Paper 153-2019

Using SAS to enhance data sharing across REDCap projects: Reducing errors, streamlining management, and improving quality

Rachel K. Myers, Children's Hospital of Philadelphia

ABSTRACT

With the increasing use of web-based data collection tools and systems, such as the widely used REDCap (Research Electronic Data Capture), there are opportunities to enhance our approach to data management, particularly related to data sharing across databases, projects, or research teams. In situations requiring data sharing across studies or the need to restrict data access to maintain blinding to treatment assignment, research teams often establish multiple project databases with different user permissions to share or restrict access to selected data elements. For on-going, prospective data sharing, teams commonly rely on manual export and import of data files, which can be burdensome, time intensive, require careful attention to file formatting, and be error prone. Enabling REDCap's API (Application Programming Interface) allows users to utilize SAS to securely share restricted or selected data across databases, eliminating the need for repeatedly constructing manual data exports, conducting data management, and modifying data import files. This paper describes SAS code for the direct export of selected data from a REDCap database into SAS for data management and manipulation and subsequent import of data into a destination REDCap database. This approach establishes a secure, replicable approach to data sharing that is particularly useful for prospective research and for projects requiring routine and on-going sharing of restricted data. Additionally, the use of the API parameters provides protection against overwriting existing data, reduces the risk of inadvertent disclosure of sensitive data, and provides a mechanism for maintaining records of all data sharing.

INTRODUCTION

Increasingly, research teams utilize electronic data capture to reduce reliance on time intensive manual data entry. REDCap (Research Electronic Data Capture) is a software toolset and workflow methodology for electronic collection and management of research data (Harris et al., 2009; Vanderbilt University, 2019). It is a secure, web-based application that permits use for various types of research, including longitudinal designs with repeated measures, survey-based data collection, and manual data entry. REDCap provides easy data manipulation (with audit trails for reporting and monitoring) and built-in functionality to permit manual export of data. A key advantage of REDCap for research teams is that most of its features can be accessed through graphical user interface (GUI) requiring minimal programming or data management experience or knowledge.

However, more advanced REDCap functions are available to you by accessing the Application Programming Interface (API), which permits scripted access to data, enhancing reproducibility, streamlining data management, and facilitating rapid data access without having to access web-based data queries. This paper outlines SAS code that uses the REDCap API functionality to streamline data export, management, and sharing of selected records and data elements across multiple REDCap databases. In our setting, we have created API parameters to permit data sharing not just within a single project, but also across projects, providing a secure way to manage data access. This is particularly useful for research teams who may need to share or restrict access to certain data or to share data while maintaining restrictions to project databases to protect research participant confidentiality and maintain data integrity. In this paper, I describe the code we developed to support data export, management, and import, with a particular focus on maintaining a clear audit trail of data preparation and migration.

ADVANTAGES OF SAS API FOR REDCAP DATA EXPORT AND IMPORT

A key motivation for using REDCap is that it is widely accessible and databases and survey forms can be created by users with no or limited programming experience. From a data management perspective, GUI menus permit users to export and import data files. However, these manual processes require extensive

menu-driven manipulation and a great deal of attention to the format of data files, including selected cases and variables. Enabling API functionality permits you to overcome these limitations and complete export and import requests with SAS code that exports data in a consistent and reproducible format. These data can be easily imported into SAS for additional data management prior to direct import into a destination REDCap database. Use of the API functionality allows you to control the inclusion of selected data elements, as well as variable names and formats, without extensive manual manipulation of data files.

Our research team has used the API functionality to streamline data collection and data sharing efforts for two prospective research studies that are recruiting identical participant populations. By enabling the API functionality within REDCap, we reduce participants' response burden so that data is collected one time and relevant data elements are shared with the respective research teams. Further, we have created API parameters to facilitate data sharing of both research and clinical data across project teams to enhance data quality, eliminate the need for time-intensive and error prone manual data entry, and maintain blinding of research staff to treatment condition. I detail the export and import API parameters and specific customizations to provide access to the SAS code and data files for users with no or limited SAS proficiency.

Figure 1 provides an overview of the approach to data export, management, and import. In our current setting, prospective research data is collected from participants via a self-administered electronic survey in REDCap (Figure 1a). SAS allows you to create a file containing the API export parameter (Figure 1b) that is sent securely to REDCap using the HTTP procedure and requested data elements are securely exported and saved locally for further data management in SAS (Figure 1c). Exported data can be further manipulated to create an API import parameter (Figure 1d) containing selected data that are sent to a destination REDCap database (Figure 1e) via PROC HTTP.

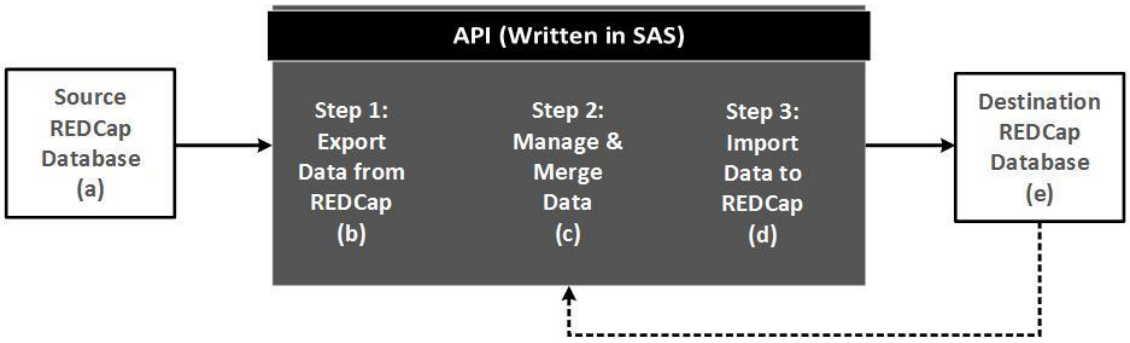


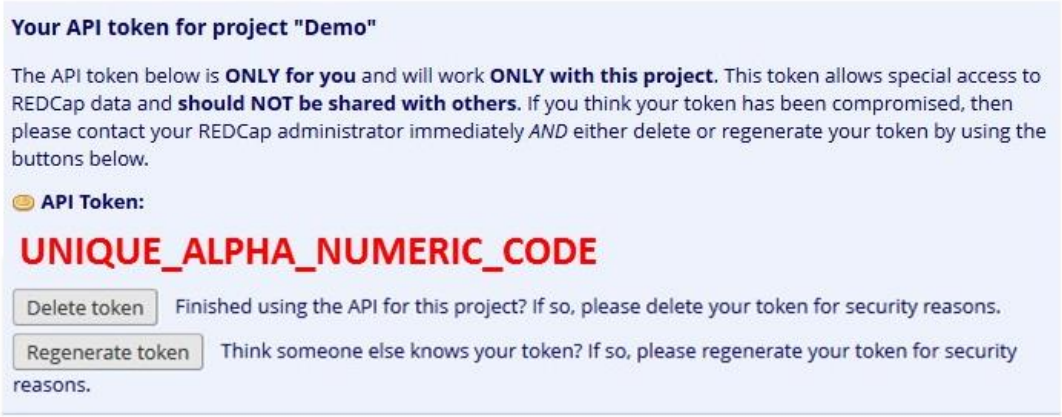
Figure 1. Overview of API process.

The API parameters written in SAS provide you with options to control the export and import of selected data elements to further streamline data management and reduce errors during data import. A key advantage of using the API functionality is that it permits multiple users to request identical data exports and to export the most current data. These export, management, and import steps can be combined into a single SAS syntax file to provide an efficient way to manage data migration. Using nested API parameters, as further described below, you can identify new records or merge data from multiple databases into a single import file that are imported into the destination REDCap database. In the following sections, I discuss the use of API parameters to manage data exports and imports and provide example SAS code to create and execute these data requests.

STEP 1: CREATE API PARAMETER TO REQUEST REDCAP DATA EXPORT

This API builds on work previously detailed by Worley and Yang (Worley & Yang, 2013). The following code makes extensive use of SAS Macro functions to streamline the creation of the API parameter files. For our projects, data files are exported and imported daily. To ensure an audit trail of all exports and imports, this code is customized to provide date stamps on all created files and to maintain files in a dated file directory. However, this step is optional if you do not require an auditable trail of data exports.

Within REDCap, you can request and obtain an API token that identifies you as a unique user of a specific database (Display 1). This token permits you to access REDCap data without manually logging into your REDCap database through the GUI. Tokens are specific to a database, so you require a unique API token for each database from which you will export (e.g. source database) or import data (e.g. destination database). You should protect this token as you would any password, as the API token provides any user with access to the token permission to access your REDCap database. If necessary, tokens can be deleted and recreated to protect your database and data.



Display 1. API Token Request Screen in REDCap.

The following code creates a date stamped directory in which all exported files are saved, as well as creates and sends the API export parameter to a source REDCap database. This code generates a .CSV data file containing selected records and variables:

```
***Create date stamped library to store export files;
   OPTIONS dlcreatedir;
   LIBNAME data "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )";

***Create .TXT file to store API export parameter to be sent to REDCap;
   FILENAME expapi
      "C:\FileLocation_%sysfunc( today(),
      MMDDYYD10 )\export_apiparameter.txt";

***Create .CSV file to store exported data;
   FILENAME exportdata
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      export_data_%sysfunc( today(), MMDDYYD10 ).csv";

***Create output file to document success of PROC HTTP request;
   FILENAME status
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\export_status.txt";

***Unique API token obtained from the REDCap project containing data to be
exported- references specific database and user;
***This code is generated within the source REDCap database;
   %LET exptoken = UNIQUE_ALPHA_NUMERIC_CODE_EXPORT_DATABASE;

***Define list of variables to be exported;
***This step is optional if all variables within the source database are
desired;
   %LET export_vars=record_id, var1, var2, var3,...,varn;
```

```
***Define any filter logic parameter terms;
***This step is optional if all records within the source database are
desired;
   %LET ftlog=[var1]='value' and [var2]>'value';

/*************************************************************************
Create API parameter file that will be sent to REDCap.

In this DATA step, _null_ specifies that no SAS data file will be created,
but rather the inputs will be written to the API export parameter text file
(expapi). This text file will be sent to REDCap by PROC HTTP.

This code references the previously defined macro variables (exptoken,
export_vars, and ftlog).
*************************************************************************/
   DATA _null_ ;
      FILE expapi;
         PUT
         "%NRStr(token=)&exptoken%NRStr(&content=record&type=flat&format=csv)
         %NRStr(&filterLogic=)&ftlog%NRStr(&fields=)&export_vars";
   RUN;
```

The preceding DATA step creates a text file that specifies: (1) the REDCap database from which you are exporting data (token=); (2) that study records will be exported (content=); (3) that one record will be exported per row (type=); (4) the export file will be saved as .CSV file (format=); (5) inclusion criteria to limit data to selected records (filterLogic=); and (6) specifies requested variables (fields=).

Several of these parameter terms are optional, but provide you with control over the data elements included in the export file. Additional options are available to limit the export to variables from specific forms or study events. Table 1 summarizes additional parameter terms that can be included in your API export parameter to further define and customize the export request. The API parameter is case sensitive, so ensure that terms are appropriately capitalized. An example of the created API export parameter text file appears in Output 1.

API Parameter Term	Optional or Required Term	Function
&content	Required	Defines that type of data requested in export. “Record” will generate an export file containing study records.
&events	Optional	If REDCap project contains longitudinal events, defines the event from which data will be exported. Multiple events may be specified.
&fields	Optional	Defines variables for inclusion in export. May include as few or as many variables as desired.
&filterLogic	Optional	Defines inclusion criteria for exported data. May contain multiple filter criteria.
&format	Required	Defines export output format
&forms	Optional	Defines database forms from which data will be exported
&records	Optional	Defines the record IDs for data to be exported
&type	Required	Defines the format of the output file. “Flat” will export a single record per row and is the default.

Table 1. Summary of Available API Parameters.

```
token=UNIQUE_ALPHA_NUMERIC_CODE_EXPORT_DATABASE&content=record&type=flat&fo
rmat=csv&filterLogic=[var1]='value' and [var2]>'value'&fields=record_id,
var1,var2,var3,...,varn
```

Output 1. Example text file containing API export parameter.

The following code completes the final step of the export request. Using PROC HTTP, the API export parameter text file is securely sent to REDCap and the requested data export file is created:

```
/*************************************************************************
Use HTTP procedure to send API export parameter file to REDCap.

This procedure requires a site specific URL (https://redcap.url/api) that
corresponds to your local REDCap database. Running the PROC HTTP also
requires an internet connection.
*************************************************************************/
   PROC HTTP
      in= expapi
      out= output
      headerout = status
      url = "https://redcap.url/api/"
      method="post";
   RUN;
```

If the API parameter is successfully sent to and received by REDCap, your .CSV file (export_data.csv) will contain the requested records and variables. The status .TXT file will contain a message similar to Output 2. “200 OK” indicates a successful API request.

```
HTTP/1.1 200 OK
Date: Fri, 12 Jul 2019 17:23:36 GMT
Server: Apache
X-Powered-By: PHP/7.2.10
```

Output 2. Output from successful PROC HTTP API export request.

STEP 2: IMPORT DATA INTO SAS FOR DATA MANAGEMENT

A key advantage of REDCap is that it provides you with syntax files to streamline the import of data into your preferred software program. After you have obtained and verified the .CSV export file, you can modify the SAS code generated by REDCap to import your data file into SAS and complete any necessary data management or manipulation, such as comparing data files or renaming variables prior to import. In our project, we have utilized nested API export parameters to facilitate the comparison of our newly exported data file to existing data in our destination REDCap database. This allows the creation of an import data file containing only new, unique records.

The following code provides an edited example of importing the .CSV file into SAS, the nested API export parameter to request data from the destination database, data comparison, and preparation of a .CSV file containing data to be imported into REDCap:

```
/*************************************************************************
The SAS code generated by REDCap can be modified to reflect your customized
list of exported variables and export file names. A truncated example of this
code is provided below.

In the following code:
“DataForImport” refers to the SAS file to be manipulated to create an import
data file.
```

```
In the INFILE statement, "ExportData" refers to the .CSV file created by the
previous API export parameter request. As this file contains column headers,
firstobs is set to 2.

It is generally unnecessary to import and assign value formats to your SAS
data file, as these will not be needed during the API Import (Step 3, below)
*********************************************************************/
   %macro removeOldFile(bye); %if %sysfunc(exist(&bye.)) %then %do;
   PROC DELETE data=&bye.; run;
      %end; %mend removeOldFile;
      %removeOldFile(work.DataForImport);
      data DataForImport; %let _EFIERR_ = 0;
   INFILE ExportData delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2 ;
      informat record_id best32.
      informat var1 format. ;
      informat var2 format. ;
      informat var3 format. ;

      <lines omitted>

      informat varn format. ;

      format record_id best32.;
      format var1 format. ;
      format var2 format. ;
      format var3 format. ;

      <lines omitted>

      format varn format.;

   input
      record_id
      var1
      var2
      var3

      <lines omitted>

      varn
   ;
   if _ERROR_ then call symput('_EFIERR_',"1"); run; quit;

   PROC CONTENTS data=DataForImport varnum; run;
```

In the preceding code, INFILE should refer to the .CSV export file created by the API export parameter request (Step 1). The CONTENTS procedure allows you to visually inspect the newly created SAS data file to ensure that the anticipated variables are present in your SAS data file.

To create a data file containing only unique records not currently present in the destination REDCap database, you can create a second API export parameter. This parameter requests data from the destination REDCap database (Figure 1e) and creates a data file that can be compared to your source data to identify new records. This step is optional if you do not need to compare existing data or import only unique records.

This request will be similar to that described above (Step 1, above), with the exception that the API token will direct the parameter to be sent to the destination REDCap database:

```
***Create .TXT file to store API export parameter to be sent to destination
REDCap database;
   FILENAME compapi
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      comparisondata_apiparameter.txt";

***Create .CSV file to store exported data elements;
   FILENAME compdata
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      existing_data_%sysfunc( today(), MMDDYYD10 ).csv";

***Create output file to document success of PROC HTTP request;
***HTTP Code 200 indicates successful export;
   FILENAME compstat
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      existingexport_status.txt";

***Unique API token obtained from the REDCap project containing data to be
exported- references specific database and user;
***This code is generated within the export REDCap database;
   %LET desttoken = UNIQUE_ALPHA_NUMERIC_CODE_DESTINATION_DATABASE;

/*************************************************************************
Define list of variables to be exported;

This step is optional if you require export of all variables from the
destination database. Typically you may only require a few variables, such as
the unique ID or record data, which will be compared across files. In this
example, record_id is a unique identifier that is unchanged across databases
and used to identify new records;
*************************************************************************/
   %LET comp_vars=record_id, var1, var2, var3,...,varn;

***Create API parameter file that will be sent to REDCap;
   DATA _null_ ;
      FILE compapi ;
         PUT
         "%NRStr(token=)&desttoken%NRStr(&content=record&type=flat&format=csv
         )%NRStr(&fields=)&comp_vars";
   RUN;


***Use HTTP procedure to send API parameter file to destination REDCap
database and request data export;
   PROC HTTP
      in= compapi
      out= compdata
      headerout = compstat
      url = "https://redcap.url/api/"
      method="post";
   RUN;
```

After verifying that the data export was successful, you can use the SAS code generated by REDCap to import the data to be used for comparison into SAS. This is similar to the process outlined above, but should be adapted for import of the newly created .CSV file (“existing_data.csv”).

You could also use a similar process to export data from a second database and then merge variables from two source databases for import into the destination REDCap database. This process provide flexibility to meet your specific data requirements.

To streamline data management, I recommend using a consistent unique record ID across all related project databases and avoiding renaming records. This facilitates easier data management when identifying new records for data import. The following code details the process for identifying new records for import into the destination REDCap database:

```
/*************************************************************************
"CompData" refers to the SAS file exported from the destination REDCap
database.

Ensure that both SAS data files (DataForImport and CompData) are sorted by
the same variable. Data within these files are compared and duplicates
removed to create a new file containing only unique records for import.
*************************************************************************/
   PROC SORT data=DataForImport; by record_id; run;

   PROC SORT data=CompData; by record_id; run;

***Create SAS data file that contains only unique cases to be imported into
destination REDCap database;
   DATA import_data;
      merge DataForImport(in=a) CompData(in=b);
      by record_id;
      if a and not b;
   RUN;

   PROC SORT data=import_data;
      by record_id;
   RUN;

   PROC CONTENTS data=import_data varnum; run;

/*************************************************************************
Use a DATA step to retain variables for the data import file, to set values
for any fixed variables, or to modify any variable names to match variable
names in the destination REDCap database.
*************************************************************************/
   DATA import_data_final;
      RETAIN record_id var1 var2 var3 ... varn;
      SET import_data (DROP= var4);
      var1 = "fixed value";
   RUN;

****Create .CSV file containing data that will be imported into destination
REDCap via API import parameter;
   DATA data.importdata; SET inport_data_final; RUN;

   PROC EXPORT data= data.importdata
      OUTFILE= " C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      ImportData_%sysfunc( today(), MMDDYYD10 ).csv"
      DBMS=CSV REPLACE; PUTNAMES=YES;
   RUN;
```

STEP 3: CREATE API PARAMETER FOR REDCAP DATA IMPORT

After creating a data file with the selected records and variables, the final step is to create an API import parameter that will be sent to REDCap via PROC HTTP. These steps are similar to those outlined for the data export request (Step 1), but include a step in which you append data to the API import parameter file.

To successfully complete the import request, you must have API import/update privileges assigned to your user credentials in the destination REDCap database. This option is available under the User Rights menu within the destination REDCap database. If your API token does not permit import, you will be unable to successfully import the data file. Additionally, you must disable the "Auto-numbering for records" option within your REDCap project, otherwise the import will fail to create the new record IDs. The following code details the creation and transfer of the API import parameter to the destination REDCap database:

```
***Create .TXT file to store API import parameter to be sent to REDCap;
   FILENAME impapi
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      import_apiparameter.txt";

***Create .CSV file to store record IDs of imported data, if successful;
   FILENAME impout
      "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      import_ids_%sysfunc( today(), MMDDYYD10 ).csv";

***Create output file to document success of PROC HTTP request;
***HTTP Code 200 indicates successful export;
   FILENAME impstat
      "C:\File Location_%sysfunc( today(), MMDDYYD10 )\import_result.txt";

/*************************************************************************
Create API import parameter file that will be sent to REDCap.

This code references the previously defined macro variables (desttoken),
which was defined in Step 2. If you did not export data from the destination
database, you must define this API token during this step.
*************************************************************************/
   DATA _null_ ;
      FILE impapi ;
      PUT "%NRStr(token=)&desttoken%NRStr(&content=record&type=flat&
      format=csv&returnformat=csv&returnContent=ids&overwriteBehavior=normal
      &dateFormat=MDY&data=)";
   RUN;
```

The preceding code specifies the following parameter terms: (1) the REDCap database to which you are importing data (token=); (2) that study records will be imported (content=); (3) that one record will be imported per row (type=); (4) the export file will be saved as .CSV file (returnformat=); (5) the return content will be record IDs (returnContent); (6) that blank values will be ignored (overwriteBehavior=); and (7) the appropriate format of dates for the destination database(dateFormat=). The data parameter is left blank until the following step, where data from the .CSV files is read into the API import parameter.

```
/*************************************************************************
Insert.CSV data file to API import parameter .TXT file.

This DATA step appends the data contained in the .CSV file (ImportData) to
the data parameter term in the API import parameter file (impapi).
```

```
"MOD" directs SAS to append data to the existing file, instead of overwriting
it.

The logical record length (lrecl) permits records of up to 1GB (1,073,741,823
bytes). This is modifiable based on your unique needs and the size and length
on your import records.

The INFILE statement identifies the file to be read into the parameter file,
while the INPUT statement reads the data into the API parameter file, and the
PUT statement writes these values into the API parameter file.
*********************************************************************/
   DATA _null_;
      FILE impapi MOD lrecl=1073741823;
      INFILE "C:\FileLocation_%sysfunc( today(), MMDDYYD10 )\
      ImportData_%sysfunc( today(), MMDDYYD10 ).csv" ls=32767;
      INPUT x : $32767.;
      PUT _infile_;
   RUN;
```

This DATA step creates the API import parameter that contains both the necessary parameter terms and the data to be sent to REDCap. Output 3 contains an example of the API import parameter file that is created by this DATA step.

```
token=UNIQUE_ALPHA_NUMERIC_CODE_IMPORT_DATABASE&content=record&type=flat&fo
rmat=csv&returnFormat=csv&returnContent=ids&overwriteBehavior=normal&dateFo
rma=MDY&data=record_id,var1,var2,var3,...,varn
101,John,Smith,10,...,10-11-2018
102,Jane,Doe,47,...,12-01-2016
103,Frank,Joseph,33,...,05-31-2015
104,Susan,Wright,51,...,09-15-2017
```

Output 3. Example text file containing API import parameter

```
/*********************************************************************
Use HTTP procedure to send API import parameter file to destination REDCap
database.

This procedure requires a site specific URL (https://redcap.url/api) that
corresponds to your local REDCap database. Running the PROC HTTP also
requires an internet connection.
*********************************************************************/
   PROC HTTP
      url ="https://redcap.url/api/"
      method="post"
      in= impapi
      out= impout
      headerout= impstat
      ;
   RUN;
```

If the API parameter is successfully sent to and received by REDCap, the .CSV file will contain a list of the imported record numbers and the status .TXT file will contain output similar to Output 4. The destination REDCap database will contain the imported records and variables.

```
HTTP/1.1 200 OK
Date: Fri, 12 Jul 2019 19:21:15 GMT
Server: Apache
X-Powered-By: PHP/7.2.10
```

Output 4. Output from successful PROC HTTP API import request.

MAKING API FILES ACCESSIBLE

One of the primary advantages of using the API parameters is that all of your data export and import requests are scripted to avoid unintentional modifications in the requested variables. You are also able to create export data files that contain a limited or restricted set of variables or records. Further, project staff with limited data management or SAS experience are able to routinely run the SAS code while maintaining the integrity of the source code and avoiding introducing errors in the export or import process.

You can save each step described above into separate syntax files to ease debugging and modifications over time. To users to conduct data exports and imports, use the %INCLUDE statement to create a single SAS syntax file. This file directs SAS to execute code in the individual syntax files, but makes the files and code invisible to users, who only have access to the combined syntax file. Staff with permission to access this combined syntax file are able to run SAS code without requiring a deep understand of or ability to manipulate the source code. Users can complete the process of exporting and importing data with a single run command.

CONCLUSION

Enabling the API functionality of REDCap permits you a great degree of control over data exports and imports. Using PROC HTTP facilitates the export and import directly from SAS, allowing you to easily manipulate and analyze your data. Key advantages of the REDCap API functionality include reducing the time to conduct data exports and imports and significantly reducing the potential for errors that are common with manual data entry or manipulation. Given the highly customizable features of the API parameters, such as restricting exports to specific records, variables, or events, you can customize exports and imports for multiple data needs. The API process eliminates the need to access the REDCap GUI and provides an easily reproducible way to create similar data exports on an on-going basis. Further, use of API functionality permits streamlined and secure data sharing both within projects and across research teams, reducing participant burden and importantly, the risk for inadvertent disclosure or sharing of confidential data.

REFERENCES

Harris, P. A., Taylor, R., Thielke, R., Payne, J., Gonzalez, N., & Conde, J. G. (2009). Research electronic data capture (REDCap) – A metadata-driven methodology and workflow process for providing translational research informatics support. *J Biomed Inform, 42*(2), 377-381.

Vanderbilt University. (2019). Research Electronic Data Capture (REDCap). Retrieved July 18, 2019, from https://www.project-redcap.org/

Worley, S., & Yang, D. (2013). "SAS and REDCap API: Efficient and Reproducible Data Import and Export." *Proceedings of the Midwest SAS Users Group 2013 Conference*, Columbus, OH. Available at: https://www.lexjansen.com/mwsug/2013/RX/MWSUG-2013-RX02.pdf

ACKNOWLEDGMENTS

I'd like to thank Alexis Brzuchalski for her thoughtful review of this paper and Melissa Pfeiffer and Kristi Metzger for their early feedback on this work. I am also deeply grateful to Leah Brogan, Valerie Everett, Katherine Feske-Kirby and Stephanie Garcia who have tested code and shared their user experiences.

The content reported in this publication was supported by the Eunice Kennedy Shriver National Institute Of Child Health & Human Development of the National Institutes of Health under Award Number

R01HD087406. The content is solely the responsibility of the author and does not necessarily represent the official views of the National Institutes of Health.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Rachel Myers
Center for Injury Research and Prevention
Children's Hospital of Philadelphia
myersr@email.chop.edu

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.