Remote Question Protocol

Serving Items in a Distributed Learning Environment

December 2, 2004

1 Nomenclature

Throughout this document the terminology of the IMS QTI version 2.0 specification [2] has been adopted where possible. In particular the QTI specification makes the following definitions.

Assessment Delivery System A system for the administration and delivery of assessments to candidates.

Delivery Engine The process that coordinates the rendering and delivery of the Item(s) and the evaluation of the responses to produce scores and Feedback.

Item Bank A system for collecting and managing collections of assessment items.

Authoring System A system used by authors for creating and editing Items.

The assessment delivery system might be a Virtual Learning Environment for example. The delivery engine might be a module within that VLE together with possible plugins for different question types.

An item bank can be a public repository or a local database and the QTI specification makes no distinction. Documents in QTI format, some extension of QTI such as MathsQTI or custom formats are stored in an item bank and describe items or item templates.

Note that some of the QTI terms can be confusing.

- Item clones are not identical to each other but instead are instances of an item template for different seeds.
- For some assessment delivery systems, rendering may mean the production of formatted display markup such as HTML.

2 Purpose

This specification aims to provide a web services protocol for delivery engines to request processing of items or item templates from remote services. This will allow existing delivery engines to easily implement support for item types which were previously unsupported. For example MathsQTI questions could be handled in this way. Furthermore it will allow for thin delivery engines to be written where all features are implemented by calling appropriate services. Such delivery engines would be very flexible and extensible.

While RQP is designed to support QTI items fully it is also designed for use with other formats. It requires that the item source start with an 'assessmentItem' tag with a 'guid' attribute but any format (not necessarily valid XML) is permitted to follow.

3 Architecture

A delivery engine may include the following components (not all of the terms are defined by QTI).

Assessment Engine Coordinates the rendering and delivery of items to form assessments and coordinates the scoring of the items in each assessment. As part of this role it manages the retrieval of items and item templates from the item bank(s).

Cloning Engine Produces an item clone from an item template and a random seed.

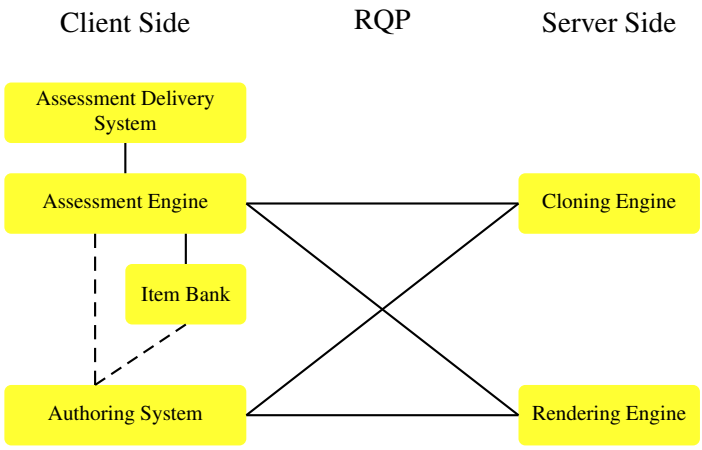
Scoring Engine Performs response processing and scoring of an item.

Rendering Engine Renders an item or produces display markup which may be rendered later. The latter will normally be the case.

The assessment engine is the core of the delivery engine and simply coordinates the calls to the other components and integrates with the assessment delivery system. It will be tightly coupled to the assessment delivery system and will almost certainly reside on the same machine.

The purpose of RQP is to provide a web services protocol which allows the cloning and rendering engines to be remote. They can be separate servers or offered as part of one server. The scoring engine is taken to be built in to the rendering engine; rendering always occurs after response processing and scoring.

The following diagram illustrates the architecture at its most general, with all services connected remotely from each other. This shows all the links which RQP allows for.



End users interact with the assessment delivery system or the authoring system (either directly or by some remote link such as a web interface).

The servers are allowed to be stateless and independent. In practice they will probably want to cache data for efficiency and the protocol provides a mechanism for this. It is also likely that the cloning and rendering engines for a particular question type will reside on the same server. In this case they are free to share the same cache and the protocol provides a mechanism for combining calls.

The protocol is designed to allow load balancing so the client may use different servers for each student attempt provided that the servers are of the same type.

4 Item Structure

For the purposes of RQP an item is defined as a unit for which a score is available. This may be a subpart of a grouped question but the arrangement and numbering of items is handled entirely by the assessment engine. An item may contain multiple interactions however the score applies to them overall and partial entry gives only partial credit. This definition matches that given by QTI.

Items may be written using any format but RQP is primarily designed for QTI with possible extensions such as MathQTI for mathematics support. We require that each item is given a globally unique identifier (GUID) upon each edit. The item source is required to start with an 'assessmentItem' tag with the 'guid' attribute set. There need not be a closing tag and the remainder of the document need not be a valid XML document. The document shall be stored by the assessment engine in an item bank (this can simply be a database or a file system directory).

QTI defines all content as part of the item body. The item body includes the statement of the question, inline or block feedback (which may be displayed in a different window for example) and interactions. It also includes modal feedback which is feedback which the student must dismiss before continuing. The QTI specification provides markup for the item body based upon a subset of XHTML together with additional markup for feedback and interactions. RQP supports precisely this functionality, although it does not enforce the use of QTI to describe the item.

In the case of an item template, the cloning engine instantiates the item body using the values of template variables which are randomized based upon a 32-bit unsigned integer seed which may optionally be supplied by the assessment engine. The assessment engine is required to store the seed given to each student or the values of the resulting template variables. Between servers of the same type the template variables must be instantiated with the same values for the same seed; this allows for load balancing.

The assessment engine is free to choose the seed for each user by any method it pleases.

QTI allows for modal, modeless and inline feedback. Modal feedback blocks user interaction until dismissed. Modeless and inline feedback is visible while the user interacts. Modeless feedback is not explicitly defined by QTI but we mean here feedback which is displayed separately from the body (eg. in a different window). RQP provides support for all three types of feedback. Systems which wish to provide modeless feedback and use QTI format items will need to decide on a class name to mark feedback blocks as modeless within the QTI document (or use some other mechanism); this is beyond the scope of RQP.

Hints and solutions are not explicitly supported by QTI; they are included as types of feedback. Solutions for example must be implemented as feedback blocks set to show when the builtin outcome variable 'completion_status' is set to 'complete'. RQP supports both hints and solutions. Systems which use QTI will need to decide on class names to mark feedback blocks as hints or solutions (or use some other mechanism); this is beyond the scope of RQP.

5 Issues RQP Solves

5.1 Caching

The server may implement a cache for the documents and the client may pass an empty document consisting of just an empty 'assessmentItem' tag with the 'guid' attribute set to identify the document in this case. The server may cache items, item templates and item clones independently or in one cache. The server is free to implement the caching mechanism as it pleases and the documents may be compiled into some internal format first. Note that the server is not required to implement caching.

Since each document is identified by a GUID, the cache expiry mechanism can be very simple. It is recommended that servers record the last usage time for each document in the cache and then makes room for new documents by removing the least recently used.

5.2 Instances of Items

The cloning and rendering calls operate on a fully specified item instance. An item instance is a regular item (or an item clone) or an item template together with the information required to generate the desired clone. RQP allows for simple servers which only support regular items in the rendering calls and for servers which can perform cloning implicitly during the rendering call.

Two arguments to each function are required to specify an item instance; the item source (which may be the source for an item template) and the template data (which may be empty).

The item source may be an empty document with only an empty 'assessmentItem' tag giving the GUID where the server supports caching and in this case the remainder of the document is loaded from the cache by the server (an exception is generated if the document is not in the cache or the server does not support caching).

The template data consists of a list of template variables. These can be omitted in which case random values will be generated, i.e. template processing will be performed. It is possible to seed the random number generator by setting the reserved variable '_RQPseed' with a 32-bit unsigned integer. If the seed variable is absent or zero the server sets the seed.

5.3 Rendering Formats

The rendering of an item involves the rendering of several different types of content. There are a number of different formats into which each type of content can be rendered. RQP provides a mechanism for servers to advertise what formats they support for each type of content and for clients to request certain formats.

RQP borrows from the MIME content type/subtype pair [1] definitions where possible. The types of content which RQP allows for are described below.

text Normal formatted text

equation Mathematical or other (e.g. chemical) equations

image Pictures and photographs, i.e. raster graphics

graph Line drawing or graph, i.e. vector graphics

audio Sounds

video Animation or video possibly with sound

model Three (or higher) dimensional models or environments

These are taken from the MIME media types where possible. The names are case-insensitive. RQP defines a number of format names under each of these categories. Again names are taken from the agreed MIME names where such names exist. The purpose of the MIME types are to uniquely identify formats whereas the RQP types are designed to specify the format to be used to represent content so RQP allows a format to appear under multiple content types.

The list of available format names is maintained by the RQP Foundation at `http://formats.qt-rqp.org/register/`. New format names must be registered with the RQP Foundation. Extension formats with names starting with 'x-' or 'X-' may be used by cooperating applications without registration.

5.4 Item State

A scoring engine takes information describing the item state and updates it based on the supplied responses. The assessment engine is responsible for storing the and managing the item state.

QTI also allows for the definition or outcome variables which store the item state. These are updated by the scoring engine during response processing for the item. RQP also inherits this model.

5.5 External Includes

The rendering of an item may generate external includes (for example image files generated for graphs). RQP takes the usual SOAP attachment approach where the files are made available via HTTP by the server and the URL to them is included in the fragment of markup returned. The client is expected to retrieve them within a "short period".

Static externals referenced by an item will either include an absolute URL which will remain untouched by the render call or will have a relative URL and the client supplied base will be prepended. Thus static includes are not seen by the server and remain the responsibility of the client.

6 Data Structures

RQP defines the following data structures (complex types) in addition to the basic types.

6.1 Error Object

All calls throw an exception in the event of an error using the following complex data type.

code Integer error code (unsigned 16-bit). RQP defines (will define) a number of standard error codes; see the table below. Values greater than ten thousand are reserved for item engine defined errors.

message A string giving a human-readable plain-text description of the error suitable for inclusion in the assessment engine logs.

6.2 Warning Object

All calls return warnings using an array of the following complex data types. Calls may also return additional values.

code Integer warning code (unsigned 16-bit). RQP defines (will define) a number of standard warning codes; see the table below. Values greater than ten thousand are reserved for item engine defined warnings.

message A string giving a human-readable plain-text description of the condition suitable for inclusion in the assessment engine logs.

6.3 RQPVar Object

An object to contain a variable as a name-value pair.

identifier String giving the name of the variable.

value The value of the variable as an array of strings. An empty array represents the null value.

6.4 Output Format Object

An object to list the formats supported for a given content type during rendering.

content A string giving the name of the content type.

interactiveFormats A string containing a comma separated list of format names. All formats in this list support interactive use.

staticFormats A string containing a comma separated list of format names. The formats in this list do not support interactive use.

defaultFormat A string giving the name of the default format for this content type.

7 A Simple Example

RQP has been carefully designed in an attempt to allow maximum flexibility and extensibility while keeping it simple to use for the most common, simple tasks. Before describing the calls in detail and documenting the full power available we present here an example of a very simple client RQP session in PHP.

(work in progress...)

8 Calls

RQP defines a core set of calls which all services must support and a set of calls for requesting cloning and rendering.

Exceptions shall be thrown in the event of an error with an error object (see section 6.1). Warnings are returned as an array of warning objects (see section 6.2).

The protocol defines the following calls. Note that the call, argument and return names are not final.

8.1 ServerInformation

Information about the server and which calls it implements is returned to the caller. All servers must implement this call. It takes no arguments.

8.1.1 Return Value

description Human readable string describing the server. This may include software versions for example.

cloning Boolean flag indicating whether the server supports the cloning calls.

implicitCloning Boolean flag indicating whether the server supports cloning from the other calls via the passing of a template and data. Note that servers are permitted to support implicit cloning but not cloning and vice versa.

rendering Boolean flag indicating whether the server supports the rendering call.

templateCaching Boolean flag indicating whether the server can cache item templates (only valid if Cloning is true). When true, the client can pass the template once and then only pass the GUID (until the template expires from the cache) when creating subsequent clones (eg. with different seeds for different students).

itemCaching Boolean flag indicating whether the server can cache items. When true the client can pass the item once and subsequently pass only the GUID (until the item expires from the cache).

inputFormats An array of strings listing the supported item formats, e.g. 'IMS QTI version 2.0'.

outputFormats An array of output format objects (see section 6.4) giving the supported content types together with the supported formats and default format for each type. Empty if Rendering is false. All servers which support rendering must support the text content type, the others are optional.

outputTemplates String giving a comma separated list of the supported output format templates. Present only if Rendering is true. This specification includes (will include) a number of format templates which list a number of alternative formats for each content type for certain common situations. Clients may pass the name of a template in a rendering call instead of the full format string. Servers may define additional format templates with names starting with 'x-' or 'X-'.

8.2 ItemInformation

The document passed (or the document to which the passed GUID refers) is examined and information about the item it describes is returned. All servers must implement this call.

8.2.1 Arguments

itemSource A string containing the item source. It may be an empty 'assessmentItem' tag in which case the server should load the rest of the document from the cache (if it has one and has the item in it).

cache Boolean flag, normally true. When false the item template and resulting item clone must not be cached by the server. Ignored when a document is not sent (i.e. when a GUID is used).

8.2.2 Return Value

warnings An array of warning objects. See section 6.2.

sourceCached Boolean flag indicating whether the client can pass just the GUID instead of the whole document in future calls to the server (provided the document has not dropped out of the cache by that time).

itemType A string giving the type of item passed, eg. 'QTI version 2.0'. The reserved value 'unknown' signifies that the server does not recognise the document format. The server should still set this value correctly in the case where it recognises the document but finds some syntax error or similar.

message A human-readable message indicating the errors (if any) with the document. Must not be empty when itemType is set to 'unknown'. The remaining return values are only valid when this field is empty.

template Boolean indicating if the document describes an item template.

adaptive Boolean indicating if the item (or item template) is adaptive.

timeDependent Boolean indicating if the item (or item template) is time dependent. The client is responsible for maintaining the duration and passing it to the server when this is true.

canComputerScore Boolean indicating if the item (or item template) can be scored automatically. When false the rendering call will not return any scoring information.

solutionAvailable Boolean indicating if the item (or item template) contains a model solution. When false the solution rendering call will return no content.

8.3 ProcessTemplate

Performs template processing to create random template variables for the item template. Does nothing if a regular item is passed.

8.3.1 Arguments

itemSource A string containing the item template (or regular item) source. It may be an empty 'assessmentItem' tag in which case the server should load the rest of the document from the cache (if it has one and has the item in it).

cache Boolean flag, normally true. When false the item template must not be cached by the server. Ignored when a document is not sent (i.e. when a GUID is used).

seed A 32-bit unsigned integer seed. If this argument is zero the server sets the seed.

8.3.2 Return Value

warnings An array of warning objects. See section 6.2.

sourceCached Boolean flag indicating whether the client can pass just the GUID instead of the whole document in future calls to the server (provided the document has not dropped out of the cache by that time).

seed The 32-bit unsigned integer seed used (equal to the seed argument unless that argument was zero).

templateVars An array of RQPvar objects (see section 6.3) with the values assigned to the template variables during template processing. The array will be empty if a regular item was passed.

8.4 Clone

Creates an item clone from an item template using the given template variables (or a seed). If a regular item is passed it is returned without modification. The item template (or item) passed may be cached by the server as may the resulting item clone unless the client sets a flag to disable this.

8.4.1 Arguments

itemSource A string containing the item template (or regular item) source. It may be an empty 'assessmentItem' tag in which case the server should load the rest of the document from the cache (if it has one and has the item in it).

templateData An array of RQPvar objects (see section 6.3) containing the template variables and (optionally) a 32-bit unsigned integer seed. If the array is empty or only contains the '_RQPseed' variable then template processing is performed first as if a call to ProcessTemplate were made with the seed parameter set to the value of the '_RQPseed' variable (zero if the array is empty).

cache Boolean flag, normally true. When false the item template and resulting item clone must not be cached by the server. Ignored when a document is not sent (i.e. when a GUID is used).

8.4.2 Return Value

warnings An array of warning objects. See section 6.2.

sourceCached Boolean flag indicating whether the client can pass just the GUID instead of the whole source document in future calls to the server (provided the document has not dropped out of the cache by that time).

cloneCached Boolean flag indicating whether the client can pass just the GUID instead of the whole clone document (returned in the clone field below) in future calls to the server (provided the document has not dropped out of the cache by that time).

clone The document for the item clone created.

seed The 32-bit unsigned integer seed used during template processing. Zero if template processing did not occur, otherwise equal to the '_RQPseed' variable supplied unless that variable was zero or missing.

8.5 UpdatePersistence

The persistent data for the item is updated with for the supplied values of the outcome variables. This call will mainly be used for testing during item authoring.

8.5.1 Arguments

itemSource A string containing the item (or item template) source. It may be an empty 'assessmentItem' tag in which case the server should load the rest of the document from the cache (if it has one and has the item in it).

templateData An array (possibly empty) of RQPvar objects (see section 6.3) containing the template variables and (optionally) a 32-bit unsigned integer seed. If a regular item is passed this argument is ignored. If an item template is passed (and the server supports it) cloning is performed first as if a call to Clone were made.

cache Boolean flag, normally true. When false the item template and resulting item clone must not be cached by the server. Ignored when a document is not sent (i.e. when a GUID is used).

persistence String containing the persistent data (the item state) in a server specific format. The empty string should be passed the first time the item is used.

outcomeVars An array of RQPvar objects (see section 6.3) containing the values of the outcome variables to be set for the item. These include the 'SCORE' variable for example (where the item supports computer scoring).

8.5.2 Return Value

warnings An array of warning objects. See section 6.2.

sourceCached Boolean flag indicating whether the client can pass just the GUID instead of the whole document in future calls to the server (provided the document has not dropped out of the cache by that time).

persistence String containing the persistent data (the item state) in a server specific format. The client is responsible to storing this and passing it in future calls.

8.6 Render

Response processing is performed if required. Then the item body is updated based upon the state of the item and translated into a format suitable for display or further processing. Where the server supports it, this call can also perform cloning before rendering.

8.6.1 Arguments

itemSource A string containing the item (or item template) source. It may be an empty 'assessmentItem' tag in which case the server should load the rest of the document from the cache (if it has one and has the item in it).

templateData An array (possibly empty) of RQPvar objects (see section 6.3) containing the template variables and (optionally) a 32-bit unsigned integer seed. If a regular item is passed this argument is ignored. If an item template is passed (and the server supports it) cloning is performed first as if a call to Clone were made.

cache Boolean flag, normally true. When false the item template and resulting item clone must not be cached by the server. Ignored when a document is not sent (i.e. when a GUID is used).

persistence String containing the persistent data (the item state) in a server specific format. The empty string should be passed the first time the item is used.

inputData String containing the response data in HTTP-POST format, i.e. an ampersand separated list of name and value pairs with an equals sign between the name and the value. If empty, no response processing and scoring occurs before rendering and the item state is not updated.

embedPrefix A prefix for the names of any embedded objects generated including HTML form elements.

interactive Boolean flag, normally true. When true interactions should be included in the output. When false the output should be made with the response variables printed in place of the interactions; this mode make the output 'printer friendly'. Note that this must be set to false if a non-interactive output format is requested.

renderFormat A string with a comma separated list of content formats (see section 5.3). The client can use this parameter to control which formats each type of content is rendered in. If the server can not support any of the desired formats it must return an error. Additionally a server may exclude some formats which have been marked as acceptable if they are incompatible with a choice made for some other content type (eg. if TEX selected for text output then JPEG will not be allowed as an image format). Furthermore the server must support content type template aliases for this string. For example the string 'QTI' might expand to 'text/xhtml,equation/mathml,graph/gif,image/jpeg'. Note that RQP has yet to specify any aliases, or how they are added and who defines them.

appletBase The base URL for the location of any applets which will be included.

mediaBase The base URL for the location of any media objects (such as images, embedded sounds, etc) which will be included.

8.6.2 Return Value

warnings An array of warning objects. See section 6.2.

sourceCached Boolean flag indicating whether the client can pass just the GUID instead of the whole document in future calls to the server (provided the document has not dropped out of the cache by that time).

persistence String containing the persistent data (the item state) in a server specific format. The client is responsible to storing this and passing it in future calls.

outcomeVars An array of RQPvar objects (see section 6.3) containing the current values of the outcome variables for the item. These include the 'SCORE' variable for example (where the item supports computer scoring).

head A string containing markup to be included in the head of the document being produced.

body A string containing markup to be included in the body of the document being produced.

foot A string containing markup to be included at the end of the document being produced (not necessarily in a footer section, just after all questions).

8.7 RenderSolution

The item solution is updated based upon the state of the item and translated into a format suitable for display or further processing. Where the server supports it, this call can also perform cloning before rendering.

8.7.1 Arguments

itemSource A string containing the item (or item template) source. It may be an empty 'assessmentItem' tag in which case the server should load the rest of the document from the cache (if it has one and has the item in it).

templateData An array (possibly empty) of RQPvar objects (see section 6.3) containing the template variables and (optionally) a 32-bit unsigned integer seed. If a regular item is passed this argument is ignored. If an item template is passed (and the server supports it) cloning is performed first as if a call to Clone were made.

cache Boolean flag, normally true. When false the item template and resulting item clone must not be cached by the server. Ignored when a document is not sent (i.e. when a GUID is used).

persistence String containing the persistent data (the item state) in a server specific format. The empty string should be passed the first time the item is used.

embedPrefix A prefix for the names of any embedded objects generated.

renderFormat A string with a comma separated list of content formats (see section 5.3). The client can use this parameter to control which formats each type of content is rendered in. If the server can not support any of the desired formats it must return an error. Additionally a server may exclude some formats which have been marked as acceptable if they are incompatible with a choice made for some other content type (eg. if TEX selected for text output then JPEG will not be allowed as an image format). Furthermore the server must support content type template aliases for this string. For example the string 'QTI' might expand to 'text/xhtml,equation/mathml,graph/gif,image/jpeg'. Note that RQP has yet to specify any aliases, or how they are added and who defines them.

appletBase The base URL for the location of any applets which will be included.

mediaBase The base URL for the location of any media objects (such as images, embedded sounds, etc) which will be included.

8.7.2 Return Value

warnings An array of warning objects. See section 6.2.

sourceCached Boolean flag indicating whether the client can pass just the GUID instead of the whole document in future calls to the server (provided the document has not dropped out of the cache by that time).

head A string containing markup to be included in the head of the document being produced.

body A string containing markup to be included in the body of the document being produced.

foot A string containing markup to be included at the end of the document being produced (not necessarily in a footer section, just after all questions).

References

[1] N. Freed and N. Borenstein. *RFC 2045-2049 – Multipurpose Internet Mail Extensions (MIME)*. Innosoft, First Virtual Holdings, November 1996. URL http://www.faqs.org/rfcs/rfc2045.html.

[2] Stephen Lay, editor. *IMS Question and Test Interoperability: Item Information Model*. IMS Global Learning Consortium, Inc, June 2004. Public Draft, URL http://www.imsglobal.org/question/qti_item_v2p0pd/infomodel.html.