Using the REDCap API for Data Import and Export

Jack Baty

13 Feb 2019

Division of Biostatistics SDA Seminar Series

Disclaimer

I will take questions but I may not have answers.

The focus of the seminar is on export and import of records using resources available to Division of Biostatistics personnel.

Outline

- What is an API?
- Advantages
- Disadvantages
- What is needed
- API Playground
- Exporting from REDCap
- Getting the data into SAS
- Importing into REDCap
- (cron, sh)

API

- Application Program Interface
- Allows communication with REDCap and server without going through the interactive REDCap interface
- See REDCap API documentation from link on API token and API Playground pages.

Advantages

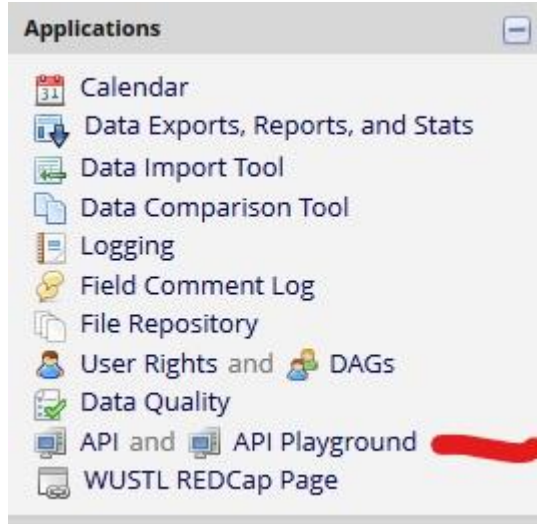
- Can customize exports and imports (variables, events, reports) and save as a program rather than doing hands-on every time
- Can export and import large data files that would choke interactive REDCap
- If exporting XML and reading with SAS XML engine, character-variable lengths will only be as long as they need to be, not $500 to store a $5 ZIP code.
- If exporting XML and reading with SAS XML engine, can add/subtract variables from export without modifying SAS code that reads file.
- Can schedule programs to run automatically using cron, the linux scheduling facility

Disadvantages

- Runs from linux (can run from Mac or Windows but I haven’t looked into it)
- Maybe not worth the trouble for small jobs or jobs that will only be run once
- Need a shell script if running from cron

What is needed

- API rights for the project (import and/or export). Set by the Project Administrator
- API token
- Access to a server


Applications
Calendar
Data Exports, Reports, and Stats
Data Import Tool
Data Comparison Tool
Logging
Field Comment Log
File Repository
User Rights and DAGs
Data Quality
API and API Playground
WUSTL REDCap Page

API

The REDCap API is an interface that allows external applications to connect to REDCap remotely, and is used for programmatically retrieving or modifying data or settings within REDCap, such as performing automated data imports/exports from a specified REDCap project. For details on the capabilities of the REDCap API and how to use it, please see the REDCap API documentation.

API Security: Best Practices
It is important to remember that when making requests to the REDCap API, you should always validate the REDCap server's SSL certificate to ensure the highest level of security during communication with the API. For details on what this means and how to do it, see the 'API Security: Best Practices' section in the REDCap API documentation.

Request API token for us nonsurg"

Use the button below to request an API key from your REDCap administrator. You will need a different token for each project you would like to access. Please note that your REDCap administrator is emailed every time a token is requested.

Request API token

API Playground- a very helpful feature

- Has dropdown lists for API options. Writes code as options are chosen
- Select the coding language for the API request: PHP, Perl, Python, Ruby, Java, R, cURL
- Code can be executed within the API Playground and the results will show in a response window
- Can copy and save code written in the Playground

API Playground- Exporting

- Specify output format: XML, CSV, JSON
- Specify formatted or raw values
- Specify some combination of records (IDs), fields, forms, events
- Can specify a filter using REDCap syntax: [age] > 60

API Playground

The API Playground is an interface that allows experimentation with the REDCap API without actually writing any code. You can explore all the different API methods and their various options to customize a given API request. You may even execute a real API request and see the exact response that REDCap returns from the request. If you are interested in creating an API script, the section at the bottom will provide code samples for various programming languages to give you a head start. For details on the capabilities of the REDCap API and how to use it, please see the REDCap API documentation.

Select an API method from the drop-down list below, after which it will load any other options that are specific to that method.

API Method: Export Records

Production - Some APIs have been disabled for the playground because they affect data

Format: XML

Type: flat

Records: -3342, -3343, -3344, -3345, -3346

If you don't make a selection you get everything

Fields: admdx, admitdt, age, anti_coag, anti_htn

Forms: barthel, demographics_and_contacts, discharge_info, priority_information

Events: event_1_arm_1, 336d5ebc54_arm_1

Raw Labels: raw

Raw Headers: raw

Checkbox Labels: false

Survey Fields: false

Data Access Groups: false

Filter Logic:

Errors: CSV

Raw Request Parameters

Displayed in the box below are all the POST parameters that would be sent in the API request based on the selections above.

```
content: record
format: xml
type: flat
rawOrLabel: raw
rawOrLabelHeaders: raw
exportCheckboxLabel: false
exportSurveyFields: false
```

Response

Click the Execute Request button to execute a real API request, and it will display the API response in a text box below.

Execute Request

PHP | Perl | Python | Ruby | Java | R | cURL

Displayed in the box below is the code you would use to execute this API request in the selected programming language.

```
#!/bin/sh
DATA="token=9539F0FFAC00A9B4AADE5013A530E4A1&content=record&format=xml&type=flat&rawOrLabel=raw&
rawOrLabelHeaders=raw&exportCheckboxLabel=false&exportSurveyFields=false&
exportDataAccessGroups=false&returnFormat=json"
CURL=`which curl`
$CURL -H "Content-Type: application/x-www-form-urlencoded" \
      -H "Accept: application/json" \
      -X POST \
      -d $DATA \
      https://redcap.wustl.edu/redcap/srvrs/prod_v3_1_0_001/redcap/api/
```

Download cURL Example Code

API Playground- Importing

- No import choices! (except Import Project Info)
- See API documentation

Exporting Examples: Kevan's critical SAS macro

```
%macro _transfr(_typ=0,_url=,_ct=,_in=,_out=,_hdrout=);
 proc http
    in=&_in
    out=&_out
    headerout=&_hdrout
    url="&_url"
    method="post"
    ct="&_ct“      ;
     run;
%mend _transfr;
```

Exporting Examples: Using the macro

```
filename ein  "&rawpath.ExSession.txt";
filename eout "&rawpath.ExSession.xml";
filename ehdrout "&rawpath.ExSessionHdr.txt";
* Build the in file.  We are exporting records (all of them) in XML format.  Exporting raw values and variable names;
data _null_;
                file ein;
                input;
                input _infile_;
datalines4;
token=***token goes here***&content=record&format=xml&type=flat&rawOrLabel=raw&rawOrLabelHeaders=raw
&exportCheckboxLabel=false&exportSurveyFields=false&exportDataAccessGroups=false &returnFormat=json
;;;;
run;
%_transfr(_typ=0,_url=%str(https://redcap.wustl.edu/redcap/srvrs/prod_v3_1_0_001/redcap/api/index.php),
   _ct=%str(application/x-www-form-urlencoded),_in=ein,_out=eout,_hdrout=ehdrout);
```

Exporting Examples: files used

- ExSession.txt contains the stuff between datalines4 and ;;;; , the arguments for the API call.
- ExSessionHdr.txt contains feedback from the API attempt: return code, timestamp, error message
- ExSession.xml contains the data exported from REDCap

Exporting Examples: What changes in the examples

```
filename ein  "&rawpath.ExSession.txt";
filename eout "&rawpath.ExSession.xml";
filename ehdrout "&rawpath.ExSessionHdr.txt";
* Build the in file.  We are exporting records (all of them) in XML format.  Exporting raw values and variable names;
data _null_;
                file ein;
                input;
                input _infile_;
datalines4;
token=***token goes here***&content=record&format=xml&type=flat&rawOrLabel=raw&rawOrLabelHeaders=raw
&exportCheckboxLabel=false&exportSurveyFields=false&exportDataAccessGroups=false &returnFormat=json
;;;;
run;
%_transfr(_typ=0,_url=%str(https://redcap.wustl.edu/redcap/srvrs/prod_v3_1_0_001/redcap/api/index.php),
   _ct=%str(application/x-www-form-urlencoded),_in=ein,_out=eout,_hdrout=ehdrout);
```

Exporting Examples: Exporting Specific Records (from Playground)

token=***TokenGoesHere***&content=record
&format=xml&type=flat&records[0]=6666&records[1]=7777
&records[2]=66901&rawOrLabel=raw
&rawOrLabelHeaders=raw&exportCheckboxLabel=false
&exportSurveyFields=false&exportDataAccessGroups=false
&returnFormat=json

Exporting Examples: Exporting Specific Records (from experience)

token=***TokenGoesHere***&content=record

&format=xml&type=flat&records=6666,7777,66901

&rawOrLabel=raw

&rawOrLabelHeaders=raw&exportCheckboxLabel=false

&exportSurveyFields=false&exportDataAccessGroups=false

&returnFormat=json

Exporting Examples: Exporting Specific Variables (from Playground)

token=***TokenGoesHere***&content=record&format=xml

&type=flat&fields[0]=adc_complete

&fields[1]=adc_dt&fields[2]=adc_sid&fields[3]=adc1&fields[4]=adc2

&rawOrLabel=raw&rawOrLabelHeaders=raw

&exportCheckboxLabel=false&exportSurveyFields=false

&exportDataAccessGroups=false&returnFormat=json

Exporting Examples: Exporting Specific Variables (from experience)

token=***TokenGoesHere***&content=record&format=xml

&type=flat&fields=adc_complete,adc_dt,adc_sid&,adc1,adc2

&rawOrLabel=raw&rawOrLabelHeaders=raw

&exportCheckboxLabel=false&exportSurveyFields=false

&exportDataAccessGroups=false&returnFormat=json

Exporting Examples: Exporting A Form

token=***TokenGoesHere***&content=record&format=xml
&type=flat&forms[0]=boavcl&rawOrLabel=raw
&rawOrLabelHeaders=raw&exportCheckboxLabel=false
&exportSurveyFields=false&exportDataAccessGroups=false
&returnFormat=json

Exporting Examples: Exporting An Event

token=***TokenGoesHere***&content=record&format=xml
&type=flat&events[0]=bl_arm_1&rawOrLabel=raw
&rawOrLabelHeaders=raw&exportCheckboxLabel=false
&exportSurveyFields=false&exportDataAccessGroups=false
&returnFormat=json

Exporting Examples: Exporting A Report

Create a report in interactive REDCap. Find the Report ID on the My Reports page.

token=***TokenGoesHere***&content=report&format=xml

&report_id=33369&rawOrLabel=raw&rawOrLabelHeaders=raw

&exportCheckboxLabel=false&returnFormat=json

Exporting Examples: Exporting CSV

- Just replace “xml” with “csv” in all the examples.
- It’s up to you to write the program to read the file.

Getting the data into SAS: XML

Use the SAS XML LIBNAME engine:

```
libname sas7256 xmlv2 "&rawpath.ExSession.xml";
 data recruit;
    set sas7256.item;
* Any added SAS code that you need or want;
run;
```

Getting the data into SAS: XML

- The export file does not contain FORMATS or LABELS.
- You can export by hand from REDCap, save the SAS code, and add it to your program.
- You don’t need INFORMAT or LENGTH statements. SAS infers those as it does when importing Excel files.

XML Gotcha

Variables that are numeric but have no data are interpreted as character. This can cause an error when assigning formats.

Work-around: don’t assign format to affected variables until they have data.

XML Gotcha

Invalid characters in unvalidated text variables can choke SAS XML engine, causing error. Usually comes from copying and pasting text from Word

Work-around: remove offending characters; don’t export unvalidated text variables; export unvalidated text variables separately as CSV

Write program to clean XML file before submitting to SAS XML engine?

Getting the data into SAS: CSV

```
* Just as you would any CSV file;
DATA CSV2SAS;
        INFILE “&rawpath.ExSession.csv” delimiter=‘,’ MISSOVER DSD
        LRECL=32767 FIRSTOBS=2;
        * your SAS code here: informats, length, input, formats;
        * You can export by hand from REDCap, save the SAS code, and
        use that;
        run;
```

Importing Example

- Fewer options
- XML advantages when exporting do not apply when importing
- Rules are similar to importing CSV in interactive REDCap
  - First variable must be ID
  - Include event name and data-access group if needed
  - All validation rules must be satisfied

Getting Data Into REDCap (CSV 1)

```
*Produce the CSV file in your preferred way;
/** create file handles */
filename ein "./smartIn.txt";  /* will contain parameters and data */
filename ehdrout "./smart_Hdr.txt"; /* will contain feedback.  Must exist before running import. Use touch command */
/** create parameter file  */
data _null_;
file ein;
 input;
put _infile_;
datalines4;
token=**Token Goes Here**&content=record&format=csv&type=flat&overwriteBehavior=normal&data=
;;;;
run;
```

Getting Data Into REDCap (CSV 2)

```
/** set the url variable */
%let _urlx=%str(https://redcap.wustl.edu/redcap/srvrs/prod_v3_1_0_001/redcap/api/index.php);
/** combine data and parameters file */
%sysexec cat ./data_up.csv >>./smartIn.txt;
data _null_;   x=sleep(2); run;
/** submit data to the server */
%sysexec curl -i -X POST -d @./smartIn.txt &_urlx   >> ./smart_Hdr.txt;
```

Importing Data Into REDCap (XML 1)

```
filename ein  "&rawpath.ExSessionIn.txt"; * will contain data and parameters ;
filename ehdrout “&rawpath.ExSession_Hdr.txt”; * will contain feedback;
/** create parameter file  */
data _null_;
  file ein;
  input;
  put _infile_;
  datalines4;   token=***TokenGoesHere***&content=record&format=xml&type=flat
&overwriteBehavior=normal&data=
;;;;
run;
```

Importing Data Into REDCap (XML 2)

```
/** set the url variable */
%let _urlx=%str(https://redcap.wustl.edu/redcap/srvrs/prod_v3_1_0_001/redcap/api/index.php);
/** combine data and parameters file */
%sysexec cat ./ExSessionIn.xml >>./ExSessionIn.txt;
data _null_;    x=sleep(2); run;
/** submit data to the server */
%sysexec curl -i -X POST -d @./ExSessionIn.txt &_urlx   >> ./ExSession_Hdr.txt;
```

cron

- cron is the linux job-scheduling utility
- Can schedule programs to run unattended on pre-determined schedule: hourly, daily, weekly, monthly, the first Wednesday of the month, at any time of the day
- https://www.geeksforgeeks.org/crontab-in-linux-with-examples/
- crontab file goes in your linux root directory:

    jack@saturn:/home/jack$

sh files: linux shell executable files

- File extension of .sh
- The Biostatistics computer system (cluster) precludes submitting SAS jobs directly from cron
- But you can run the SAS job directly from the linux prompt as you would any SAS job
- Use two .sh files when exporting/importing REDCap data using cron

https://www.javatpoint.com/steps-to-write-and-execute-a-shell-script

sh files: linux shell executable files. 1st file

```
#! /bin/sh
# runBRC_API.sh is file run by cron.  It calls a second file.
cd /home/crrg/api
qsub -q fast@saturn -l nodes=1:sas -l mem=120mb   ./api_main.sh
```

sh files: linux shell executable files. 2nd file

```
#! /bin/sh
# This file is, api_main.sh, called by the previous one, runBRC_API.sh
cd /home/crrg/api
/usr/local/bin/sas  ./BRCExport_csv.sas –noterminal
```

Questions?