WebLabUX

Web Service Specification

Winter Quarter 2014

Table of Contents

Design Patterns 4

Data response buffers 4

Error response buffers 4

Debug response buffers 4

Account access and authorization 5

Reference accounts 6

Web Service Endpoints 7

account 7

user 7

changePassword 10

debug 10

config 10

gratuity 12

gratuity 12

study 13

log 13

allStudies 14

load 14

session 15

sessions 16

study 17

transition 19

session 20

config 20

currentStep 21

start 21

startNextStep 22

finishCurrentStep 23

signIn 24

signOut 25

study 26

allStudyIds 26

config 26

general 29

measure 31

name 32

period 33

step 35

variable 36

variation 38

# Design Patterns

This section contains information that applies to all web services in this specification.

## Data response buffers

The requested data is returned in the “data” object of the response buffer as this example shows.

{

"data": [

{

"studyId": "12345",

"periodName": "AutomatedTest",

"email": "test@example.com",

"comments": null

}

]

}

## Error response buffers

All calls should return an HTTP error status of 200 whenever possible. Errors that occur within the command should return the corresponding HTTP error and message in an “error” object as this example shows.

{

"error": {

"status": 501,

"message": "Command not recognized.",

}

}

* **Status** = HTTP Status of request
* **Message** = Error explanation in text

## Debug response buffers

These can be defined as required by the module. The only requirement is that the debug information is included in the “debug” object as shown in this example.

The debug buffer should also include a “module” field that contains the filename of the module returning the debug buffer.

{

"debug": {

"module": "C:\\wlux\_web\_service\\website\\data\\gratuity\_get\_study.php",

"cmdData": {

"studyId": "12345",

"periodName": "AutomatedTest"

},

"sqlSelect1": {

"sqlQuery": "SELECT studyId, periodName, email, comments FROM gratuity\_log WHERE studyId = 12345 AND periodName = \"AutomatedTest\"",

"sqlError": "00000",

"message": ""

}

},

"data": [

{

"studyId": "12345",

"periodName": "AutomatedTest",

"email": "test@example.com",

"comments": null

}

]

}

## Account access and authorization

Initially, only basic account authorization is supported as follows:

1. The user must log into the WebLabUX web site using their web site credentials. This is configured through the web server (e.g. .htaccess) and provides access to the WebLabUX app and home page.
2. On the WebLabUX home page, the user must log in to their WebLabUX account. This limits their access to WebLabUX information as follows:
   1. Admin accounts: all access to all information
   2. Researcher accounts: all access to protected information they own (ownerId = the researcher’s accountId)
   3. Reviewer accounts: read access to protected information to which a Researcher or Admin account has granted access. (This can be added later. For now, admin and researcher access are sufficient)
   4. Researcher and reviewer account access is exclusive. If a researcher or reviewer is signed in, no one else can log into the same account until they sign out or an admin signs them out. Admin accounts do not have exclusive access. *With great power comes great responsibility.*
3. WebLabUX access is managed in the UI. The UI must pass an authorization header in each call to the web service that requires one. If the authorization header is valid, the service will respond to the command; otherwise, it will return a 401 status code.
4. Web site access is managed and configured through the web server.

Access to protected information is determined by the authorization header passed in the request. The calling application must supply a valid authorization header when requesting protected information. When protected information is requested:

1. The Authentication header is parsed
2. The username and password are decoded
3. The username and password are used to determine the allowed access of that account
4. The access allowed is compared to the request and
   1. The request is allowed and processed
   2. The request is denied and a 401 Unauthorized status is returned

### Reference accounts

* defaultResearcher:1Password = ZGVmYXVsdFJlc2VhcmNoZXI6MVBhc3N3b3Jk
* defaultAdmin:1Password = ZGVmYXVsdEFkbWluOjFQYXNzd29yZA==

# Web Service Endpoints

This section describes the endpoints supported by the web service.

## account

Information about a WebLabUX user.

Protected data—Authorization required

### user

#### GET

Gets information about a user.

Protected data—Read access required

The caller must be an admin or a user requesting their own account.

An account is identified uniquely by its *accountId* or its *username*.

The password is not returned because it is encrypted when it is stored.

##### REQUEST

{

"user": {

"username": "ResearcherBob"

“ownerId” : 9999

}

}

##### RESPONSE

{

"data": {

"accountId": 23,

"username": "ResearcherBob",

“ownerId” : 9999,

"firstName": "Bob",

"lastName": "Researcher",

"greetingName": "Bob",

"orgName": "UW - HCDE",

"email": "bob@example.com",

"accountType": "researcher",

"wluxUrlRoot": "http://wlux.uw.edu/weblabux",

"clientUrlRoot": "http://mytestsite.example.com",

"userFileRoot":"/usr/web/files",

"userWebRoot": "/usr/web/www/files",

"defaultTimeZone": "UTC+08",

"dateCreated": 9992999203,

"dateModified": 9992999203

}

}

##### ERRORS

* 404 - Not Found
* 401 - Not Authorized

#### POST

Creates a new user account.

Protected data—Admin access required

The password field is encrypted before it is stored in the account database.

##### REQUEST

{

"user": {

"username": "ResearcherBob",

"ownerId" : 9999,

"password": "Plain text password",

"firstName": "Bob",

"lastName": "Researcher",

"greetingName": "Bob",

"orgName": "UW - HCDE",

"email": "bob@example.com",

"accountType": "researcher",

"wluxUrlRoot": "http://wlux.uw.edu/weblabux",

"clientUrlRoot": "http://mytestsite.example.com",

"userFileRoot": "/User/WebLabUx/UserFiles/Researcher/Files",

"userWebRoot": "/Server/WebLabUx/UserFiles/Researcher/css",

"userFileRoot":"/usr/web/files",

"userWebRoot": "/usr/web/www/files",

"defaultTimeZone": "UTC+08"

}

}

##### RESPONSE

{

"data": {

"accountId": 23,

"username": "ResearcherBob",

“ownerId” : 9999,

"password": "132eerl3uio34()\*()\*)22980",

"firstName": "Bob",

"lastName": "Researcher",

"greetingName": "Bob",

"orgName": "UW - HCDE",

"email": "bob@example.com",

"accountType": "researcher",

"wluxUrlRoot": "http://wlux.uw.edu/weblabux",

"clientUrlRoot": "http://mytestsite.example.com",

"userFileRoot": "/User/WebLabUx/UserFiles/Researcher/Files",

"userWebRoot": "/Server/WebLabUx/UserFiles/Researcher/css",

"userFileRoot":"/usr/web/files",

"userWebRoot": "/usr/web/www/files",

"defaultTimeZone": "UTC+08",

"dateCreated": 9992999203,

"dateModified": 9992999203

}

}

##### ERRORS

* 401 - Not Authorized

#### PUT

Updates an existing user account.

Protected data—Update access required

The password field cannot be changed with this method.

##### REQUEST

{

"user": {

"username": "ResearcherBob",

“ownerId” : 9999,

"orgName": "UW – Engineering”

}

}

##### RESPONSE

{

"data": {

"accountId": 23,

"username": "ResearcherBob",

“ownerId” : 9999,

"firstName": "Bob",

"lastName": "Researcher",

"greetingName": "Bob",

"orgName": "UW - Engineering",

"email": "bob@example.com",

"accountType": "researcher",

"wluxUrlRoot": "http://wlux.uw.edu/weblabux",

"clientUrlRoot": "http://mytestsite.example.com",

"userFileRoot":"/usr/web/files",

"userWebRoot": "/usr/web/www/files",

"defaultTimeZone": "UTC+08",

"dateCreated": 9992999203,

"dateModified": 9992999203

}

}

##### ERRORS

* 404 - Not Found
* 401 - Not Authorized

#### DELETE

Deletes an existing user account.

Protected data—Admin access required

##### REQUEST

{

"user": {

"username": "ResearcherBob",

“ownerId” : 9999

}

}

##### RESPONSE

{

"data": {

"deleted": 1

}

}

##### ERRORS

* 404 - Not Found
* 401 - Not Authorized

### changePassword

Changes the password of an existing account.

#### POST

Protected data—Update access required

##### REQUEST

{

"changePassword": {

"username": "ResearcherBob",

"currentPassword": "myoldpassword",

"newPassword": "MyNewPasswordInPlainText”

}

}

##### RESPONSE

{

"data": {

"accountId": 23,

"username": "ResearcherBob"

}

}

##### ERRORS

* 404 - Not Found
* 401 - Not Authorized

## debug

control’s system debug messaging

### config

#### GET

Get the current debug info state of each module.

Protected data—Admin access required

REQUEST

{

"config":{}

}

RESPONSE

{

"data": {

"account": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"gratuity": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"log": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"session": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"signIn": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"signOut": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"study": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

}

}

}

ERROR

* 401 – Not Authorized

#### POST

REQUEST

Set the current debug info state of one or more modules module.

Protected data—Admin access required

{

"config": {

" account ": {

"POST": { "debug": 1, "sql": 0 },

"GET": { "debug": 1, "sql": 0 }

},

" gratuity ": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 }

}

}

}

RESPONSE

{

"data": {

"account": {

"POST": { "debug": 1, "sql": 0 },

"GET": { "debug": 1, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

},

"gratuity": {

"POST": { "debug": 0, "sql": 0 },

"GET": { "debug": 0, "sql": 0 },

"PUT": { "debug": 0, "sql": 0 },

"DELETE": { "debug": 0, "sql": 0 }

}

}

}

ERROR

* 401 - Not Authorized

## gratuity

Information used to collect and send gratuity information to participants

### gratuity

record a participant’s gratuity information

#### POST

##### REQUEST

Unprotected data—No authorization header required

{

"gratuity": {

"studyId": 1234,

"periodName": "Freshmen 2014",

"email": "you@me.com",

"comments": "I hope this really works!!!!"

}

}

##### RESPONSE

{

"data": {

"studyId": 1234,

"periodName": "Freshmen 2014",

"email": "you@me.com",

"comments": "I hope this really works!!!!"

}

}

##### ERRORS

* 400 – Bad Request
  + - Study period not active
    - Study not found

### study

Gets the participants who registered for a gratuity

#### GET

REQUEST

Protected data—Read access required

Only studies to which the caller has access are returned.

{

"study": {

"studyId": 1234,

"periodName": "Freshmen 2014

}

}

RESPONSE

{

"data": [

{

"studyId": "1234",

"periodName": "Freshmen 2014",

"email": "me@you.com",

"comments": "I hope this really works!"

},

{

"studyId": "1234",

"periodName": "Freshmen 2014",

"email": "you@me.com",

"comments": "I hope this really works!!!!"

}

]

}

ERRORS

* 404 – Not Found
  + - Study not found
    - period not found

## log

NEEDS REVIEW AND/OR REDESIGN

Information about the sessions and tasks that have been performed.

### allStudies

Information about the entries in the logs

#### GET

Get a list of sessions or studies with log records.

Protected data—Read access required

Only studies to which the caller has access are returned.

##### REQUEST

{

"allStudies": { }

}

##### RESPONSE

{

"data": {

"count": 4,

"studyIds": [

"1234",

"2525",

"1999",

"2001"

]

}

}

##### ERRORS

* 401 – Not Authorized
* 404 – Not found

### load

Record pages read on the client site

#### POST

Create a new log record for a page load.

Unprotected data—No authorization header required

(This is the same functionality that the log- POST-open method provided)

##### REQUEST

{

"load": {

"clientTimestamp": 1375043148,

"sessionId": 1377397539,

"pageUrl": "http://students.washington.edu/rbwatson/hearts.html"

}

}

##### RESPONSE

{

"data": {}

}

##### ERRORS

None

If a non-existent session is specified, record the data submitted and leave the derived data empty/null.

### session

Session log information

#### GET

Gets information from the log about the specified participant session

Protected data—Read access required

##### REQUEST

{

"session": {

"sessionId": 1377453392

}

}

##### RESPONSE

{

"data": {

"studyId": "1234",

"schdulePeriodName": "Freshman 2014",

"sessionId": "1377453392",

"studyVariationNo": 3,

"startTime": "2013-08-24 19:46:07",

"endTime": "2013-08-24 19:48:12",

"tasks": [

{

"taskId": "1",

"taskVariationNo": 1,

"startTime": "2013-08-24 19:46:10",

"endTime": "2013-08-24 19:46:42"

},

{

"taskId": "2",

"taskVariationNo": 1,

"startTime": "2013-08-24 19:47:07",

"endTime": "2013-08-24 19:48:12"

}

]

}

}

##### ERRORS

* 404 – Not Found
  + - Session not found
* 401 - Not Authorized

### sessions

Information about the sessions of a specific study

#### GET

Gets information from the log about the specified session or sessions

Protected data—Read access required

##### REQUEST

This requests gets all the sessions recorded for this study during the specified schedule period.

{

"sessions": {

"studyId": 1234,

"schdulePeriodName": "Freshman 2014"

}

}

##### RESPONSE

{

"data": {

"studyId": "1234",

"schdulePeriodName": "Freshman 2014",

"sessions": [

{

"sessionId": "1377397537",

"studyVariationNo": 3,

"startTime": "2013-08-24 19:46:07",

"endTime": "2013-08-24 19:48:12",

"tasks": [

{

"taskId": "1",

"taskVariationNo": 1,

"startTime": "2013-08-24 19:46:10",

"endTime": "2013-08-24 19:46:42"

},

{

"taskId": "2",

"taskVariationNo": 1,

"startTime": "2013-08-24 19:47:07",

"endTime": "2013-08-24 19:48:12"

}

]

},

{

"sessionId": "1377397538",

"studyVariationNo": 1,

"startTime": "2013-08-25 19:46:07",

"endTime": "2013-08-25 19:48:12",

"tasks": [

{

"taskId": "1",

"taskVariationNo": 1,

"startTime": "2013-08-25 19:46:10",

"endTime": "2013-08-25 19:46:42"

},

{

"taskId": "2",

"taskVariationNo": 1,

"startTime": "2013-08-25 19:47:07",

"endTime": "2013-08-25 19:48:12"

}

]

},

{

"sessionId": "1377397539",

"studyVariationNo": 2,

"startTime": "2013-08-26 19:46:07",

"endTime": "2013-08-26 19:48:12",

"tasks": [

{

"taskId": "1",

"taskVariationNo": 1,

"startTime": "2013-08-26 19:46:10",

"endTime": "2013-08-26 19:46:42"

},

{

"taskId": "2",

"taskVariationNo": 1,

"startTime": "2013-08-26 19:47:07",

"endTime": "2013-08-26 19:48:12"

}

]

}

]

}

}

### study

Get transaction log data for the specified study

#### GET

Gets information from the log about the specified study

Protected data—Read access required

Only studies to which the caller has access are returned.

##### REQUEST

{

"study": {

"studyId": 1234,

"schdulePeriodName": "Freshman 2014"

}

}

##### RESPONSE

{

"data": {

"studyId": 1234,

"schdulePeriodName": "Freshman 2014",

"sessions": [

{

"sessionId": "1377397537",

"studyVariationNo": 3,

"tasks": [

{

"taskId": 1,

"taskVariationNo": 3,

"taskLog": [

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

},

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

}

]

},

{

"taskId": 2,

"taskVariationNo": 3,

"taskLog": [

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

},

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

}

]

}

]

},

{

"sessionId": "1377397533",

"studyVariationNo": 2,

"tasks": [

{

"taskId": 1,

"taskVariationNo": 3,

"taskLog": [

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

},

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

}

]

},

{

"taskId": 2,

"taskVariationNo": 3,

"taskLog": [

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

},

{

"serverTimestamp": 1375043148000,

"clientTimestamp": 1375043148,

"recordType": "transition",

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

}

]

}

]

}

]

}

}

##### ERRORS

* 401 – Not Authorized
* 404 – Not found

### transition

Record page interactions that leave one page to go to another

#### POST

Create a new interaction record.

Unprotected data—No authorization header required

##### REQUEST

{

"transition": {

"clientTimestamp": 1375043148,

"sessionId": 1377397539,

"fromUrl": "http://students.washington.edu/rbwatson/hearts.html",

"toUrl": "http://students.washington.edu/rbwatson/spades.html",

"linkClass": "a\_class",

"linkId": "a\_Id",

"linkTag": "a"

}

}

##### RESPONSE

{

"data": {}

}

##### ERRORS

None

If a non-existent session is specified, record the data submitted and leave the derived data empty/null.

## session

Manages participant sessions.

### config

Information about the session configuration.

#### GET

Gets data about the current session configuration. This command is intended to be called from a client site under test to configure the site’s variation parameters

Unprotected data—No authorization header required

##### REQUEST

{

"config": {

"sessionId": 223334056

}

}

##### RESPONSE

{

"data": {

"sessionId": 223334056,

"studyId": 1234,

"stepIndex": 2,

"variationCssUrls": [

"http://wlux.uw.edu/weblabux/css/1234/3/2/1/var.css",

"http://wlux.uw.edu/weblabux/css/1234/3/3/3/var.css",

"http://wlux.uw.edu/weblabux/css/1234/3/1/1/var.css"

],

"taskBarCssUrl": "http://wlux.uw.edu/weblabux/css/taskBar.css",

"taskReturnUrl": "http://wlux.uw.edu/weblabux/finishTask.html",

"exitButtonText": "Exit task",

"tabShowText": "show",

"tabHideText": "hide",

"taskHtml": "<p>Do the task, please</p>"

}

}

##### ERRORS

* 404 – Not Found
  + - Session not found

### currentStep

Gets the configuration for the pre/post-task pages used by the current step.

#### GET

Get information about the current session step

REQUEST

{

"currentStep": {

"sessionId": 223334056

}

}

##### RESPONSE

{

"data": {

"studyId": "1234",

"sessionId": "1377453392",

"stepIndex": 1,

"preTaskPageUrl": "http://wlux.uw.edu/wluxStepStart.php",

"preTaskPageHtml": "<p>These are the instructions for Task 1</p>",

"preTaskPageNextUrl": "http://students.washington.edu/rbwatson/hearts.html",

"postTaskPageUrl": "http://wlux.uw.edu/wluxStepFinish.php",

"postTaskPageHtml": "<p>These are the instructions for end of Task 1</p>",

"postTaskPageNextUrl": "http://wlux.uw.edu/wluxStepStart.php"

}

}

##### ERRORS

* 404 – Not Found
  + - Session not found
    - No more tasks

### start

Starts a new participant session in accordance with the study definition

#### POST

Create a new participant session for the specified study,

Unprotected data—No authorization header required

##### REQUEST

{

"start": {

"studyId": 1234

}

}

##### RESPONSE

{

"data": {

"studyId": 1234,

"sessionId": 1377453392,

"studyVariationNo": 3,

"startTime": "2013-08-25 10:56:32",

"pageUrl": "http://wlux.uw.edu/wluxStepStart.php",

"pageHtml": "<p>These are the instructions for Task 1</p>",

"pageNextUrl": "http://wlux.uw.edu/wlux/stepStartPage.php"

}

}

##### ERRORS

* 404 – Not Found
  + - Study not found

### startNextStep

#### POST

Starts the next task in the specified session according to the sequencing logic

Unprotected data—No authorization header required

##### REQUEST

{

"startNextStep ": {

"sessionId": 1377453392

}

}

##### RESPONSE

{

"data": {

"studyId": "1234",

"sessionId": "1377453392",

"stepIndex": 1,

"preTaskPageUrl": "http://wlux.uw.edu/wluxStepStart.php",

"preTaskPageHtml": "<p>These are the instructions for Task 1</p>",

"preTaskPageNextUrl": "http://students.washington.edu/rbwatson/hearts.html",

"studyTaskPageUrl": "http://students.washington.edu/rbwatson/hearts.html",

"postTaskPageUrl": "http://wlux.uw.edu/wluxStepFinish.php",

"postTaskPageHtml": "<p>These are the instructions for end of Task 1</p>",

"postTaskPageNextUrl": "http://wlux.uw.edu/wluxStepStart.php"

}

}

If a new task is started before the previous task was finished:

* the previous task is finished
* The next task is started
* the data from the previous task is returned with that of the new task

{

"data": {

"studyId": "1234",

"sessionId": "1377453392",

"stepIndex": 1,

"preTaskPageUrl": "http://wlux.uw.edu/wluxStepStart.php",

"preTaskPageHtml": "<p>These are the instructions for Task 1</p>",

"preTaskPageNextUrl": "http://students.washington.edu/rbwatson/hearts.html",

"studyTaskPageUrl": "http://students.washington.edu/rbwatson/hearts.html",

"postTaskPageUrl": "http://wlux.uw.edu/wluxStepFinish.php",

"postTaskPageHtml": "<p>These are the instructions for end of Task 1</p>",

"postTaskPageNextUrl": "http://wlux.uw.edu/wluxStepStart.php"

},

"finishData": {

"taskId": "1",

"finishTime": "2013-08-25 11:27:19",

}

}

##### ERRORS

* 404 – Not Found
  + - Session not found
    - No more tasks

The 404 response message includes a link to the finish page, in case that’s necessary.

{

"error": {

"status": 404,

"message": "No more steps”,

"finishPageUrl": "http://wlux.uw.edu/wluxStudyFinish.php"

}

}

##### Remarks

A step can have the following properties.

1. If (preTaskPageHtml != null)
   1. Display preTaskPageHtml in content frame
   2. Set link of **Next** button to preTaskPageNextUrl
2. If (preTaskPageHtml == null)   
   AND (studyTaskPageUrl != null)
   1. Navigate directly to studyTaskPageUrl
3. If (preTaskPageHtml == null)   
   AND (studyTaskPageUrl == null)   
   AND (postTaskPageHtml != null)
   1. Display postTaskPageHtml in content frame
   2. Set link of **Next** button to postTaskPageUrl
4. Any other configuration found in the database results in a 500 Server error. (Any combinations of these fields that are not supported should be prevented when the configuration is stored—i.e. it should have resulted in a 400 Bad request error

### finishCurrentStep

#### POST

Finishes the task in in progress. Finishing the last task also finishes the session.

Unprotected data—No authorization header required

##### REQUEST

{

" finishCurrentStep": {

"sessionId": 1377453392

}

}

##### RESPONSE

{

"data": {

"taskId": 2,

"finishTime": "2013-08-25 11:32:44",

"postTaskPageUrl": "http://wlux.uw.edu/wluxStepFinish.php",

"postTaskPageHtml": "<p>You finished Task 1</p>",

"postTaskPageNextUrl": "http://wlux.uw.edu/wluxStepStart.php"

}

}

##### ERRORS

* 404 – Not Found
  + - Session not found

## signIn

Authorizes a user account.

#### POST

Returns an access token if the user account and credentials are valid.

Unprotected data— No authorization header required

##### REQUEST

{

"signIn": {

"username": "ResearcherBob",

“password” : “Plain text password”

}

}

##### RESPONSE

{

"data": {

"username": "ResearcherBob",

"accountId" : 23,

"token": " QWxhZGRpbjpvcGVuIHNlc2FtZQ=="

}

}

Initially, the token will be the encoded data used in the authorization header. E.g.

Authorization: Basic <token value>

Using the response buffer from above, the Authentication header used by this account to make subsequent calls to the WebLabUX web service functions would contain:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

##### ERRORS

* 401 - Not Authorized (login failure)
* 400 – Bad request
  + - user already signed in

## signOut

Cancels an active access token.

~~NOTE: I’m not sure how or if this is necessary within the current authorization scheme. As it is, the authorization is done per call, so the sign out functionality is contained in the browser/app. When the user signs out of the app, the app stops using the user’s credentials when interacting with the data service. I’ll leave it in for now, but it could be removed in future updates of the spec.~~

Sign out is required to support exclusive access to an account.

#### POST

Ends the specified token’s access.

Protected data—authorization header required

* Admin can force a sign out of any user
* Researcher and reviewer can only sign out the account identified by the token.

##### REQUEST

Only one of these two fields is required. If both are specified, the token takes precedence.

{

"signOut": {

"username": "ResearcherBob",

"token": "sdfewrpoip3240-9v0-d-"

}

}

##### RESPONSE

{

"data": {}

}

##### ERRORS

* 400 – Bad request
  + - User not logged in
* 401 – Not authorized (when token doesn’t have access to sign out the account).

##### 

## study

Information about a study definition (configuration)

Protected data—authorization header required

### allStudyIds

Gets summary information about the studies defined in the database.

Protected data—Authorization required

#### GET

Get the requested study or study element IDs.

Protected data—Read access required

##### REQUEST

{

"allStudyIds": {}

}

##### RESPONSE

Uses the account context of the call and returns the IDs of the studies defined for that account.

{

"data": {

"count": 4,

"studyIds": [

"1234",

"2525",

"1999",

"2001"

]

}

}

##### ERRORS

* 401 - Not Authorized

### config

A complete list of the elements of a study definition

A study can be specified by the *studyId* (alone) or by the *ownerID* and *studyName*.

#### GET

Gets a complete study definition

Protected data—Read access required

##### REQUEST

{

"conifg": {

"studyId": 9999

}

}

Or

{

"conifg": {

"studyName": "My favorite study",

"ownerId": 1234

}

}

##### RESPONSE

{

"data": {

"studyId": 9999,

"ownerId": 1234,

"studyName": "My favorite study",

"studyDescription": "A long description",

"researcherOrg": "org",

"researcherFirstName": "Alex",

"researcherLastName": "Researcher",

"researcherEmail": "aResearcher@example.com",

"researcherAddress": null,

"ethicsBoardEmail": null,

"defaultTimeZone": "UTC+08",

"defaultVariationMethod": "randomSequential",

"defaultTaskBarRoot": "wlux",

"defaultTaskBarCssPath": "css\\taskbar.css",

"defaultVariationRoot": "client",

"defaultVariationCssPath": "css\\vars\\var1.css",

"defaultExitButtonText": "Exit task",

"defaultTabShowText": "Show",

"defaultTabHideText": "Hide",

"clientTimeStampFormat": "unixtime",

"dateCreated": 9992999203,

"dateModified": 9992999203,

"studyStatus": "draft",

"studyVariations": [

{

"studyVariationId": 9938443,

"studyVariableId": 1343334,

"variationName": "Pointy Finger",

"variationDesc": "The pointy finger cursor",

"variationRoot": "wlux",

"variationCssPath": "css/finger.css"

},

{

"studyVariationId": 9938443,

"studyVariableId": 1343335,

"variationName": "I-beam cursor",

"variationDesc": "The I-beam cursor",

"variationRoot": "wlux",

"variationCssPath": "css/ibeam.css"

},

{

"studyVariationId": 1343334,

"studyVariableId": 1343336,

"variationName": "Drop down",

"variationDesc": "Drop down menus",

"variationRoot": "wlux",

"variationCssPath": "css/dropdown.css"

},

{

"studyVariationId": 1343334,

"studyVariableId": 1343337,

"variationName": "Slide Out",

"variationDesc": "Slide out menus",

"variationRoot": "wlux",

"variationCssPath": "css/slideout.css"

}

],

"studyMeasures": [

{

"studyMeasureId": 34930304,

"variableType": "participantCount",

"dataType": "integer",

"displayUnits": null,

"computeBy": "session",

"displayBy": "session"

},

{

"studyMeasureId": 34930305,

"variableType": "taskTime",

"dataType": "float",

"displayUnits": "milliseconds",

"computeBy": "task",

"displayBy": "task"

}

],

"studyVariables": [

{

"studyVariableId": 1343334,

"variableName": "cursor style"

},

{

"studyVariableId": 1343334,

"variableName": "menu style"

}

],

"studyPeriods": [

{

"studyPeriodId": 99334043,

"periodName": "Freshman year",

"periodDescription": "Studying when the freshman arrive at school",

"periodParticipantLimit": 1000,

"periodStartTime": 9992999203,

"periodEndTime": 9993999203,

"periodAnnouncementText": "Hello Freshmen! Start your year off by taking a survey!",

"periodEmailList": "\*.freshman@uw.edu",

"periodTaskSequence": "sequential",

"periodStudyVariations": "singleRandom",

"periodTimeZone": "UTC+08"

},

{

"studyPeriodId": 99334044,

"periodName": "Junior year",

"periodDescription": "Studying when the Juniors arrive at school",

"periodParticipantLimit": 1000,

"periodStartTime": 9992999203,

"periodEndTime": 9993999203,

"periodAnnouncementText": "Hello Junior! Start your year off by taking a survey!",

"periodEmailList": "\*.Junior@uw.edu",

"periodTaskSequence": "sequential",

"periodStudyVariations": "singleRandom",

"periodTimeZone": "UTC+08"

}

],

"steps" : [

TODO: add in step objects

]

}

}

##### ERRORS

* 404 - Study not found
* 401 - Not Authorized

### general

The general information about a study.

#### GET

Get the general information about a study.

##### REQUEST

* studyId (returns the general properties for the study)

##### RESPONSE

{

"data": {

"studyId": 9999,

"ownerId": 1234,

"studyName": "My favorite study",

"studyDescription": "A long description",

"researcherOrg": "org",

"researcherFirstName": "Alex",

"researcherLastName": "Researcher",

"researcherEmail": "aResearcher@example.com",

"defaultTimeZone": "UTC+08",

"defaultVariationMethod": "randomSequential",

"defaultTaskBarRoot": "wlux",

"defaultTaskBarCssPath": "css\\taskbar.css",

"defaultVariationRoot": "client",

"defaultVariationCssPath": "css\\vars\\var1.css",

"defaultExitButtonText": "Exit task",

"defaultTabShowText": "Show",

"defaultTabHideText": "Hide",

"clientTimeStampFormat": "unixtime",

"dateCreated": 9992999203,

"dateModified": 9992999203,

"studyStatus": "draft"

}

}

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### POST

Add the general information for a new a study.

##### REQUEST

* studyId (creates a new study)

##### RESPONSE

{

"data": {

"studyId": 9999,

"ownerId": 1234,

"studyName": "My favorite study",

"studyDescription": "A long description",

"researcherOrg": "org",

"researcherFirstName": "Alex",

"researcherLastName": "Researcher",

"researcherEmail": "aResearcher@example.com",

"defaultTimeZone": "UTC+08",

"defaultVariationMethod": "randomSequential",

"defaultTaskBarRoot": "wlux",

"defaultTaskBarCssPath": "css\\taskbar.css",

"defaultVariationRoot": "client",

"defaultVariationCssPath": "css\\vars\\var1.css",

"defaultExitButtonText": "Exit task",

"defaultTabShowText": "Show",

"defaultTabHideText": "Hide",

"clientTimeStampFormat": "unixtime",

"dateCreated": 9992999203,

"dateModified": 9992999203,

"studyStatus": "draft"

}

}

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### PUT

Update the general information about a study.

##### REQUEST

* studyId (updates 1 study object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### DELETE

Marks the study as deleted by flagging the study general record and all records in other tables that are associated with this study. NO RECORDS ARE DELETED FROM THE DATABASE!!!

##### REQUEST

* studyId (deletes the study)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

### measure

General information about a measure (dependent or measured variable) used in a study.

#### GET

Get a measure entry or all the measures used by a study.

##### REQUEST

* studyId (returns all measures for the study)
* measureId (returns 1 measure object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### POST

Add a new measure to a study.

##### REQUEST

* studyId (adds a measure to the study)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### PUT

Update a measure entry for a study

##### REQUEST

* measureId (updates 1 measure object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### DELETE

Delete a measure or all the measures in a study.

THIS COMMAND DELETES THE RECORD FROM THE DATABASE.

##### REQUEST

* studyId (deletes all measures for the study)
* measureId (deletes 1 measure object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

### name

Get information about a study by name

Protected data—Authorization required

#### GET

Returns information about the specified study. Used to simply check to see if a study with a specified name already exists.

Protected data—Read access required

##### REQUEST

{

"name": {

"ownerId": 1234,

"studyName": "My Favorite study"

}

}

##### RESPONSE

Returns the study ID if a study with the study name already exists

{

"data": {

"studyId": 9999,

"ownerId": 1234,

"studyName": "My Favorite study"

}

}

Or, if the study name has not been used by this account, the study ID value returned is 0.

{

"data": {

"studyId": 0,

"ownerId": 1234,

"studyName": "My Favorite study"

}

}

##### ERRORS

##### 401 - Not Authorized

### period

Information about a study period—the time during which participants can take the study.

FOR THIS VERSION, STUDY PERIODS CANNOT OVERLAP.

#### GET

Get the information about one study period or all study periods in a study.

##### REQUEST

* studyId (returns all periods in the study)
* periodId (returns the specified period)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### POST

Add a new study period to a study.

REQUEST

* studyId (creates a new period for the study)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
    - Overlaps an existing study period
* 401 – Not authorized
* 404 – Not found
  + - Study not found

#### PUT

Update the information about a study period.

##### REQUEST

* periodId (updates the specified period)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### DELETE

Delete a study period or all study periods for a study.

THIS COMMAND DELETES THE RECORD FROM THE DATABASE.

##### REQUEST

* studyId (deletes all periods in the study)
* periodId (deletes the specified period)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

### step

Information about a step in a study’s protocol.

#### GET

Get information about a single protocol step or all protocol steps in a study.

##### REQUEST

* studyId (returns all steps in the study)
* studyStepId (returns the specified study step)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### POST

Add a new protocol step to a study.

##### REQUEST

Note the request buffer needs a field to tell where in the protocol sequence this record goes. For V1, we’ll support only “after the last step” and consider “after step ‘n’” in a future version.

* studyId (creates a new study step)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### PUT

Update a protocol step record.

##### REQUEST

* studyStepId (updates the specified study step)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### DELETE

##### REQUEST

* studyId (deletes all study steps in the study)
* studyStepId (deletes the specified study step )

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

### variable

Information about an independent (manipulated) variable used in a study.

#### GET

Get information about a variable or all variables in a study.

##### REQUEST

* studyId (returns all variables for the study)
* variableId (returns 1 variable object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### POST

Add a new variable to a study.

##### REQUEST

* studyId (adds a variable to the study)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### PUT

Update the information about a variable in a study.

##### REQUEST

* variableId (updates 1 variable object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### DELETE

Delete a variable from a study or all variables in a study.

THIS COMMAND DELETES THE RECORD FROM THE DATABASE.

##### REQUEST

* studyId (deletes all variable for the study)
* variableId (deletes 1 variable object)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

### variation

Information about variations (levels) of the [independent] variables

#### GET

Get the information about one variation or all variations used in a study.

##### REQUEST

* studyId (returns all variations in the study)
* variationId (returns the specified variation)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### POST

Add a new variation to a study.

##### REQUEST

* studyId (creates a new variation)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### PUT

Update the information about one variation in a study.

##### REQUEST

* variationId (updates the specified variation)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found

#### DELETE

Delete one variation from a study.

THIS COMMAND DELETES THE RECORD FROM THE DATABASE.

##### REQUEST

* studyId (deletes all variations in the study)
* variationId (deletes the specified variation)

##### RESPONSE

##### ERRORS

* 400 – Bad request
  + - Parameter error
* 401 – Not authorized
* 404 – Not found