Proposal for Web Services: Custom Fields

Current Revision: January 5, 2011
First Revision: August 4, 2010

Introduction and Use Cases

In our API review in March, we suggested that we should enable users to annotate at both the **transaction** and **step** levels. Interviews with some users suggested that the **step** level was the only useful level for them to annotate. Other researchers suggested enabling annotation at the **student** level, and recent interviews with the SimStudent team revealed that many of their columns, which they now represent as transaction-level custom fields, could be modeled at other levels (see Table 1). We believe that supporting custom fields that can reference any of a variety of levels should be a requirement for web services (and later, the DataShop web application).

Table 1 - Sample of Current SimStudent Transaction Custom Fields

Custom Field Name	Description	Proposed Level
CF(ACTION)	a description of what was done – e.g., SimStudent asked for confirmation, Student gave input	
CF(ACTION_TYPE)	a designation of category that the CF(ACTION) falls into	
CF(DETAILS)	General free text type info	
CF(EXPECTED_INPUT)	True SAI's Input that should have been used	
CF(SAI_AGENT)	Who is the one doing the SAI, student or SimStudent	
CF(STEP)	the step name in terms of the state of the algebra instead of the component used, e.g. x-2=3[add 2] would be a step when the state of the equation is at x-2=3 and the student has declared they will add 2 next.	Student-Step
CF(USERID)	researchers' own anonymized ID that corresponds with the ID they put on their pre/post tests	Student
CF(DURATION)	duration to complete the noted activity, does not necessarily correspond with time since last log message	Student-Step-Problem View
Abstract Problem Type	abstract the type of problems used. for example, if the problems were of type 3x+4=5 or of type 3x-8=9 they would fall into the category of Ax+B= C and similar for other types of problems.	Problem
Completed / Incomplete	This was to figure out if the students completed a problem fully or not. Also to calculate how many problems were actually completed vs. the number or problems entered.	Student-Problem-Problem View

Additional use cases:

- One researcher wants student-level measures. He also would like to correlate measures (e.g., exam questions?) by time (transaction) and KC.
- Will enable researchers to use output from existing models
    - Gaming Detector [Arnon Hershkovitz, Mihaela Cocea, Summer 2008]
    - Bayesian Knowledge Tracing. [Hao Cen, migrated from feature wish list]
- Needed for M&M and CMDM clusters [Ryan Baker, October 2008]
- We want to enable someone to attach the results of an LFA-like model to a dataset [Ken]

Current Proposal

Under this proposal, a custom field would have a **name**, **description**, **data type**, **shared flag**, and a **level**—one or more of **transaction**, **student, step, problem, problem view**, **KC model**, and **KC** (see Table 2). Then a custom field could reference any of those levels by ID. Some of these are a new GUID (guaranteed unique ID) and some are referred to by name. We don't want to force the user to use GUIDs for everything and we do not necessarily want to expose more database IDs to the user. We already expose database IDs for the dataset, sample, and KC models in web services now.

Table 2

Granularity	Key	Number of columns in key
transaction	transaction_ID	1
student	student_ID	1
problem	problem_hierarchy, problem_name	2
step	step_ID	1
student, problem	student_ID, problem_hierarchy, problem_name,	3
student, problem, problem view	student_ID, problem_hierarchy, problem_name, problem_view	4
student, step	student_ID, step_ID	2
student, step, problem view	student_ID, step_ID, problem_view	3
kc_model	kc_model	1
kc	kc_model, kc	2
student, kc model	student_ID, kc_model	2
student, kc	student_ID, kc_model, kc	3
student, step, problem view, kc	student_ID, step_ID, problem_view, kc_model, kc	5

For example, a custom field could assign a value to a student but within the context of a KC model (e.g., a student intercept). The custom field's level would be "KC model, student".

To reference values, the user would use a key composed of column values instead of a GUID for some types of annotation. This would entail the following changes:

- Add a "Transaction ID" column to the transaction export
- Add a "Problem View" column to the transaction export
- Add a "Problem Hierarchy" column to the transaction export
- Add a "Step ID" column to transaction, student-step, and problem breakdown exports

For the following values, the user would use existing fields:

- Anonymous Student ID
- Problem Name
- KC Model Name
- KC Name

One would then use the sets of columns to reference values of those types, as shown in Table 2.

Support for Custom Fields in the Web Application

There are a variety of ways we could support custom fields in the web application:

- Enable export of custom fields in existing export formats (denormalized)
- Enable export of custom fields in new normalized views
- Allow users to select custom fields
- Allow users to upload/modify/delete custom fields
- Enable sampling by custom field (custom field as a sample filter type)
    - Would need to consider that:
        - some columns may not appear in the current export being viewed
        - a custom field is composed of many columns possibly, so which of the columns are you searching?
        - a sample on kc model or kc does not make sense as the kc model is a separate selection from sample.
- Enable graphing of custom field values

Denormalized Views

The locations where a denormalized view would be available are shown below:

Table 3 – Where we'd show (export) custom fields of the various level types in the web application and web services

	DataShop Export Types					
CF Level	transaction	student-step	student-problem	kc model	*student**	*dataset service (kc model)*
transaction	✓					
student	✓	✓	✓		✓	
problem	✓	✓	✓			
step	✓	✓				
student, problem	✓	✓	✓			
student, problem, problem view	✓	✓	✓			
student, step	✓	✓				
student, step, problem view	✓	✓				
kc model				✓		✓
kc	✓	✓	✓	✓		
student, kc model						
student, kc						
student, step, problem view, kc	✓	✓	✓			

* The student export is a possible export we could implement in the future.

Normalized View

We are considering making available a normalized view of custom fields. The current design proposal is to make this available from:

1. a page in the web application where the user can both select custom fields for inclusion in reports or export a normalized view; and
2. a web services call to get custom-field data

Table 4 - Normalized table example

KC Model	KC	Anon Student Id	Step Id	Problem View	CF (lfa-predicted-error-rate)
KTracedSkills	Find square…	Stu_4a7d1…	975d454f…	1	0.232
KTracedSkills	Find square…	Stu_4a7d1…	975d454f…	2	0.112
KTracedSkills	Find square…	Stu_4a7d1…	3259f905…	1	0.891
KTracedSkills	Find square…	Stu_4a7d1…	e42b0069…	1	0.233

Integrity and Usability Issues

When users can create custom fields that reference objects in the dataset such as KCs and KC models, and the DataShop web application allows deleting KC models, a potential usability issue can occur. Currently, we overlooked this issue in the design and implementation of KC "sets": if a user deletes a KC model that has KCs included in a KC set, the system does not prompt the user about this issue, it just allows the deletion without warning.

We should design custom fields (and revise the KC sets feature) so that deleting a KC model prompts the user if a custom field exists that references any KCs in that model. But should deletion be allowed? Similarly but from the opposite end, should we prevent a user from creating a custom field if one or more rows reference an object that doesn't exist?

Access and Security

We think column-based (custom-field-based) access control is too fine-grained. For now, let's consider that anyone with "view" permission on a dataset can add custom fields, and delete their own. In addition, a user should be allowed to specify whether a custom field is shared or not (as a sample in DataShop can be shared or not) with others who have the ability to view the dataset.

Modifying "logged" custom fields: We think that logged custom fields should be immutable.

Other ideas proposed:

- Anyone can add vs. Only those with edit can add (a project setting)
- Later have UI for access in general, including setting what a PI can do.

Web Services XML Format

The following example demonstrates a possible implementation of custom fields that supports the upper end of complexity. The thing being described is the results of an LFA-like model.

1. Create the custom field for predicted error rate, student values, KC values:

```
<?xml version="1.0" encoding="UTF-8"?>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-predicted-error-rate</name>
            <description>Output of LFA model: predicted error. This model has an AIC
            of 22, BIC of 21.41, a Log Likelihood of -0, and 11 total
            parameters.</description>
            <type>number</type>
            <level>student, step, problem_view, kc, kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-student-values</name>
            <description>Output of LFA model: student intercept
            values.</description>
            <type>number</type>
            <level>student, kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-kc-intercept-logit</name>
            <description>Output of LFA model: kc intercept (logit)
            values.</description>
            <type>number</type>
            <level>kc, kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-kc-intercept-probability</name>
            <description>Output of LFA model: kc intercept (probability) values.
            </description>
            <type>number</type>
            <level>kc, kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-kc-slope</name>
            <description>Output of LFA model: kc slope values.</description>
            <type>number</type>
            <level>kc, kcm</level>
      </custom_field>
</pslc_datashop_message>
```

2. Set the predicted-error-rate data:

```
<custom_field student_ID="Stu_b7e5d96b529ef095fa8534700ec5a536"
step_id="69c84447b0e89a3cf67c39e280b63ba9" problem_view="1" kc="enter-least-
common-denominator" kcm="Default">
       <value>0.232</value>
</custom_field>
etc.
```

3. Set the student intercept values:

```
<custom_field student_id="11" kcm="Default" id="46">
       <value>0.232</value>
</custom_field>
<custom_field student_id="12" kcm="Default" id="46">
       <value>0.441</value>
</custom_field>
<custom_field student_id="13" kcm="Default" id="46">
       <value>0.123</value>
</custom_field>
```

4. Set the KC intercept (logit) values and KC intercept (probability) values:

```
<custom_field kc="kc-a" kcm="Default" id="47">
       <value>-64.1</value>
</custom_field>
<custom_field kc="kc-b" kcm="Default" id="47">
       <value>-58.97</value>
</custom_field>

<custom_field kc="kc-a" kcm="Default" id="48">
       <value>0</value>
</custom_field>
<custom_field kc="kc-b" kcm="Default" id="48">
       <value>1</value>
</custom_field>
```

5. Set the KC slope values:

```
<custom_field kc="kc-a" kcm="Default" id="49">
       <value>0</value>
</custom_field>
<custom_field kc="kc-b" kcm="Default" id="49">
       <value>39.15</value>
</custom_field>
```

Previous Discussions

Add GUIDs

Under the older proposal, a custom field would have a **name**, **description**, **data type**, and a **level**—one or more of **transaction**, **step**, **student**, **KC model**, **KC**, and **problem**. Then a custom field could reference any of those levels by GUID (a guaranteed unique ID, not a database ID).

For example, a custom field could assign a value to a student but within the context of a KC model (e.g., a student intercept). The custom field's level would be "KC model, student".

To support this type of system, DataShop would need to provide IDs for objects of these various levels to users (through exporting, most likely). With KC model importing, we introduced a "step ID", a 33-character string that uniquely identifies the step. We would need to do something similar for transaction, student, KC model, KC, and problem.

Alternative Proposal, Sep 22, 2010

Initially, we thought we would need to add GUIDs for the following columns so that users could identify the following values uniquely:

- Transaction
- Step
- Problem
- Student
- KC
- KC Model
- Student-Step
- Student-Problem

We found that this could potentially double the size of a transaction export file and would be very unwieldy for users, especially those that prefer to use Excel.

On Sep 22, we proposed a simpler approach: use a key composed of column values instead of a GUID for some types of annotation. The proposal is as follows:

- Add a transaction ID to the transaction export
- We don't need a new anonymous student ID GUID as anonymous ID is already unique
- Add a "Problem View" column to the transaction export
- Add a "Problem Hierarchy" column to the transaction export
- Use the following sets of columns to reference values of those types:

We also discussed valid combinations of levels for custom fields. We listed the following (number of columns in parentheses):

- Transaction (1)

- Problem (2)
- Step (3)
- Student (1)
- Student-Step-Problem View (5)
- Student-Problem-Problem View (4)
- KC model (1)
- KC (2)
- Student-KCM (2)
- Student-KC (3)
- Student-Step-Problem View-KC (7)

For other combinations, do we prevent the user from creating such a custom field?

We wondered about the idea of a "primary" level, but agreed it wasn't needed. If a custom field references various levels, the value for the custom field appears in the column for that row *only if* all other referenced values exist.

More Discussion, Oct 1, 2010

On Oct 1, we suggested that we continue to use a step ID as opposed to representing step ID with 3 columns (problem hierarchy, problem name, and step name). That would simplify things a bit, and we already export a step GUID with the KC Model export.

Alternatives

The main alternatives to this proposal are to allow custom fields that describe steps and transactions only, or only transactions. The main benefits are that the custom fields can be exported more simply; custom fields have fewer unique IDs to reference; and the user has fewer unique IDs present in web-service exports. The main argument against this approach is that we're missing an opportunity to consider a very flexible system that would let users annotate almost any aspect of the data precisely. There is already some evidence that they would want to do this.

Discovering CFs for a Project

How would a user get a list of all the custom fields for all the datasets for a given project? Currently, we have proposed a way to get the custom fields for a dataset /datasets/[id]/customfields/[?id]. For a project, it could be similar /projects/[id]/customfields/[?id].

Also, should the user be able to add custom fields to a project? Would a project-level CF a single entity saved amongst datasets *or* just an attribute of a CF?

- Single entity: Define it once for the project and then add data using that CFs ID.
- Attribute: Define the custom field each time you use it. It becomes "shared" within a project by virtue of it having the same name, type, level, project as another CF.

Notes from August 6 meeting

- Uniqueness criteria for a custom field: a custom field's name only needs to be unique within a dataset
- We should allow a custom field that is shared amongst datasets within a project, which promotes standardization. How does a user define that a field is shared within a project? (See *Discovering CFs*, below.)
- Can a student custom field be shared across datasets? What does that mean again? Something about referring to the same student from different datasets. That's a student addressability issue, not a CF issue; ie, what's a student ID?

Predicted Error Rate and "Opportunity"

Predicted error rate is, in the case of LFA, supplied in terms of student, KC, step, and opportunity. But we wondered whether opportunity is a valid column for a custom field. It's just a column derived from other columns, so one could specify opportunity for a predicted error rate by giving other columns. We wondered if this was confusing since a transaction export of that custom field wouldn't have opportunity. But one could export by step and include the custom field to see opportunity.

Appendix A

[OLDER PROPOSAL] The following example demonstrates a possible implementation of custom fields that supports the upper end of complexity. The thing being described is the results of an LFA-like model.

1. Create the custom field for predicted error rate, student values, KC values:

```
<?xml version="1.0" encoding="UTF-8"?>
<pslc_datashop_message>
      <custom_field>
            <name>lfa-predicted-error-rate</name>
            <description>Output of LFA model: predicted error. This model has an AIC
            of 22, BIC of 21.41, a Log Likelihood of -0, and 11 total
            parameters.</description>
            <type>number</type>
            <level>student-step,kc,kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-student-values</name>
            <description>Output of LFA model: student intercept
            values.</description>
            <type>number</type>
            <level>student,kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-kc-intercept-logit</name>
            <description>Output of LFA model: kc intercept (logit)
            values.</description>
            <type>number</type>
            <level>kc,kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-kc-intercept-probability</name>
            <description>Output of LFA model: kc intercept (probability) values.
            </description>
            <type>number</type>
            <level>kc,kcm</level>
      </custom_field>
</pslc_datashop_message>

<pslc_datashop_message>
      <custom_field>
            <name>lfa-kc-slope</name>
            <description>Output of LFA model: kc slope values.</description>
            <type>number</type>
            <level>kc,kcm</level>
      </custom_field>
</pslc_datashop_message>
```

2. Set the predicted-error-rate data:

```
<custom_field student_step_id="123" kcm_id="321" id="45">
       <value>0.232</value>
</custom_field>
<custom_field student_step_id="124" kcm_id="321" id="45">
       <value>0.112</value>
</custom_field>
<custom_field student_step_id="125" kcm_id="321" id="45">
       <value>0.891</value>
</custom_field>
<custom_field student_step_id="126" kcm_id="321" id="45">
       <value>0.233</value>
</custom_field>
etc.
```

3. Set the student intercept values:

```
<custom_field student_id="11" kcm_id="321" id="46">
       <value>0.232</value>
</custom_field>
<custom_field student_id="12" kcm_id="321" id="46">
       <value>0.441</value>
</custom_field>
<custom_field student_id="13" kcm_id="321" id="46">
       <value>0.123</value>
</custom_field>
```

4. Set the KC intercept (logit) values and KC intercept (probability) values:

```
<custom_field kc_id="11" kcm_id="321" id="47">
       <value>-64.1</value>
</custom_field>
<custom_field kc_id="12" kcm_id="321" id="47">
       <value>-58.97</value>
</custom_field>
<custom_field kc_id="13" kcm_id="321" id="47">
       <value>46.22</value>
</custom_field>

<custom_field kc_id="11" kcm_id="321" id="48">
       <value>0</value>
</custom_field>
<custom_field kc_id="12" kcm_id="321" id="48">
       <value>1</value>
</custom_field>
<custom_field kc_id="13" kcm_id="321" id="48">
       <value>0</value>
</custom_field>
```

5. Set the KC slope values:

```
<custom_field kc_id="11" kcm_id="321" id="49">
```

```
        <value>0</value>
</custom_field>
<custom_field kc_id="12" kcm_id="321" id="49">
        <value>39.15</value>
</custom_field>
<custom_field kc_id="13" kcm_id="321" id="49">
        <value>46.22</value>
</custom_field>
```

Output the entire thing in a tabular format:

lfa-predicted-error-rate (id=45)

Output of LFA model: predicted error. This model has an AIC of 22, BIC of 21.41, a Log Likelihood of -0, and 11 total parameters.

kcm_id	kc_id	step_id	problem_view	lfa-predicted-error-rate
321	11	123	1	0.232
321	12	124	1	0.112
321	13	125	1	0.891
321	14	126	1	0.233

[OLDER IDEA]

Output of LFA model: predicted error. This model has an AIC of 22, BIC of 21.41, a Log Likelihood of -0, and 11 total parameters.

kcm_id	kc_id	student_step_id	lfa-predicted-error-rate
321	11	123	0.232
321	12	124	0.112
321	13	125	0.891
321	14	126	0.233

[FOLLOWING IS UNCHANGED]

lfa-student-values (id=46)

Output of LFA model: student intercept values.

kcm_id	student_id	lfa-student-values
321	11	0.232
321	12	0.441
321	13	0.123

lfa-kc-intercept-logit (id=47)

Output of LFA model: kc intercept (logit) values.

kcm_id	kc_id	lfa-kc-intercept-logit
321	11	-64.1

321	12	-58.97
321	13	46.22

lfa-kc-intercept-probability (id=48)

Output of LFA model: kc intercept (proability) values.

kcm_id	kc_id	lfa-kc-intercept-probability
321	11	0
321	12	1
321	13	0

lfa-kc-slope (id=49)

Output of LFA model: kc slope values.

kcm_id	kc_id	lfa-kc-slope