ACCEPT

SEVENTH FRAMEWORK PROGRAMME

THEME ICT-2011.4.2(a)

Language Technologies

ACCEPT
Automated Community Content Editing PorTal

www.accept.unige.ch

Starting date of the project: 1 January 2012

Overall duration of the project: 36 months

Adapted Evaluation Portal Prototype to Allow for the Collection of User Ratings

Workpackage n° 5	Name: Portal Integration
Deliverable n° 5.3	Name: Adapted evaluation portal prototype to allow for the collection of user ratings.
Due date: 30 September 2012	Submission date: 1 October 2012

Dissemination level: PU

Organisation name of lead contractor for this deliverable: SYMANTEC

The research leading to these results has received funding from the European Community's Seventh Framework Programme (*FP7/2007-2013*) under *grant agreement n° 288769.*

SEVENTH FRAMEWORK PROGRAMME

Contents

Objectives of the Deliverable

The main objective of this deliverable, which is part of Task 5.3, was to adapt an existing evaluation portal where user ratings can be collected to assess the quality of source, machine-translated and post-edited content. Some of the functionality of this portal was going to be based on Symantec's existing evaluation portal (www.eval4all.com), in order to ultimately support:

- The evaluation criteria defined in WP9

- The scoring of machine-translated and post-edited segments against reference translations using automatic metrics (so that it can be made available for analysis in WP9)

- The export of data using XLIFF (so that it can be made available for analysis in WP9)

- The rewarding of usage (to help community management in WP6).

This deliverable comprises two parts: the Evaluation Project Management Section and the ACCEPT Evaluation API. Both are hosted on the ACCEPT portal on an Amazon EC2 instance (www.accept-portal.eu).

Glossary

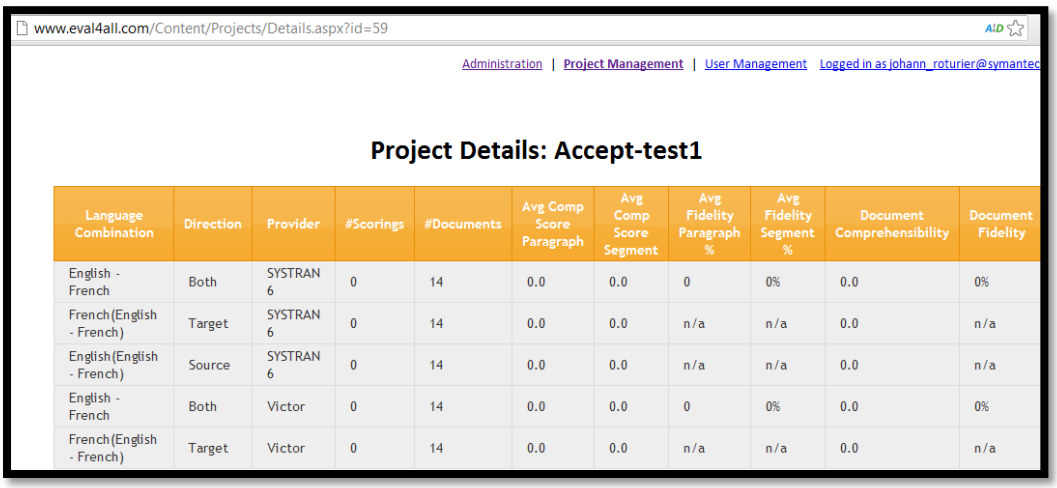
The following terms are used frequently in this deliverable so we provide the following definitions for reference purposes:

Question Category	This is a container for one or multiple languages, where one or multiple questions can be added. An example of a Question Category would be "Fluency", where all questions would be related to the fluency of text.
Question	This is a question that should be answered by users. Questions are language-specific. There can be multiple questions associated with a category.
Question Answer	This is an answer to a question. There can be multiple answers associated with a question.

General Approach

The approach used for this deliverable has been based on offering as much flexibility as possible to evaluation project creators. To this end, the focus has been placed on providing a framework for creating evaluation projects, including projects whose data sets reside outside of the ACCEPT Portal. A flexible approach of this kind was not provided by Symantec's existing evaluation portal, for two main reasons:

- Project data had to be imported into the eval4all.com system. This meant that it was not possible to collect ratings for data residing outside the system. Since one of the objectives of the ACCEPT project is to collect genuine user feedback in-context, this was a severe limitation.
- Evaluation criteria were hardcoded at the system-level. This meant that it was not possible to adjust evaluation criteria on a per-project basis. Again, this was a severe limitation in the light of the work that is being conducted in WP9. This limitation is illustrated in Figure 1: Eval4all.com Project Details Page, where results for two hardcoded evaluation criteria ("Comprehensibility" as a 1-5 scale and "Fidelity" as a binary category) are presented.



Language Combination	Direction	Provider	#Scorings	#Documents	Avg Comp Score Paragraph	Avg Comp Score Segment	Avg Fidelity Paragraph %	Avg Fidelity Segment %	Document Comprehensibility	Document Fidelity
English - French	Both	SYSTRAN 6	0	14	0.0	0.0	0	0%	0.0	0%
French(English - French)	Target	SYSTRAN 6	0	14	0.0	0.0	n/a	n/a	0.0	n/a
English(English - French)	Source	SYSTRAN 6	0	14	0.0	0.0	n/a	n/a	0.0	n/a
English - French	Both	Victor	0	14	0.0	0.0	0	0%	0.0	0%
French(English - French)	Target	Victor	0	14	0.0	0.0	n/a	n/a	0.0	n/a

Figure 1: Eval4all.com Project Details Page

In order to address these limitations, we have designed a new evaluation framework from the ground up. As part of this deliverable, the new evaluation framework is being leveraged by the Evaluation Section of the ACCEPT Portal (via a project management page and a simple demo page) and by the ACCEPT Evaluation API. In a future release, we will use this framework to support eval4all.com-based features (such as random task assignment or data upload).

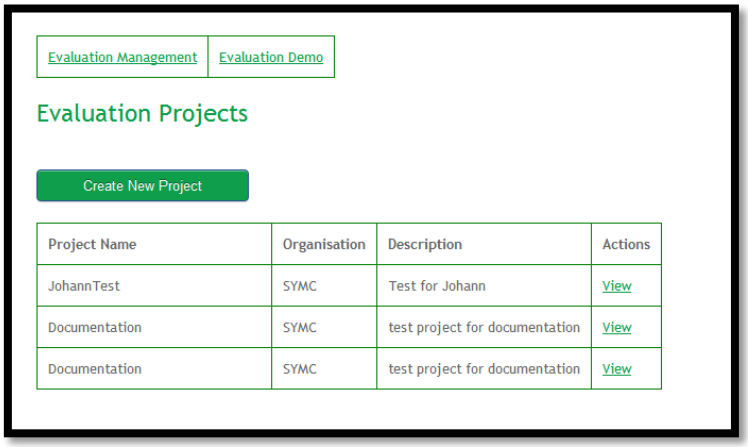
The ACCEPT Portal's Evaluation Section

Overview

The ACCEPT Portal described in Deliverable 5.1 has been extended to include an Evaluation section (under the "Evaluate" tab). This section may be used by any registered user to create and edit Evaluation projects, as well as monitor user ratings. However, projects are only visible by the project creator.

Evaluation Projects Page

On the Evaluation Projects page, a logged-in user can see all of the projects they have created. To create a new project, a user may click on "Create New Project". To view the details of a project, a user may click "View" on the right column of the project record.



Project Name	Organisation	Description	Actions
JohannTest	SYMC	Test for Johann	View
Documentation	SYMC	test project for documentation	View
Documentation	SYMC	test project for documentation	View

Figure 2: Main Project Page

Create Evaluation Project Page

To create a project, the Name, Description and Organisation fields must be filled in.

- Name: Project Name (Does not have to be unique)
- Description: Short Project Description
- Organisation: Name of the organisation that owns the project (e.g. “Symantec”).
- API Key Domain: Specify the domain name or IP address the API access will be restricted to (e.g. www.symantec.com). Multiple values separated with a semi-colon are possible.

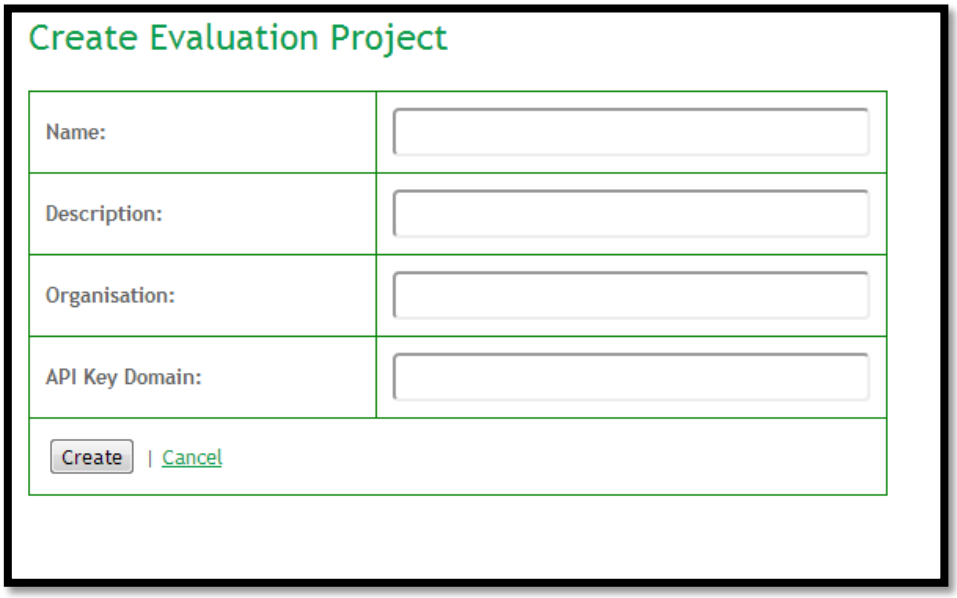


Figure 3: Project Creation Page

Once “Create” is clicked, the user is redirected to the Evaluations Project Page, where the newly-created project will be listed. When a project is created, an API Key is automatically generated and associated with the project. If details must subsequently be changed (e.g. project name or description), it is possible to change them by clicking on “Edit”.

Edit Evaluation Project Page

The project page shows information about the current project, including the project’s ID. Some of this information can be edited by clicking the “Edit” button.

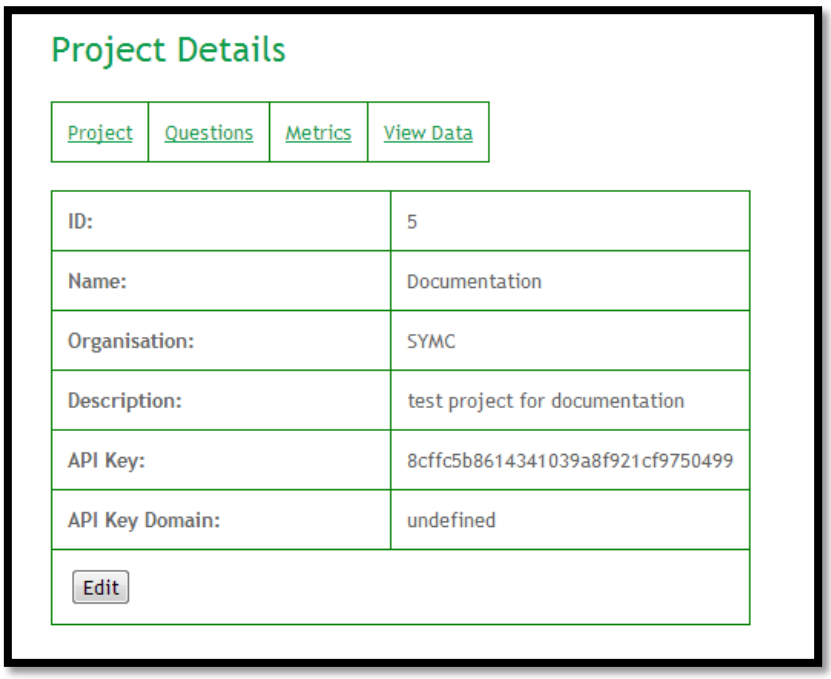


Figure 4: Project Management Page

Evaluation Project Questions

Once a project has been created, the first step is to define a question category, which is used as a container for one or more questions (as defined in the Glossary). Some of these question categories will be defined in WP9.

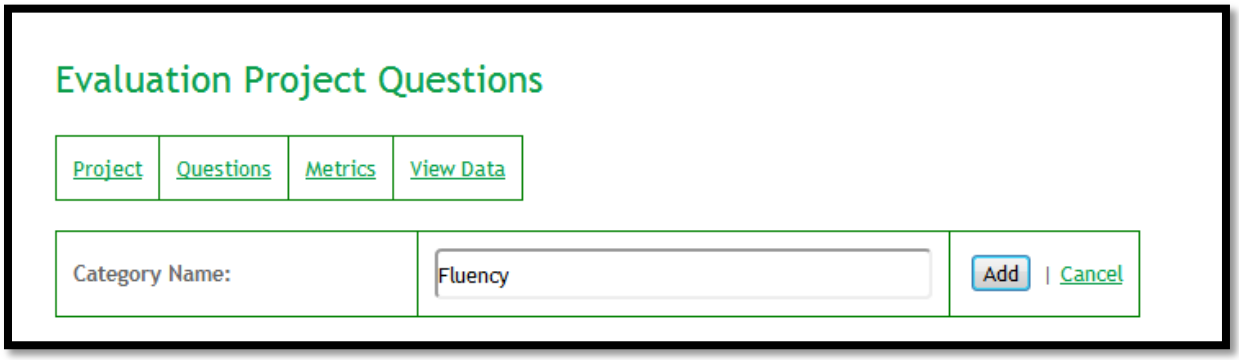


Figure 5: Creating a Question Category

Once a category has been defined, a first question can be added, as shown in Figure 6: Adding a Question to a Question Category. In this example, a “Fluency” question category has been defined as a container for one or more questions. Figure 6: Adding a Question to a Question Category shows that an English question is being added to this container. A question has multiple attributes besides the question text itself (which happens to be “How fluent is this translated content” in this example):

- A language (e.g. English if the Question text is in English)
- An Action text (which may be used to instruct users how to submit an answer)
- An Action Confirmation text (which may be used to show users that their answer has been submitted).

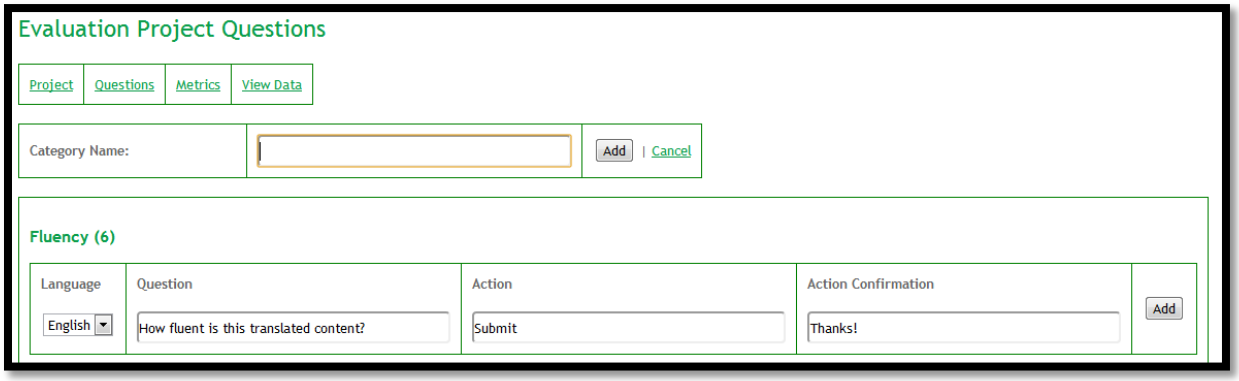


Figure 6: Adding a Question to a Question Category

Once the question attributes have been added, the question's answers (both text and value) may be added, as shown below:

Figure 7: Adding Answers

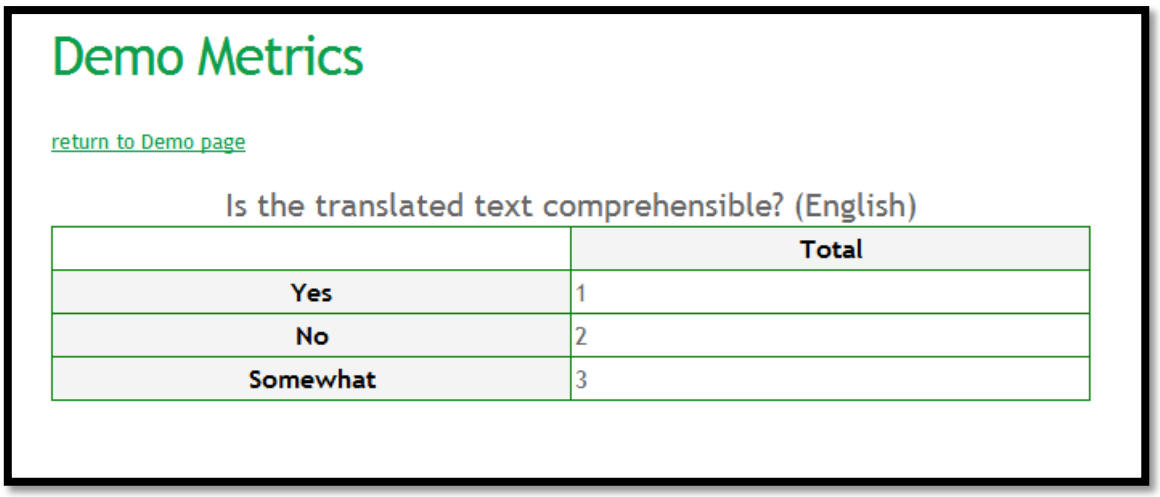
In the example above, each answer has two parts: the actual answer text to present to the user and a value. The value field, which is optional, allows project creators to give information on how these answers may be presented to the user. For example, in the example above a ranking scheme from "incomprehensible" to "flawless" has been used. In order to make sure that answers are presented in a natural order on the client-side, the values may be used to sort and order the answers (from 0 to 4).

Once at least one question and one answer have been defined for a given project, answers may be collected using the ACCEPT Evaluation API, which is described in the following section: The ACCEPT Evaluation API.

Note: Answers that require free-form text from the user may be submitted using additional parameters. More information is provided in the following section: Score Method.

Evaluation Project Metrics

Answers that are being submitted on the client-side are recorded in the ACCEPT Evaluation database. A summary of these answers is provided on the Metrics Page:



Demo Metrics

return to Demo page

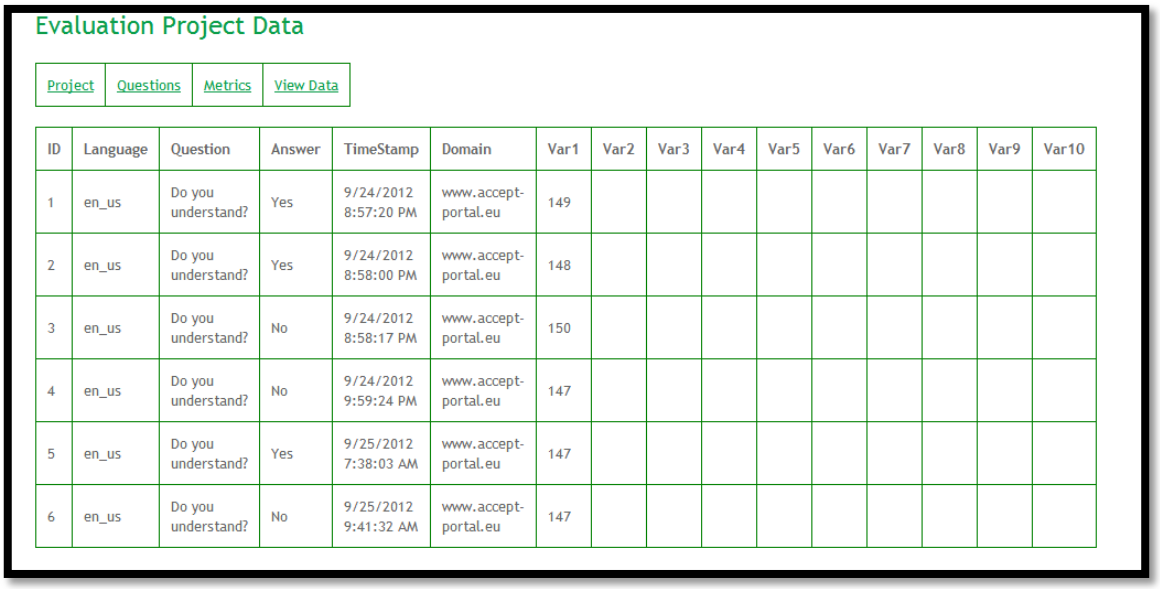
Is the translated text comprehensible? (English)

	Total
Yes	1
No	2
Somewhat	3

Figure 8: Evaluation Metrics Page

Evaluation Project Data

The full set of evaluation project data may be found on the View Data page:



Evaluation Project Data

Project | Questions | Metrics | View Data

ID	Language	Question	Answer	TimeStamp	Domain	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10
1	en_us	Do you understand?	Yes	9/24/2012 8:57:20 PM	www.accept-portal.eu	149									
2	en_us	Do you understand?	Yes	9/24/2012 8:58:00 PM	www.accept-portal.eu	148									
3	en_us	Do you understand?	No	9/24/2012 8:58:17 PM	www.accept-portal.eu	150									
4	en_us	Do you understand?	No	9/24/2012 9:59:24 PM	www.accept-portal.eu	147									
5	en_us	Do you understand?	Yes	9/25/2012 7:38:03 AM	www.accept-portal.eu	147									
6	en_us	Do you understand?	No	9/25/2012 9:41:32 AM	www.accept-portal.eu	147									

Figure 9: Evaluation Data Page

The ACCEPT Evaluation API

Overview

The ACCEPT API described in Deliverable 5.1 has been extended to support the following evaluation-related methods:

- Questions
- Score

These methods have been created with a view to provide project administrators with as much flexibility as possible.

Methods Details

Questions Method

This is a GET method.

The **Questions** method will return a list of project-specific questions. These questions can be managed (added etc.) via the Accept Portal Evaluation Project page.

The URL for calling the **Questions** method is:

http://www.accept-portal.com/Api/v1/Evaluation/Questions/{ID}

where {ID} is the Evaluation project ID). For an Evaluation project with an ID of "1" the call would be:

http://www.accept-portal.com/Api/v1/Evaluation/Questions/1

The following parameters can be passed **on** the URL:

Parameter	Details	Example
key	This is a MANDATORY parameter. If this parameter is not passed then the API call will fail. This value can be seen on the Evaluation Project page.	key=21fdc25bebdf456db5c9e0993977bb85
language	This is an OPTIONAL parameter. If this parameter is passed then only the questions in the specified language will be returned. Language code is based on RFC 4646. Examples of language codes are: en, fr, en-us, fr-fr	language=en_us
category	This is an OPTIONAL parameter. If this parameter is passed then only the questions in the specified category will be returned.	category=1

question	This is an OPTIONAL parameter. If this parameter is passed then only this question will be returned.	question=1

If no parameters (apart from Key) are passed then ALL questions will be returned.

An example of an API call containing all parameters is:

http://www.accept-portal.com/Api/v1/Evaluation/Questions/1?key=21fdc25bebdf456db5c9e0993977bb85&language=en_us&category=1&question=1

A JSON structure will be returned (the example below has one question returned):


```
{
	"ResponseObject":[
		{
			"Name":"QualityQuestion",
			"LanguageQuestions":[
				{
					"Question":"What is your name?",
					"Action":"Submit",
					"Confirmation":"Thank You",
					"Answers":[
						{
							"Name":"Rob",
							"Count":7,
							"Id":1
						},
						{
							"Name":"John",
							"Count":5,
							"Id":2
						}
					],
					"Language":
					{
						"Name":"English",
						"Code":"en_us",
						"Id":1
					},
					"Count":12,
					"Id":1
				}
			],
			"Count":0,
			"Id":1
		}
	],
	"ResponseStatus":"OK",
	"TimeStamp":"\/Date(1348482649898)\/",
}
```


Return Values

ResponseStatus	Status of the API call. Possible options are: OK = Successful call. FAILED = API call failed.
ResponseObject	This is an array of question categories associated with the project. See below for more information on the structure. This will only be returned if ResponseStatus is OK.
TimeStamp	Datetime of API response.

Exception	If the ResponseStatus is equal to “FAILED” then the “Exception” item will also be returned. This will contain an explanation of why the call failed.
Context	If the ResponseStatus is equal to “FAILED” then the “Context” item will also be returned. This will give some context as to where the call failed.

ResponseObject

This will contain an array of question categories.

Id	This is the Id of the Question Category.
Name	The name of the category. This will only be returned if ResponseStatus is OK.
LanguageQuestions	This is an array of questions associated with the Category. See below for more information on the structure. This will only be returned if ResponseStatus is OK.
Count	This is the current count for the number of times a Score has been submitted for this Question Category. **Note:** The Count could be used to decide WHAT question to ask.

LanguageQuestions

This will contain an array of questions.

Id	This is the Id of the Question.
Question	This is the text for the question
Action	This is the text for an action that can take place related to this question. Examples could be: “Submit” or “Vote” or “Please click on this button to submit your vote!”
Confirmation	This is the text for a confirmation message that can be displayed once a user has clicked on an action. Examples could be: “Thank you for voting!”
Answers	This is an array of possible answers to the question. See below for more information on the structure.
Language	This is the language associated with the question. See below for more information on the structure.
Count	This is the current count for the number of times a Score has been submitted for this question.

Answers

This will contain an array of answers.

Id	This is the Id of the Answer
Name	This is the text for the answer
Count	This is the current count for the number of times a score has been submitted for this answer.

Language

Id	This is the Id of the Language.
Name	This is the name of the language.
Code	This is the short language code, e.g. “en”, “fr”, “en-us”

Score Method

This is a POST method.

The **Score** method will submit the selected answer and other optional data.

The URL for calling the **Score** method is:

http://www.accept-portal.com/Api/v1/Evaluation/Score/{ID}

where {ID} is the Evaluation project ID). For an Evaluation project with an ID of “1” the call would be:

http://www.accept-portal.com/Api/v1/Evaluation/Score/1

The following parameters can be passed as JSON in the Request body:

Parameter	Details	Example
key	This is a **MANDATORY** parameter. If this parameter is not passed then the API call will fail. This value can be seen on the Evaluation Project page.	key=21fdc25bebdf456db5c9e0993977bb85
answer	This is a **MANDATORY** parameter. If this parameter is not passed then the API call will fail. This value is the ID of the answer.	answer=1
param1	This is an **OPTIONAL** parameter. This value will accept content up to a length of 25,000 Unicode characters.	param1=This is the source text
param2	This is an **OPTIONAL** parameter. This value will accept content up to a length of 25,000 Unicode characters.	param2= This is the target text
param3	This is an **OPTIONAL** parameter. This value will accept content up to a length of 2,500 Unicode characters.	param3= This is a comment
param4	This is an **OPTIONAL** parameter. This value will accept content up to a length of 250 Unicode characters.	param4=This is a username
param5	This is an **OPTIONAL** parameter. This value will accept content up to a length of 250 Unicode characters.	param5=This is the browser language
param6	This is an **OPTIONAL** parameter. This value will accept content up to a length of 250 Unicode characters.	param6=This is the browser type

param7	This is an **OPTIONAL** parameter. This value will accept content up to a length of 25,000 Unicode characters.	param7=This is how long the user spent looking at the page before voting
param8	This is an **OPTIONAL** parameter. This value will accept content up to a length of 250 Unicode characters.	param8=This is the OS type
param9	This is an **OPTIONAL** parameter. This value will accept content up to a length of 250 Unicode characters.	param9=This is the user's screen resolution
param10	This is an **OPTIONAL** parameter. This value will accept content up to a length of 250 Unicode characters.	param10=This is the web application version number

Note: The optional parameter examples above are informative and are not meant to limit what can be posted.

For the examples above the Request body should look similar to:

```
{
	"key": "21fdc25bebdf456db5c9e0993977bb85",
	"answer": "2",
	"param1": "Hello world",
	"param2": "Bonjour le monde",
	"param3": "This is a very basic example",
	"param4": "Jacques",
	"param5": "fr-FR ",
	"param6": "Chrome",
	"param7": "1m 34sec ",
	"param8": "Win32 ",
	"param9":  "1440 × 900 pixels ",
	"param10": " My WebApp v1.0.1.2 "
}
```

The Content-Type must be set to "application/json".

A JSON structure will be returned (the example below has one question returned):

```
{
	"ResponseObject":
	{
		"ProjectID":1,
		"ScoreID":15,
		"Success":true
	},
	"ResponseStatus":"OK",
	"TimeStamp":"\/Date(1348494657358)\/",
}
```

Return Values

ResponseStatus	Status of the API call. Possible options are:

	OK = Successful call. FAILED = API call failed.
ResponseObject	This will contain information on the score submission. See below for more information on the structure. This will only be returned if ResponseStatus is OK.
TimeStamp	Datetime of API response.
Exception	If the ResponseStatus is equal to "FAILED" then the "Exception" item will also be returned. This will contain an explanation of why the call failed.
Context	If the ResponseStatus is equal to "FAILED" then the "Context" item will also be returned. This will give some context as to where the call failed.

ResponseObject

This will contain information on the score submission.

Success	This is a boolean that will indicate whether the submission was successful or not.
ScoreId	This is the Id of the Score. If Success is false then this value will be zero.
ProjectId	This is the Id of the Project. If Success is false then this value will be zero.

Client-side Example Used on the Demo Page

In order to demonstrate how the API may be used, we have created a Demo page on the ACCEPT Portal (http://www.accept-portal.eu/AcceptPortal/en/Evaluation/Demo). The code snippets below, which are those used on this Demo page, show how to call the Questions method to ask for feedback. The HTML shown in Figure 10: HTML Table containing two rows contains two example content records in an HTML table. The first field contains a tag with the name "voting". The second field contains some example content.

```
<!-- An example table containing two records -->
<table border="0">
        <tr>
        <td>
        <div class="lia-message-view message-uid-147" id="messageview">
                <table border=1>
                <tr>
                <td><div name="voting" /></td>
                <td>
                        <div class="content-source">This is a test sentence!</div>
                </td>
                <td>
                        <div class="content-mt"> C'est une test phrase!</div>
                </td>
                </tr>
                </table>
        </div>
```

```
        </td>
        </tr>
        <tr>
        <td>
        <div class="lia-message-view message-uid-148" id="messageview">
                <table border=1>
                <tr>
                <td><div name="voting" /></td>
                <td>
                        <div class="content-source">This is another test sentence!</div>
                </td>
                <td>
                        <div class="content-mt">C'est une autre test phrase!</div>
                </td>
                </tr>
                </table>
        </div>
        </td>
        </tr>
</table>
```

Figure 10: HTML Table containing two rows

We want to replace the " " with our Questions. This requires two things:

1. Get the list of questions by making an API call
2. Process the returned data to display the questions

To get the list of questions, the jQuery AJAX method may be used, and the returned data may be processed in the success function.

```
<script type="text/javascript" language="javascript">
  $(document).ready(function () {
    $.ajax({
      url: 'http://www.accept-
portal.com/Api/v1/Evaluation/Questions/1?key=<api_key>&language=en_us&category=1&question
=1',
      contentType: "application/json",
      type: "GET",
      async: false,
      cache: false,
      dataType: 'jsonp',
      success: function (data) {
        var text = '';
        var q = data.ResponseObject[0];
        for (var i = 0; i < q.LanguageQuestions.length; i++) {
          question = q.LanguageQuestions[i].Question;
          questionId = q.LanguageQuestions[i].Id;
          language = q.LanguageQuestions[i].Language.Code;
          languageid = q.LanguageQuestions[i].Language.Id
          action = q.LanguageQuestions[i].Action;

          $("div[class^='lia-message-view message-uid-']").each(function () {
```

```
                var id = $(this).attr("class");
                var PostID = $(this).attr("class").replace("lia-message-view message-uid-", "");
                var frmName = "voting_" + PostID;
                text = '<form id="' + frmName + '"><table border=1><tr><td bgcolor=lightgray><b>' +
question + '</b></td></tr>';
                text += '<tr><td><div id="voteoptions">';
                for (var x = 0; x < q.LanguageQuestions[i].Answers.length; x++) {
                   answer = q.LanguageQuestions[i].Answers[x];
                   if (x == 0)
                      text += '<input id="radioCount_' + answer.Id + '" type="radio" name="choice"
value="' + answer.Id + '" checked>' + answer.Name;
                   else
                      text += '<input id="radioCount_' + answer.Id + '" type="radio" name="choice"
value="' + answer.Id + '">' + answer.Name;
                }
                text += '<br/>';
                text += '<input  value="' + action + '" type="button" onclick="SubmitVote(\'' + frmName
+ '\');" />';
                text += '</div>';
                text += '<div id="voteresult" style="display:none">';
                text += '<font color="green">RESULT: ' + q.LanguageQuestions[i].Confirmation +
'</font>';
                text += '</div>';
                text += '</td></tr>';
                text += '</table></form>';
                $(this).find('div[name="voting"]').html(text);
             });
          }
       },
       error: function (jqXHR, textStatus, errorThrown) {
          alert(errorThrown);
       }
    });

  });

</script>
```

Figure 11: Javascript Example Using the Questions Method of the ACCEPT Evaluation API

The Submission javascript code is as follows:

```
<script type="text/javascript" language="javascript">

  function SubmitVote(frmName) {
     var PostID = frmName.replace("voting_", "")
     $('#' + frmName + " input:radio").each(function () {
        if ($(this).is(':checked')) {
           debugger;
           var answerID = $(this).val();
           $.ajax({
              url: 'http://www.accept-portal.com/Api/v1/Evaluation/Score/1',
              dataType: 'json',
```

```
            contentType: "application/json",
            type: 'post',
            data: '{ "key": "<api_key>", "answer": ' + answerID + ', "param1": ' + PostID + '  }',
            success: function (e) {
               $('#' + frmName).find('#voteoptions').css("display", "none");
               $('#' + frmName).find('#voteresult').css("display", "block");
            },
            error: function (jqXHR, textStatus, errorThrown) {
               //alert(errorThrown);
            }
         });
      }
   });
  }
</script>
```

Figure 12: Javascript Example Using the Score Method of the ACCEPT Evaluation API

API Security and Authentication

It is expected that the ACCEPT API will commonly be called on the client side on the user's browser. This introduced a complication in implementing an authentication model for the API. API keys are being used; however, if they are being referenced client side, then a user could simply view the source of the web page and easily see the key that is being used.

To counter this, the concept of approved domains was introduced. All API keys will have an approved domain associated with them. The intention here is that in addition to checking that the API key is valid, the system will also check to make sure that that key is valid for that domain. This domain verification is being performed by checking the Referrer URL within the HTTP request header for every API call. In practice the Referrer is the URL of the previous host which led to the current API request. This allows us to determine where an API call is coming from by grabbing the domain within the Referer URL and comparing it with the one associated with the API key provided.

To enable this validation check when the API is being used from a script, we also check the UserHostAddress if the HTTP Referrer is empty. A flaw in this logic is that the HTTP Referrer can be easily spoofed. This is not of concern at the moment as the main focus at this point is to collect data and not to block unauthorized users. Another issue is that the Referer value can also be hidden or misspelled by a server for privacy concerns. In this case the authentication process will fail. In this scenario the UserHostAddress will be used as a fall-back.

Future Work

The following features will be considered for the next version of the Evaluation Component:

- Sharing questions and categories between projects, possibly by importing questions and/or answers from existing projects, or by providing a table of the most popular categories/questions etc.
- Submitting multiple scores in one call (e.g. an array of scores)
- Updating ratings (e.g. if the user voting changes their mind). Currently each rating submission is an insert so there is no possibility of going back.