# XPoint methods

Methods supported by the Questionmark XPoint API

Last updated 13 February 2013

The term ‘statement’ in this document refers to an activity statement as defined by the XAPI (Tin Can) 0.95 specification at http://tincanapi.wikispaces.com/Tin+Can+API+Specification+.95

## Accounts

All methods are multi-tenant, with the account ID being specified as part of the URL, e.g.

[[http://xpoint.questionmark.com/ers/](http://qmerstest.cloudapp.net/ers/)**<accountId>**/statements/](http://localhost/lrs/%3caccountId%3e/statements/)

If an accountId is specified that doesn’t match an existing account status code 404 Not found will be returned

## Statement methods

### Get

GET /statements/

Get a list of statements specified by actor name, email address or verb

Input :

name – the name of the actor

email – the email address of the actor

verb – the verb used in a statement

activity – the name of the activity used in a statement

lang – the language code for the verb and activity, defaults to ‘en-US’

$top – the maximum number of statements to return

$skip – the number of statements to omit from the beginning

$orderby – the field to use to sort the statements

$filter – an expression to be used for filtering statements, e.g. $filter=result/score/scaled+gt+0.5

These follow the OData convention - See <http://www.odata.org/developers/protocols/uri-conventions#QueryStringOptions>

All parameters are optional. Defaults are :

$top – 50

$skip – 0

$orderby – stored desc – returns most recent statements first

$filter – blank – returns all statements

Returns:

A statement list object in JSON

statements – an array of statements matching the filter

next – if more statements matching the filter are found than the limit defined, a partial URL that returns the next set of statements. If there are fewer statements found than the limit this element is omitted.

previous - a partial URL that returns the previous set of statements. If there are previous statements this element is omitted.

GET /statements/?$top=10&$skip=0&$orderby=result/score/scaled&verb=passed

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 7190

{"statements":[ array of statements ],

"next":"?$top=10&$skip=10&$orderby=result%2fscore%2fscaled&$filter="}}

### Count

GET /statements/count/

Get a count of statements specified by actor name, email address or verb

Input :

name – the name of the actor

email – the email address of the actor

verb – the verb used in a statement

lang – the language code for the verb, defaults to ‘en-US’

Returns:

the number of statements matching the filter, or all statements in the LRS if no parameters or filter defined

GET /statements/count/?$verb=passed

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 2

19

### Total

GET /statements/total/

Get a count of all statements

Input :

Returns:

the number of statements in the LRS

GET /statements/total

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 2

39

### GetById

GET /statements/statementId={statementId}

*Note no ‘?’ character to be used due to an issue with MVC routing*

Get a single statement specified by Id

Input :

statementId – the ID of an existing statement

Returns :

The statement in JSON format. If no statement is found with that ID returns 404 Not Found

GET /statements/statementId=b672d57c-eba4-4515-ac6d-bef7c6746260

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 499

{"id":"b672d57c-eba4-4515-ac6d-bef7c6746260","actor":{"objectType":"Person","mbox":[""],"name":["John"]},"verb":"completed","inProgress":false,"object":{"objectType":"Activity","id":"http://localhost/perception5/session.php/2300408448380466","definition":{"type":"assessment","name":{"en-US":"True or false"},"description":{"en-US":"True or false"}}},"result":{"score":{"scaled":1.0,"raw":1,"min":0,"max":1}},"timestamp":"2012-07-05T14:28:28.911Z","stored":"2012-07-05T14:28:30.504Z","voided":false}

### Add

POST /statements/

Add one or more statements to the LRS

Input :

An array of one or more statements in JSON format

Returns :

An array of one or more IDs corresponding to the statements added

POST /statements/

Content-Type: application/json; charset=utf-8

Content-Length: 338

[{"actor":{"mbox":["mailto:paul@questionmark.com"],"name":["Paul"]},"object":{"definition":{"description":{"en-US":"My Assessment"},"name":{"en-US":"My Assessment"},"type":"assessment"},"id":"http:\/\/localhost\/perception5\/session.php\/6912287363757842"},"result":{"score":{"max":4,"min":0,"raw":1,"scaled":"1.05"}},"verb":"passed"}]

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 40

["35dd9f2a-3a1c-4d4d-a728-f651b19a13d8"]

### Store

PUT /statements/statementId=

Adds a statement to the LRS if it doesn’t already exist. Note does NOT modify existing statements

Input :

statementId – the ID of the statement to add

A single statement in JSON format

Returns :

204 No Content if no statement with that ID existed and the statement was added

409 Conflict if a statement with that ID exists already

PUT /statements/statementId=abc123

Content-Type: application/json; charset=utf-8

Host: localhost

Content-Length: 335

{"actor":{"mbox":["mailto:paul@questionmark.com"],"name":["Paul"]},"object":{"definition":{"description":{"en-US":"My Assessment"},"name":{"en-US":"My Assessment"},"type":"assessment"},"id":"http:\/\/localhost\/perception5\/session.php\/6912287363757842"},"result":{"score":{"max":4,"min":0,"raw":1,"scaled":"1.05"}},"verb":"passed"}

HTTP/1.1 204 No Content

### Delete

DELETE /statements/statementId=

Remove an existing statement from the LRS

Input :

statementId – the ID of an existing statement

Returns :

204 No Content if the statement was found and deleted

404 Not Found if no statement with that ID was found

DELETE /statements/statementId=abc123

HTTP/1.1 204 No Content

### GetActor

GET /statements/actor/

Get a list of statements specified by actor name or email address, and optionally verb

Input :

name – the name of the actor

email – the email address of the actor

verb – the verb used in a statement

$top – the maximum number of statements to return

$skip – the number of statements to omit from the beginning

$orderby – the field to use to sort the statements

$filter – an expression to be used for filtering statements

One of name or email must be specified.

Returns :

A statement list object in JSON containing all statements that match the actor’s name and/or email address and the verb if specified

next – if more statements matching the filter are found than the limit defined, a partial URL that returns the next set of statements. If there are fewer statements found than the limit this element is omitted.

previous - a partial URL that returns the previous set of statements. If there are previous statements this element is omitted.

404 Not Found if no matching statements found

204 No Content if neither name or email are specified

GET /statements/actor/?name=Paul&verb=passed

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 3074

{"statements":[ array of statements]}

### CountActor

GET /statements/actor/count/

Get a count of statements specified by actor name or email address, and optionally verb

Input :

name – the name of the actor

email – the email address of the actor

verb – the verb used in a statement

One of name or email must be specified.

Returns :

The number of statements that match the actor’s name and/or email address and the verb if specified

GET /statements/actor/count/?name=Paul&verb=passed

HTTP/1.1 200 OK

Content-Length: 1

6

### TotalActor

GET /statements/actor/total/

Get a count of the number of unique actors in the LRS distinguished by actor name

Input :

Returns :

The number of unique actors in the LRS distinguished by actor name

GET /statements/actor/total

HTTP/1.1 200 OK

Content-Length: 2

16

### SummaryActor

GET /statements/actor/summary/

Get a list of unique actors in the LRS distinguished by activity name, optionally restricted to those involved in statements specified by activity or verb

Input :

activityId – the ID of an activity

activityName – the name of an activity

activityType – the type of an activity, e.g. ‘assessment’

verb – the name a verb

verbId – the id a verb

lang – the language in which the name of the activity or verb is described, defaults to ‘en-US’

$top – the maximum number of statements to return

$skip – the number of statements to omit from the beginning

Returns :

A array of unique actors involved in statements that match the activity and/or verb specified, including a count of the number of statements involving each actor

A count of the number of unique actors found

A count of the total number of statements found

GET /statements/actor/summary?verb=created

HTTP/1.1 200 OK

Content-Length: 570

{

"actors": [

{

"count": 2,

"objectType": "Person",

"mbox": "",

"name": "Paul"

},

{

"count": 1,

"objectType": "Person",

"mbox": "",

"name": "Jim"

],

"count": 2,

"total": 3

}

### GetActivity

GET /statements/activity/

Get a list of statements specified by activity name or id, and optionally verb

Input :

name – the name of the activity

lang – the language in which the name of the activity is described, defaults to ‘en-US’

id – the ID of the activity

verb – the verb used in a statement

$top – the maximum number of statements to return

$skip – the number of statements to omit from the beginning

$orderby – the field to use to sort the statements

$filter – an expression to be used for filtering statements

One of name or id must be specified.

Returns :

A statement list object in JSON containing all statements that match the activity name and/or id and the verb if specified

next – if more statements matching the filter are found than the limit defined, a partial URL that returns the next set of statements. If there are fewer statements found than the limit this element is omitted.

previous - a partial URL that returns the previous set of statements. If there are previous statements this element is omitted.

404 Not Found if no matching statements found

204 No Content if neither name or id are specified

GET /statements/activity/?name=Simple+Example+Assessment&lang=en-US&verb=passed

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 12452

{"statements":[ array of statements ]}

### CountActivity

GET /statements/activity/count/

Get a list of statements specified by activity name or id, and optionally verb

Input :

name – the name of the activity

lang – the language in which the name of the activity is described, defaults to ‘en-US’

id – the ID of the activity

verb – the verb used in a statement

Returns :

The number of statements that match the activity name and/or id and the verb if specified

204 No Content if neither name or id are specified

GET /statements/activity/count/?name=Simple+Example+Assessment&lang=en-US&verb=passed

HTTP/1.1 200 OK

Content-Length: 2

18

### TotalActivity

GET /statements/activity/total/

Get a count of the number of unique activities in the LRS distinguished by activity id

Input :

Returns :

The number of unique activities in the LRS distinguished by activity id

GET /statements/activity/total

HTTP/1.1 200 OK

Content-Length: 2

36

### SummaryActivity

GET /statements/activity/summary/

Get a list of unique activities in the LRS distinguished by activity id, optionally restricted to those involved in statements specified by actor or verb

Input :

name – the name of an actor

email – the email address of an actor

verb – the name a verb

verbId – the id a verb

lang – the language in which the name of the activity or verb is described, defaults to ‘en-US’

$top – the maximum number of statements to return

$skip – the number of statements to omit from the beginning

Returns :

A array of unique actvities involved in statements that match the actor and/or verb specified, including a count of the number of statements involving each activity

A count of the number of unique activities found

A count of the total number of statements found

GET /statements/activity/summary?verb=created

HTTP/1.1 200 OK

Content-Length: 510

{

"activities": [

{

"count": 2,

"objectType": "Activity",

"id": "http://example.scorm.com/tincan/example/simplestatement",

"definition": {

"name": {

"en-US": "simple statement"

},

"description": {}

}

},

{

"count": 2,

"objectType": "Activity",

"definition": {

"name": {

"en-US": "test "

},

"description": {}

}

}

],

"count": 2,

"total": 4

}

### TotalVerb

GET /statements/verb/total/

Get a count of the number of unique verbs in the LRS distinguished by verb id

Input :

Returns :

The number of unique verbs in the LRS distinguished by verbs id

GET /statements/verb/total

HTTP/1.1 200 OK

Content-Length: 2

20

### SummaryVerb

GET /statements/verb/summary/

Get a list of unique verbs in the LRS distinguished by verb id, optionally restricted to those involved in statements specified by activity or actor

Input :

name – the name of an actor

email – the email address of an actor

activityId – the ID of an activity

activityName – the name of an activity

activityType – the type of an activity, e.g. ‘assessment’

lang – the language in which the name of the activity is described, defaults to ‘en-US’

$top – the maximum number of statements to return

$skip – the number of statements to omit from the beginning

Returns :

An array of unique verbs involved in statements that match the activity and/or actor specified, including a count of the number of statements involving each verb

A count of the number of unique verbs found

A count of the total number of statements found

GET /statements/verb/summary?name=Paul

HTTP/1.1 200 OK

Content-Length: 510

{

"verbs": [

{

"count": 5,

"id": "http://adlnet.gov/expapi/verbs/completed",

"display": {

"en-US": "completed"

}

},

{

"count": 17,

"id": "http://adlnet.gov/expapi/verbs/failed",

"display": {

"en-US": "failed"

}

},

{

"count": 10,

"id": "http://adlnet.gov/expapi/verbs/passed",

"display": {

"en-US": "passed"

}

}

],

"count": 3,

"total": 32

}

### Notify

POST /notify/

Adds a statement using data passed by HTTP POST

Input :

UserName – the name of the actor

UserEmail – email address of the actor

Provider – URL of the activity provider

Id – ID of the activity

Title – display name for the activity

PercentScore – percentage score achieved

RawScore – score in points achieved

MaxScore – maximum possible score in points

Outcome – outcome of the activity (the verb for the statement)

Returns:

The statement added in JSON

POST /notify/

Content-Type: application/x-www-form-urlencoded

UserName=Paul&Url=http%3A%2F%2Flocalhost%2Fperception5%2Fsession%2Ephp&UserEmail=paul%40questionmark%2Ecom&PercentScore=100&Outcome=passed&Id=3657891228763742&RawScore=4&MaxScore=4&Title=My%20Assessment

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

{"id":"dc8d82fc-c7b6-4297-ae40-9402e702be70","actor":{"objectType":"Person","mbox":["mailto:paul@questionmark.com"],"name":["Paul"]},"verb":"passed","inProgress":false,"object":{"objectType":"Activity","id":"http://localhost/perception5/session.php/3657891228763742","definition":{"type":"assessment","name":{"en-US":"My Assessment"},"description":{"en-US":"My Assessment"}}},"result":{"score":{"scaled":1.0,"raw":4,"min":0,"max":4}},"timestamp":"2012-07-06T09:27:20.715093Z","stored":"2012-07-06T09:27:20.715093Z","voided":false}

## Authentication

Statement requests are authenticated using Basic authentication - <http://en.wikipedia.org/wiki/Basic_access_authentication>

Request headers should include

Authorization: Basic AuthString

Where AuthString is a Base64 encoding of the account ID and the key for that account separated by a colon

Account ID = ‘example’

Key = ‘xJAko6XKODyiVHZd’

string accountAndKey = "example" + ":" + " xJAko6XKODyiVHZd ";

string authString =  Convert.ToBase64String(new ASCIIEncoding().GetBytes(accountAndKey));

giving the header

Authorization: Basic ZXhhbXBsZTogeEpBa282WEtPRHlpVkhaZCA=

Account requests must use the ID and key of the special admin account.

Requests without the Basic authentication header will return

401 Authorization Required

WWW-Authenticate: Basic