantic_event
        transaction_id="7775F27F-7AE4-0F70-8E62-76249B32DB06" name="ATTEMPT" />
    <event_descriptor>
        <selection>MS_component</selection>
        <action>MultipleSelectionButtonPressed</action>
        <input>Not checked for choice 1: Not checked for choice 2:
           Not checked for choice 3: Not checked for choice 4:
           Not checked for choice 5</input>
    </event_descriptor>
</tool_message>


<tutor_message
    context_message_id="02CE3AE5-F6D5-9177-913F-C34730F1096C">
    <semantic_event
        transaction_id="7775F27F-7AE4-0F70-8E62-76249B32DB06" name="RESULT" />
    <event_descriptor>
        <selection>MS_component</selection>
        <action>MultipleSelectionButtonPressed</action>
        <input>Checked for choice 1: Not checked for choice 2:
           Checked for choice 3: Not checked for choice 4:
           Checked for choice 5</input>
    </event_descriptor>
    <action_evaluation>INCORRECT</action_evaluation>
    <tutor_advice>One or more answers are wrong.</tutor_advice>
    <skill>
         <name>multiple_selection</name>
        <category>multiple_selection</category>
    </skill>
    <skill>
        <name>audio_question</name>
        <category>multiple_selection</category>
    </skill>
</tutor_message>
```

B. Frequently Asked Questions (FAQ)

B.1.1. Do I need a <meta> element in every message?

Yes, unless you are logging directly to an OLI log database using the OLI logging service—in this case, you never need to supply a meta element. The DataShop uses the context_message_id, the user_id, and the session_id to group the tool-tutor message pairs together with their context message.

B.1.2. If an element is required, can I just include it but leave it empty?

No, required elements should not be left blank/empty. They are required for a reason.

B.1.3. How should I represent an untutored action?

An untutored action can be represented by a single tool message with no tutor message following it.

B.1.4. In this document, you say that the <interpretation> element should not be used, but there is no guidance as to what to replace it with.

There is no replacement for the interpretation element. For Andes data, the DataShop team can run a program over the logs which will convert the multiple steps in the interpretation elements into multiple transactions [https://pslcdatashop.web.cmu.edu/help?page=terms#transaction].

C. History of changes to the Tutor Message format

Version 4 additions

- All selection, action, and input elements now have `id` and `type` attributes (2007/7/30).

- Added the following elements to the context message's `<dataset>` element (2007/6/28):

  - `<conversion_date>`

  - `<converter_info>`

  - `<replay>`

- Added the `anonFlag` (Boolean) attribute to the `<meta>` element's `<user_id>` element, which can indicate whether the user ID needs to be anonymized by the DataShop. (It's default value is `false`, meaning that the data have not yet been anonymized.) (2007/1/30)

- Added two new possible values for the `<problem>` element's `tutorFlag` attribute: `pre-test` and `post-test` (2007/1/30).

- Added new `buggy` attribute to the `<skill>` element, indicating whether or not the skill is a buggy skill. It's default value is `false`.

Major differences between version 2 and version 4
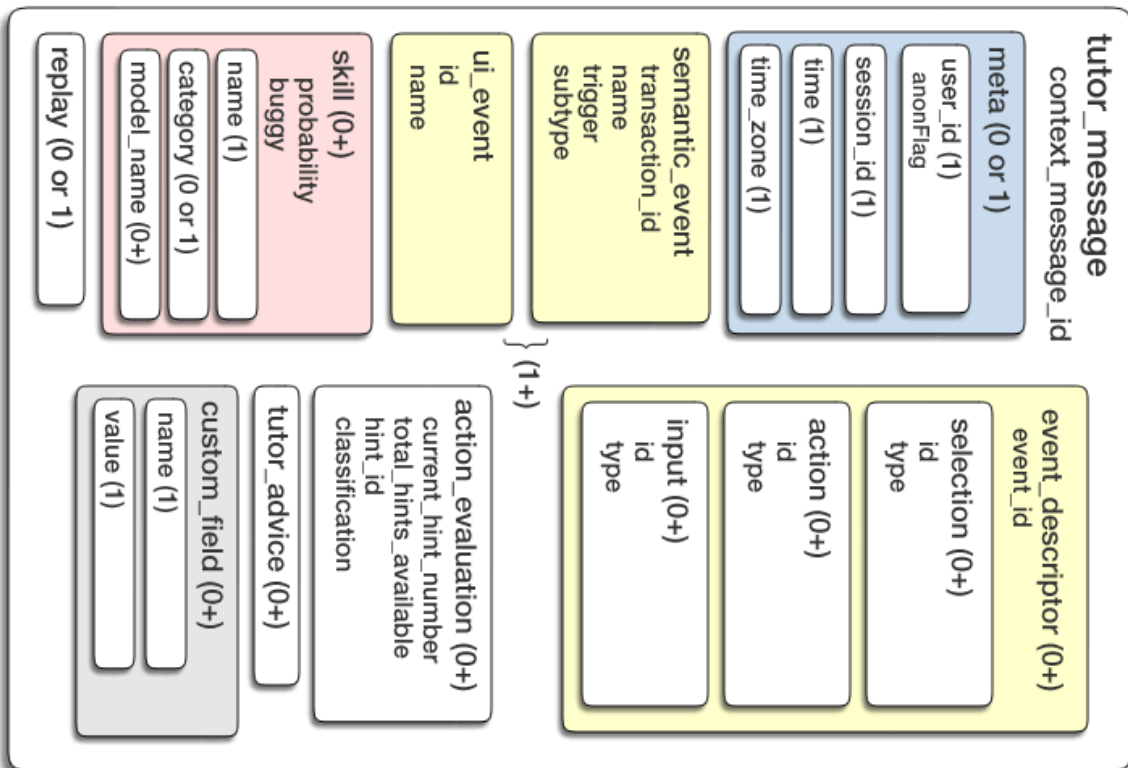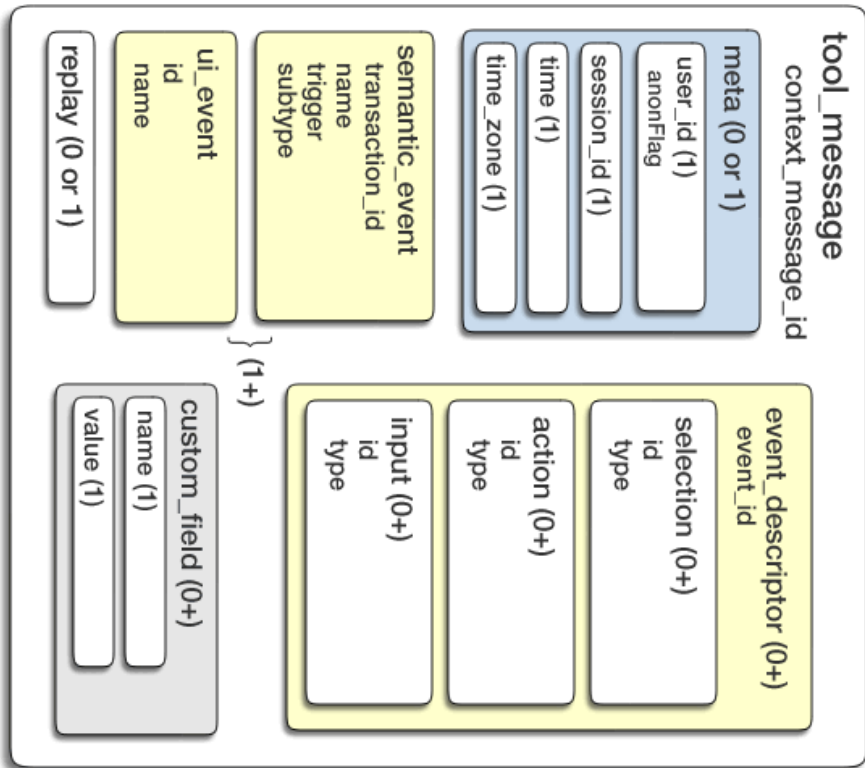
- `attempt_id` renamed to `context_message_id`

- New message type: `context_message`

  - Replaces `curriculum_message`

  - Flexible dataset hierarchy

  - `<condition>` element

    - Provides information on a study condition

- `tutor_message`

  - `<interpretation>` element added to handle the "cognitive tutor's idea" of what the student was doing

  - `<custom_field>` element added to store variables that are specific to the tutor application.

- `skill` has new sub-elements:

  - `name`

  - `category`

  - `model_name`

- `semantic_event` changes:

  - Removed `id` and `semantic_event_id`

  - Replaced with `transaction_id`

  - new attribute `subtype` for when additional classification of a semantic event is required

    - The Andes usage example is to have a semantic_event with the name `HINT_MSG` and a subtype of `DIALOG`.

- `action_evaluation` has two new attributes:

  - `hint_id` to identify hints that cannot be identified by their text.

  - `classification` for classifying errors or hints. For example, hint of type `BOTTOM_OUT`.

Release history

- Version 4: March 2006
- Version 3: June 2005 (not supported)
- Version 2: February 2005

D. Content model figures

tool_message
context_message_id

meta (0 or 1)
- user_id (1)
  anonFlag
- session_id (1)
- time (1)
- time_zone (1)

semantic_event
transaction_id
name
trigger
subtype

ui_event
id
name

replay (0 or 1)

} (1+)

event_descriptor (0+)
event_id

selection (0+)
id
type

action (0+)
id
type

input (0+)
id
type

custom_field (0+)
name (1)
value (1)

tutor_message
context_message_id

meta (0 or 1)
- user_id (1)
  anonFlag
- session_id (1)
- time (1)
- time_zone (1)

semantic_event
transaction_id
name
trigger
subtype

ui_event
id
name

skill (0+)
probability
buggy
name (1)
category (0 or 1)
model_name (0+)

replay (0 or 1)

} (1+)

event_descriptor (0+)
event_id

selection (0+)
id
type

action (0+)
id
type

input (0+)
id
type

action_evaluation (0+)
current_hint_number
total_hints_available
hint_id
classification

tutor_advice (0+)

custom_field (0+)
name (1)
value (1)

context_message
context_message_id
name

meta (0 or 1)
user_id (1)
anonFlag
session_id (1)
time (1)
time_zone (1)

class (0 or 1)
name (0 or 1)
school (0 or 1)
period (0 or 1)
description (0 or 1)
instructor (0 or 1)

dfa (0+)

skill (0+)
probability
buggy
name (1)
category (0 or 1)
model_name (0+)

dataset (1)
name (1)
converter_info (0 or 1)
conversion_date (0 or 1)

level (1)
type
name (1)
level (0 or 1)

problem (0 or 1)
tutorFlag
other
name (1)
context (0 or 1)

condition (0+)
name (1)
type (0 or 1)
desc (0 or 1)

custom_field (0+)
name (1)
value (1)

message
context_message_id
name

property (0+)

entry (0 or 1)