Literate Programming Patterns and Practices with REDCap and REDCapR

Will Beasley, Thomas Wilson, David Bard
OUHSC Pediatrics,
Biomedical & Behavioral Methodology Core (BBMC)

Statistical Computing User Group (SCUG)
Sept 2, 2014

Differences in the past 1.5 years

Compared to the March 2013 presentation…

- There’s less coverage of REDCap’s strengths & weaknesses.
- No coverage of lower-level strategies, such as direct API calls or raw CSV export/imports.
- More specifics about reporting.
- More guidance on security strategies.

Continuous Quality Improvement (CQI)

- Defined by HRSA as “a continuous process that employs rapid cycles of improvement”

- We provide a detailed, yet generalizable, illustration of CQI benefiting from REDCap and literate programming, which hopefully will increase the
  - speed of development,
  - consistency of implementation,
  - adherence to recommended security practices,
  - complexity of statistical analyses,
  - breadth of audience, and
  - frequency of informative CQI cycles.

Literate programming

- Literate programming tools can combine statistical text, tables, and graphs in a coherent document that is accessible to unfamiliar audiences.
- The automation of these tools eliminates the need to repeatedly copy and paste analytic results after underlying data sources are updated.

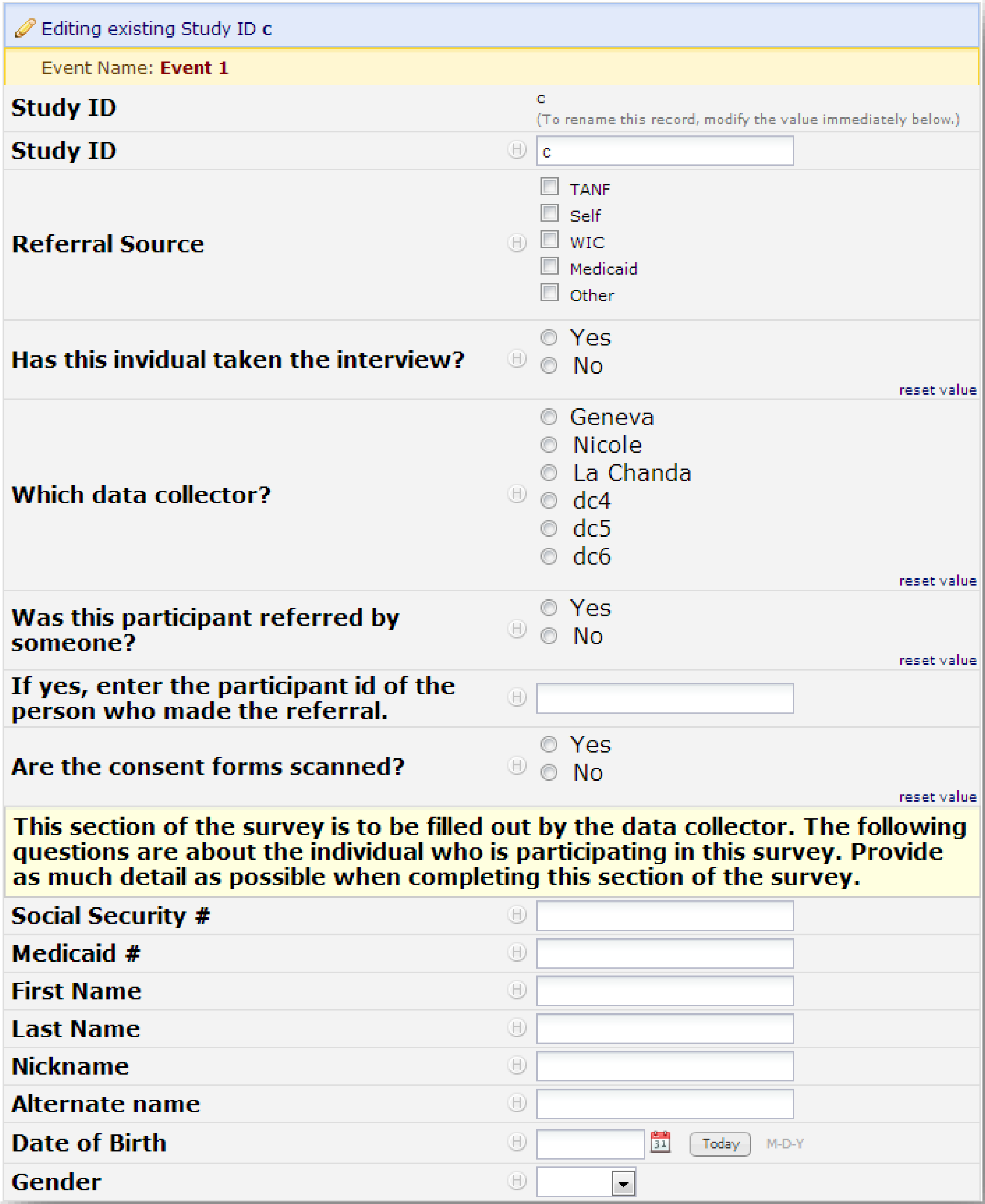
Collaboration among

1. The 4 statisticians on the project.
   sharing software development.
2. The 20 people on the project.
   exchanging participant-level data.
3. The 3 partnering organizations. (OSDH, WIC, OHCA)
   -receiving their subject-level & agency-level data.
   -distributing our results –fresh & frequently.
4. Academics in different areas. (particularly at OUHSC)
   exchanging tools and workflows.
5. Researchers in other states pursuing similar goals.
   publishing ideas and replicating previous work.

REDCap overview (http://project-redcap.org/)

- Secure web application for building and managing surveys and databases.
  - Developed by informatics core at Vanderbilt with support from NCRR and NIH, and designed for academic biomedical researchers.
- Provides:
  - Auditing capabilities, and other HIPAA-friendly features by default.
  - A centralized, back-end storage component.
  - Tools to create an interactive front-end html GUI.
  - An API to import & export data.
  - Example templates.
  - Instructional videos for training.
  - User-group network of institutional researchers.
  - Also included: built-in project calendar, scheduling module, ad hoc reporting tools, and advanced features, such as branching logic, file uploading, and calculated fields.
- It can reduce
  - Developing a lot of new software applications.
  - Anxieties related to security of home-grown software.

Example data entry



Editing existing Study ID c

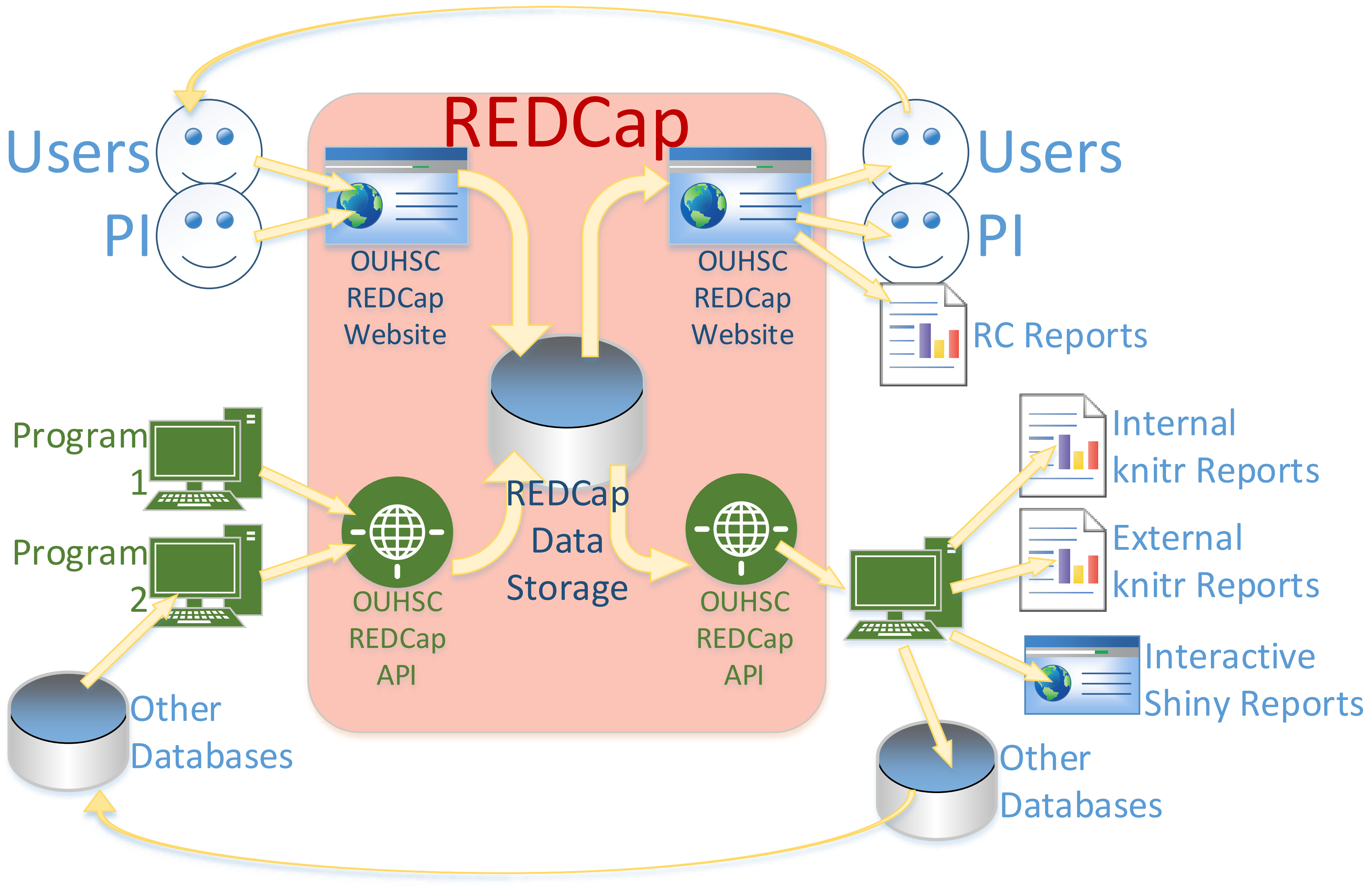
Event Name: **Event 1**

Study ID	c (To rename this record, modify the value immediately below.)
Study ID	c
Referral Source	☐ TANF ☐ Self ☐ WIC ☐ Medicaid ☐ Other
Has this invidual taken the interview?	○ Yes ○ No
Which data collector?	○ Geneva ○ Nicole ○ La Chanda ○ dc4 ○ dc5 ○ dc6
Was this participant referred by someone?	○ Yes ○ No
If yes, enter the participant id of the person who made the referral.	
Are the consent forms scanned?	○ Yes ○ No

This section of the survey is to be filled out by the data collector. The following questions are about the individual who is participating in this survey. Provide as much detail as possible when completing this section of the survey.

Social Security #	
Medicaid #	
First Name	
Last Name	
Nickname	
Alternate name	
Date of Birth	Today M-D-Y
Gender	

Possible REDCap Workflows



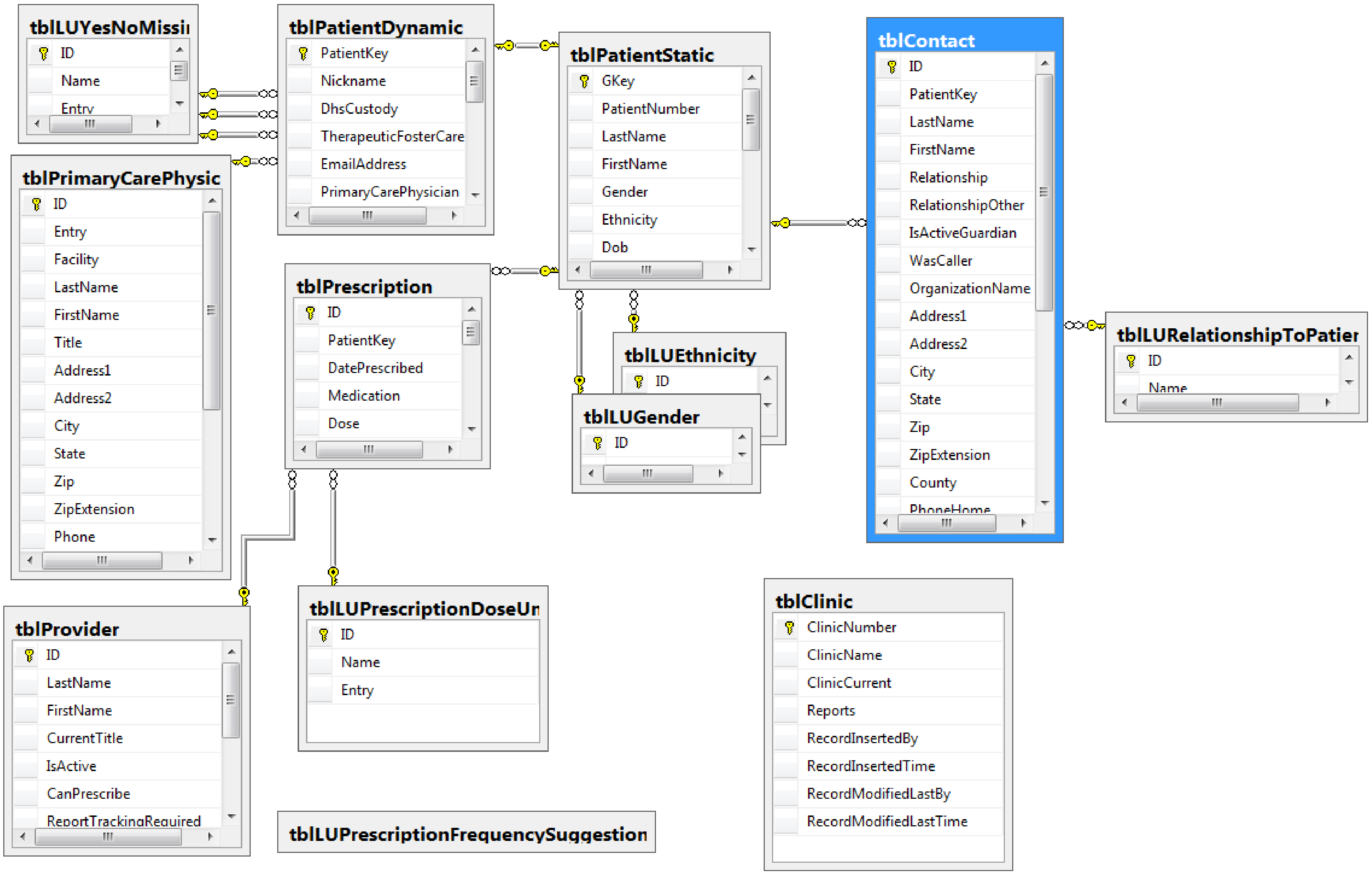
Scenarios favoring REDCap

- Project requires a centralized data store, but multiple locations for data entry.
  - Avoid syncing different locations manually.

- You want a flexible, universal framework to create consistent data systems for multiple clinical projects (research and possibly operations).
  - Reduces your development time & your staff's training time.
  - Reduces writing new text for grant proposals and IRB?

- Project has a relatively flat data structure.
  – Typically accommodates 2 or 3 levels well, but is clumsy beyond that. County, Practice, Provider, Patient, Time, Family.

Scenario NOT favoring REDCap



Scenarios favoring REDCap

- There are lots of dimensions and trade-offs when designing clinical research, and REDCap is close to the sweet spot for most designs.

- Access, SQL Server, Survey Monkey, Qualtrics, Concerto, E-prime, Excel, EMR.

Accessing REDCap from R

1. Manual Import & Export (eg, through CSV files)
   – Require human interaction every time.

2. R calls REDCap's API (application programming interface)
   – An API allows nonhumans to interact with each other directly.

3. REDCapR call REDCap's API (an R library)
   – Provides functions that wrap around calls to API.
   – Write 1 line of R code instead of ~40 lines.

Required pieces of information for API

1. The URL of the REDCap server.

2. A “token”, which is a hash that combines:
   - The specific REDCap project (within the REDCap server).
   - The specific user.
   - The user’s password.

Security

- Could spend 4 hours discussing security details.
  - Consult REDCap IT staff and/or our team.
- Use a *private* GitHub repository. (free for academics)
- Be careful with REDCap tokens. (ie, passwords)
- Get PHI into REDCap & SQL as early as possible.
  - We regularly receive CSVs & XLSXs from partners.
  - DB files aren’t accidentally copied or emailed.
  - And try to store derivative datasets in REDCap & SQL instead of on the file server.

Underlying Security Concepts Part 1

- Principle of least privilege: expose as little as possible.
  - Limit the number of team members.
  - Limit the amount of data (consider rows & columns).
  - Obfuscate values and remove unnecessary PHI in derivative datasets.
- Redundant layers of protection.
  - A single point of failure shouldn’t be enough to breach PHI security.

Underlying Security Concepts Part 2

- Simplify when possible.
  - Store data in only two houses. (REDCap & SQL Server)
  - Easier to identify & manage than a bunch of PHI CSVs scattered across a dozen folders, with versions.
    - Manipulate your data programmatically, not manually.
  - Windows AD account controls everything, indirectly or directly. (VPN, Odyssey, file server, SQL Server, & REDCap)

- Lock out team members where possible.

  It's not that you don't trust them with a lot of unnecessary data, it's that you don't trust their ex-boyfriend and their coffee shop hacker.

Installing REDCapR

```r
### Read short intro at
#   https://github.com/OuhscBbmc/REDCapR

### Install the 'devtools' package from CRAN
install.packages("devtools")



### Install the 'REDCapR' package from GitHub
devtools::install_github(repo="OuhscBbmc/REDCapR")

### Load the 'REDCapR' package into R's memory
#   so the functions are more easily accessible.
require(REDCapR)
```

Minimal Code without REDCapR

```r
### This is if you DO NOT use REDCapR, or some similar package.

### Query REDCap API
rawCsvText <- RCurl::postForm(
  uri="https://bbmc.ouhsc.edu/redcap/API/",
  token="39ZZZZZZZZZ377009A4434F3D86E363", #Be careful w/ real tokens.
  content='record',
  format='csv',
  type='flat',
  .opts=curlOptions(ssl.verifypeer=FALSE)
)

### Convert raw text into a data.frame
head(rawCsvText) #Inspect the raw data, if desired.
ds <- read.csv(text=rawCsvText, stringsAsFactors=FALSE)
head(ds) #Inspect the new data.frame, if desired.
```

Exporting records (less secure)

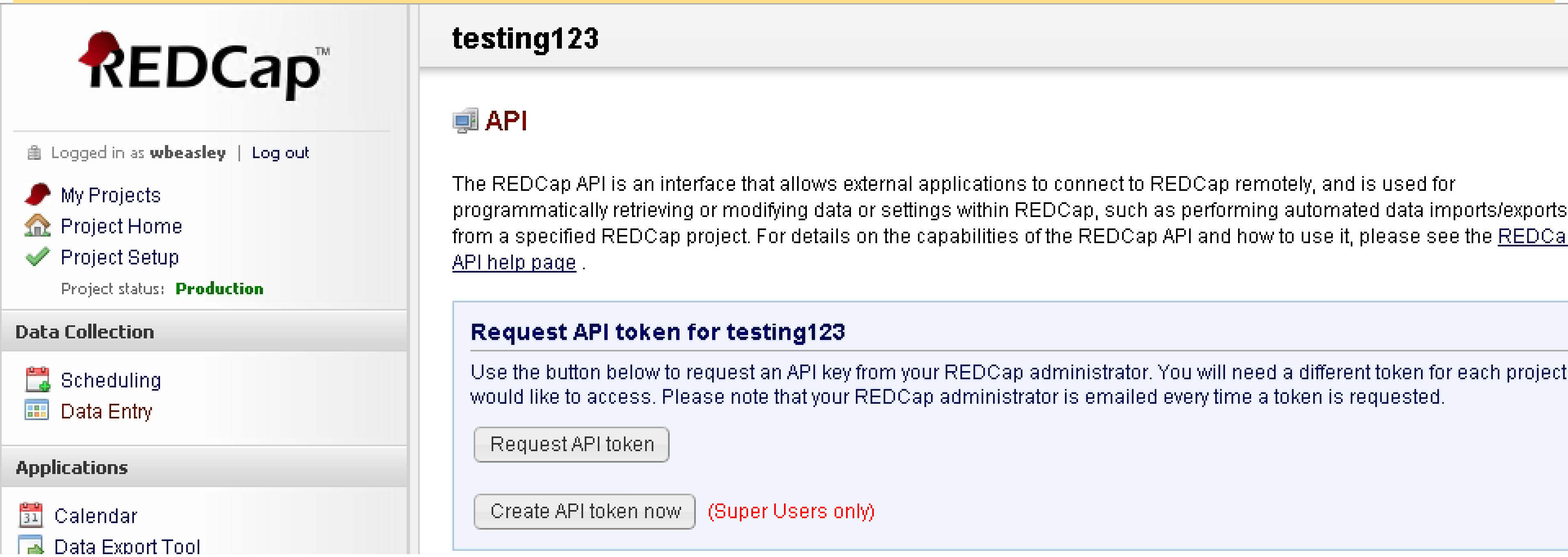
```
### Declare the address of the server and
#   your token (ie, hash of project_id, username, password)
uri <- "https://bbmc.ouhsc.edu/redcap/api/"
token <- "9A81268476645C4E5F03428B8AC3AA7B"

### Call the server
result_read <- redcap_read(redcap_uri=uri, token=token)

### Extract the dataset from the results
ds <- result_read$data
ds
```

```
  record_id first_name age
1         1    Nutmeg   10
2         2    Tumtum   11
3         3    Marcus   79
4         4     Trudy   61
5         5  John Lee   58
```

Exporting records (less secure)

```
### Inspect all returned content
$success
[1] TRUE

$status_code
[1] 200

$outcome_message
[1] "5 records and 3 columns were read from REDCap in 0.34 seconds.
The http status code was 200."

$records_collapsed
[1] ""

$fields_collapsed
[1] "recordid,first_name,age"

$elapsed_seconds
[1] 0.3417
```

Requesting an API token



testing123

API

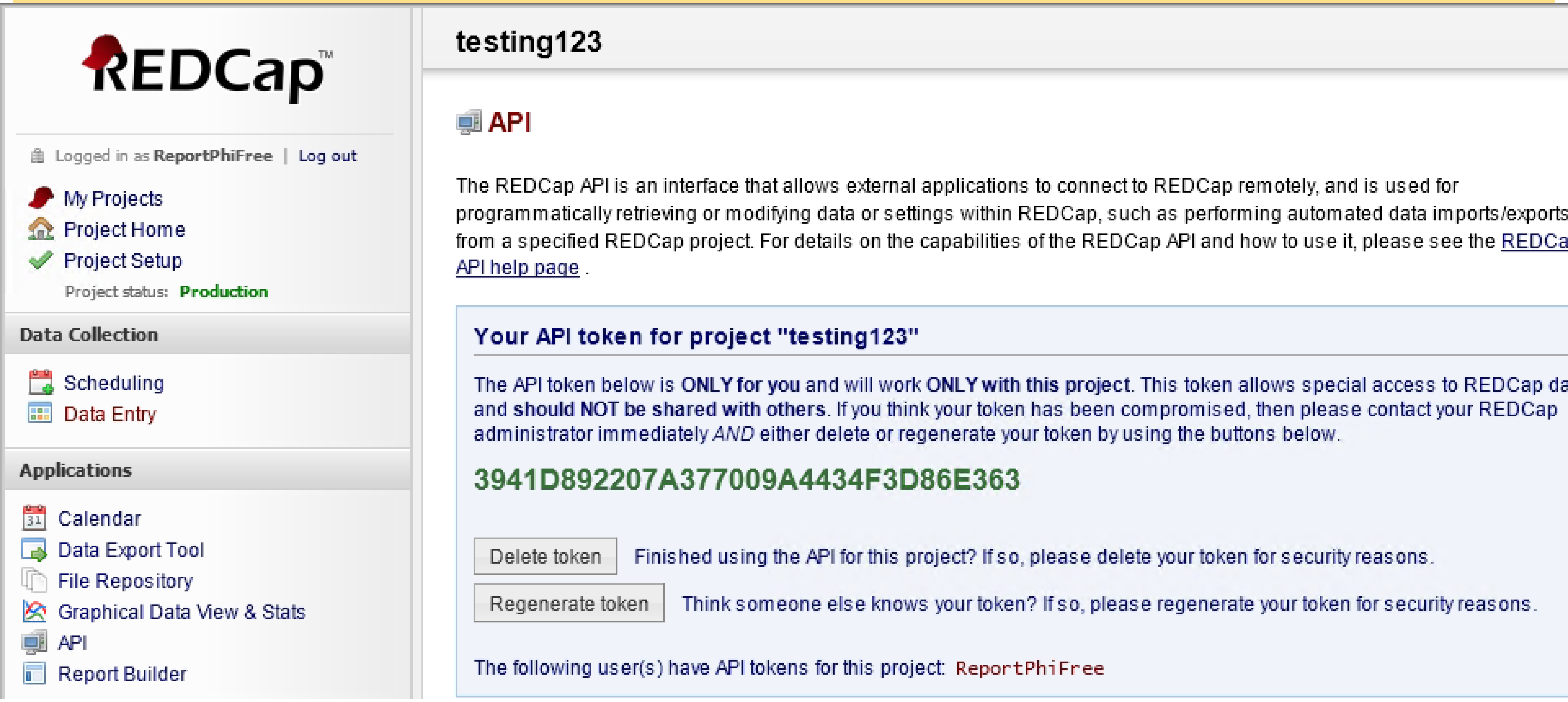
The REDCap API is an interface that allows external applications to connect to REDCap remotely, and is used for programmatically retrieving or modifying data or settings within REDCap, such as performing automated data imports/exports from a specified REDCap project. For details on the capabilities of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

Request API token for testing123

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

- Request your token for a specific project.
- Then wait for the Admin’s approval.

Retrieve a token



testing123

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 REDCa API help page .

Your API token for project "testing123"

The API token below is **ONLY for you** and will work **ONLY with this project**. This token allows special access to REDCap da and **should NOT be shared with others**. If you think your token has been compromised, then please contact your REDCap administrator immediately *AND* either delete or regenerate your token by using the buttons below.

3941D892207A377009A4434F3D86E363

Delete token — Finished using the API for this project? If so, please delete your token for security reasons.

Regenerate token — Think someone else knows your token? If so, please regenerate your token for security reasons.

The following user(s) have API tokens for this project: ReportPhiFree

- Copy the green text into your code.

(This token has been regenerated for security purposes. It won't work anymore. In a sense, we changed the password.)

GitHub:
Version Control Software

- Think MS Word's 'Track Changes' feature, but
  - Retains the entire history of each document.
  - Allows <u>parallel</u> development between people.

- Synchronizes changes among different contributors.
  - A central repository exists on the server.
  - Each developer maintains her own local repository.

- You can establish your first repository and learn the essentials within two hours.

Token Storage (part 1)

Wish List:

1. Code is portable across computers.
2. Code is entirely contained in Git repository.
3. Git repository contains no PHI, passwords, or tokens.
4. Local machine contains no PHI, passwords, or tokens.
5. Tokens are stored, so user doesn't have to retype `9A81268476645C4E5F03428B8AC3AA7B` every operation.

Two feasible options:

– Encrypt and store on local machine (like ssh-agent). Violates #4.

– Use Windows AD/LDAP credentials to call SQL Server. Requires ODBC DSN on local computer, so violates #2.

Token Storage (part 2)

We felt option #2 is the best for OUHSC’s LDAP infrastructure:

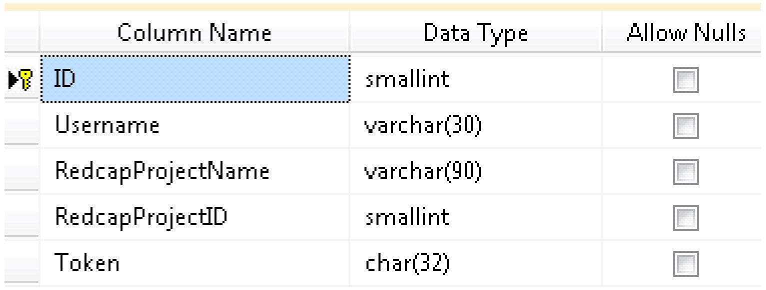
LDAP credentials passed to SQL Server through an ODBC DSN on local computer.

- User needs to maintain only her Windows AD password.
- Password is required only once at OS login.
- That single password is managed securely by the OS, and transmitted across the wire, where the SQL Server database then uses it.
- Unauthenticated users can’t even get into the database, much less retrieve an unauthorized token.
- Git Repository contains no password, token, or even database server address.

→ We host a small database dedicated to serving tokens.

Token Storage (part 3)

- Table:



	Column Name	Data Type	Allow Nulls
🔑	ID	smallint	☐
	Username	varchar(30)	☐
	RedcapProjectName	varchar(90)	☐
	RedcapProjectID	smallint	☐
	Token	char(32)	☐

- Stored Procedure:

  `system_user` returns LDAP username (eg, 'OUHSC/krichards')

```
SELECT Token FROM [RedcapPrivate].tblToken
WHERE Username = system_user
      AND RedcapProjectName = @RedcapProjectName
```

- R Code:

  references the DSN `TokenSecurity` and requests user's token for `Project2` REDCap project.

```
channel <- RODBC::odbcConnect(dsn="TokenSecurity")

token <- RODBC::sqlQuery(channel,
   "EXEC [Redcap].[prcToken] @RedcapProjectName = 'Project2'",
    stringsAsFactors=FALSE)[1, 'Token']

RODBC::odbcClose(channel)
```

Token Storage (part 4)

Safeguards & Concerns:

- Admins for SQL Server are the same for REDCap server, so the threat envelope isn’t larger.
- Table’s accessibility is tighter than the stored procedure’s
- Unfortunately, SQL Server and Windows AD discourage other OSes.

Future plans:

- Wrap in a REDCapR function.

```
token <- retrieve_token(dsn="TokenSecurity", project="Project2")
```

- User inserts their own token through REDCapR.

```
set_token(dsn="TokenSecurity", project="Project2",
   token=prompt_input("Enter Token:")
)
```

Other Storage (part 5)

Use pattern for any content too sensitive for repo.

- Works best for meta-data, not subject-data.
- eg, URL of REDCap Server.
- eg, File path of CSV on file server containing PHI.

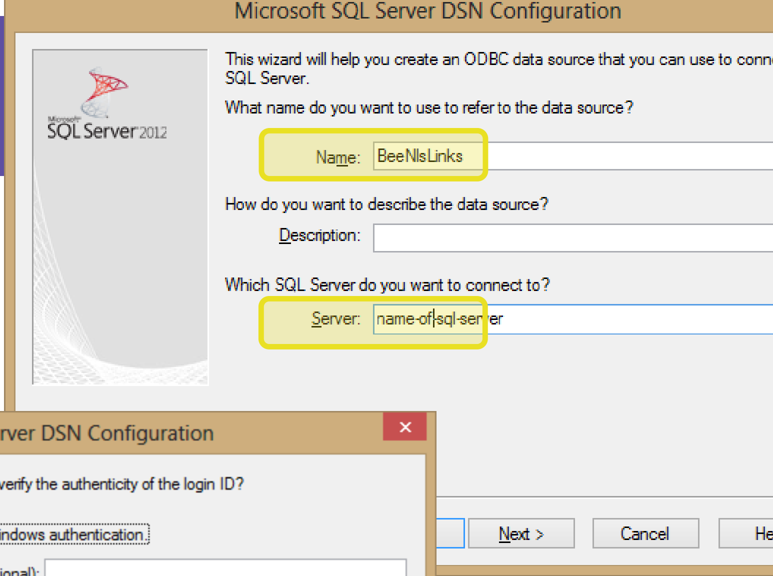
Possible to redirect different users to different values.

- If 5 users have API access to a single REDCap project,
  the database table will have 5 rows,
  each with a unique token.

Establish DSN

1. Download most recent driver
2. Set server name
3. Set to “Integrated Security”
4. Set database name
5. Verify connection

-No passwords-


Microsoft SQL Server DSN Configuration

This wizard will help you create an ODBC data source that you can use to conn
SQL Server.

What name do you want to use to refer to the data source?

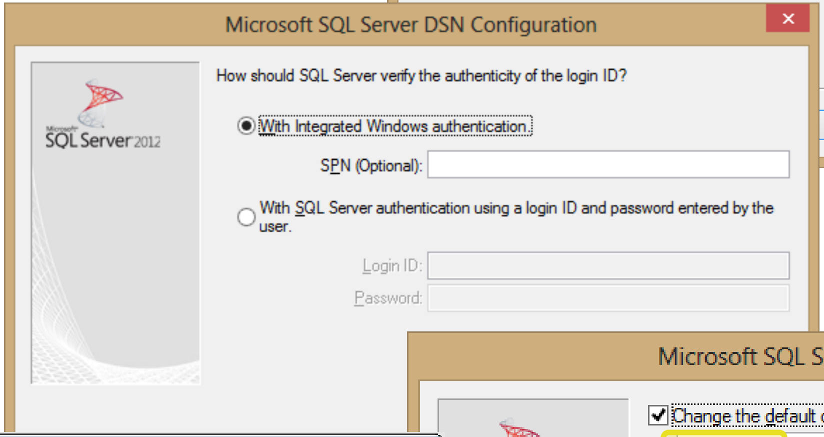
Name: BeeNlsLinks

How do you want to describe the data source?

Description:

Which SQL Server do you want to connect to?

Server: name-of-sql-server


Microsoft SQL Server DSN Configuration

How should SQL Server verify the authenticity of the login ID?

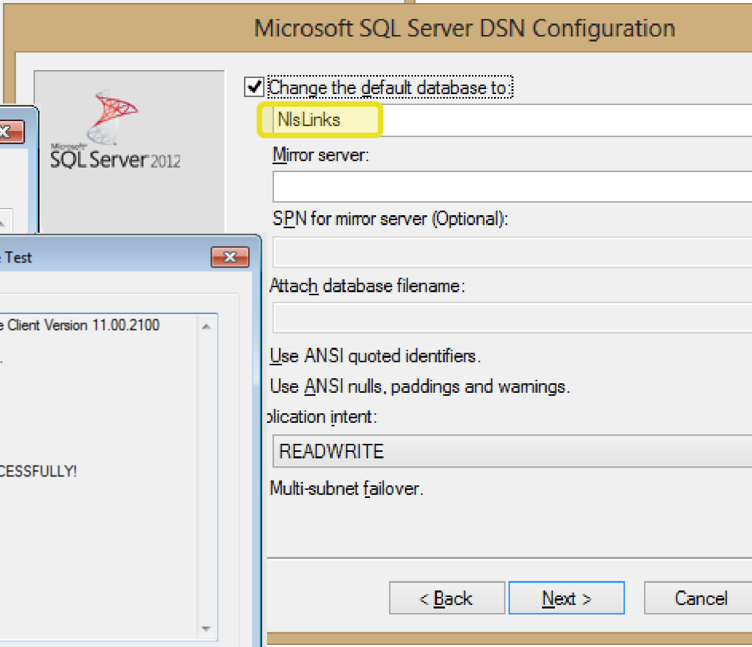
[✓ With Integrated Windows authentication.]

SPN (Optional):

With SQL Server authentication using a login ID and password entered by the user.

Login ID:

Password:


Microsoft SQL Server DSN Configuration

[✓] Change the default database to:

NlsLinks

Mirror server:

SPN for mirror server (Optional):

Attach database filename:

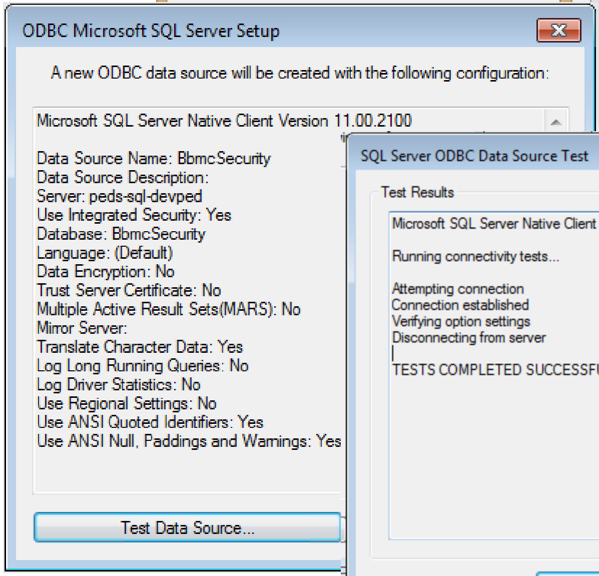
Use ANSI quoted identifiers.

Use ANSI nulls, paddings and warnings.

plication intent:

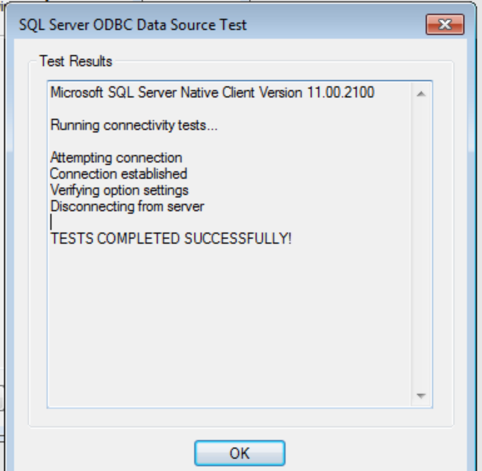
READWRITE

Multi-subnet failover.


ODBC Microsoft SQL Server Setup

A new ODBC data source will be created with the following configuration:

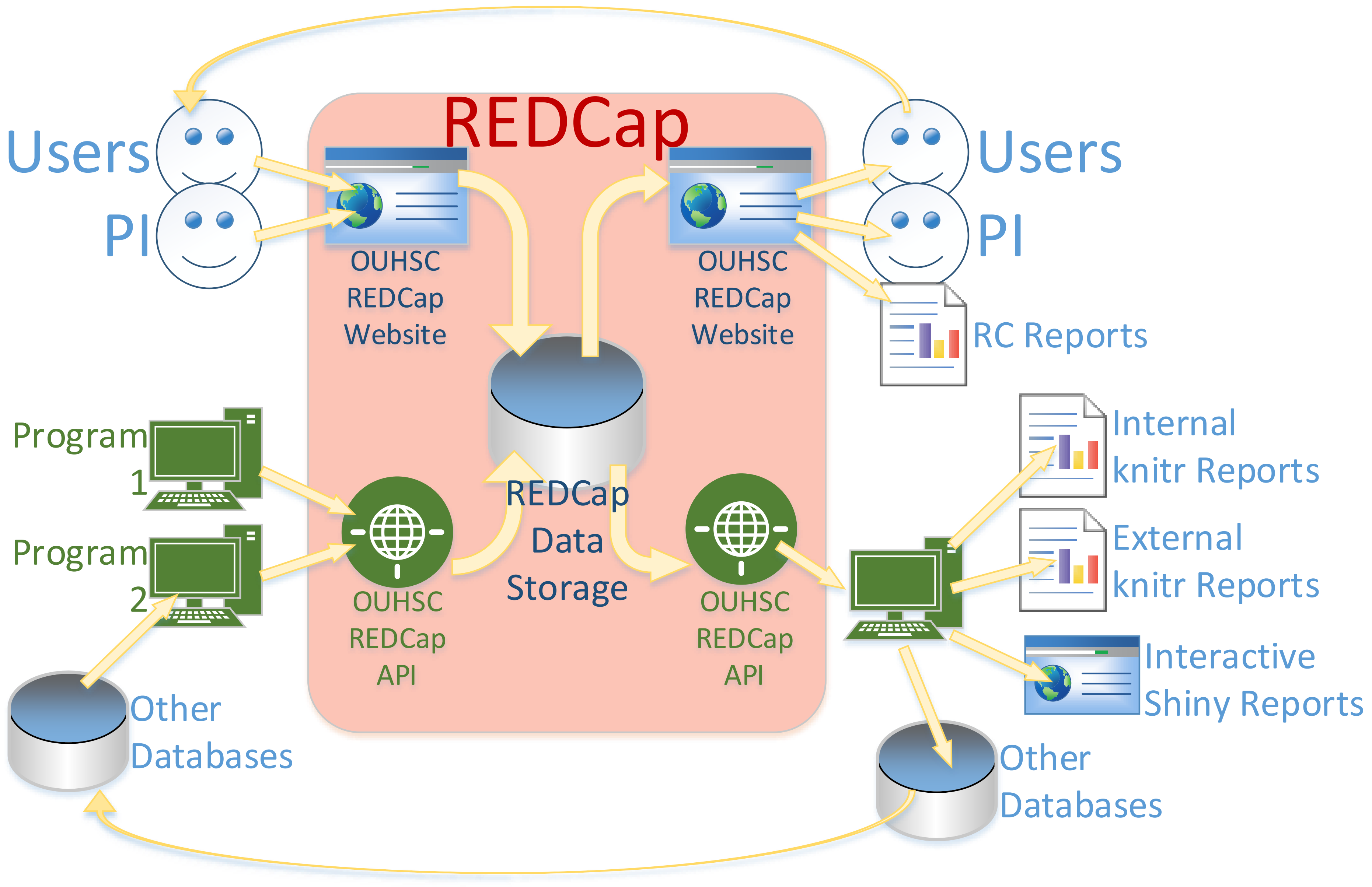
```
Microsoft SQL Server Native Client Version 11.00.2100

Data Source Name: BbmcSecurity
Data Source Description:
Server: peds-sql-devped
Use Integrated Security: Yes
Database: BbmcSecurity
Language: (Default)
Data Encryption: No
Trust Server Certificate: No
Multiple Active Result Sets(MARS): No
Mirror Server:
Translate Character Data: Yes
Log Long Running Queries: No
Log Driver Statistics: No
Use Regional Settings: No
Use ANSI Quoted Identifiers: Yes
Use ANSI Null, Paddings and Warnings: Yes
```


SQL Server ODBC Data Source Test

Test Results

```
Microsoft SQL Server Native Client Version 11.00.2100

Running connectivity tests...

Attempting connection
Connection established
Verifying option settings
Disconnecting from server

TESTS COMPLETED SUCCESSFULLY!
```

Possible REDCap Workflows



Perks of REDCapR (part 1)

1. **Batching:** making smaller calls to server, and combining the results to appear as if only one call was made.
   - Avoids server-time outs.
   - Can suspend between calls, to avoid tying up server.
2. **Translates:** resolves differences between API and R.
   - eg, R stores IDs as a vector `c(10, 20, 30)`, while the API needs a string `"10,20,30"`
3. **Validates:** proactively looks for common mistakes.
   - Helps catch errors sooner,
   - Better error messages b/c it's closer to error's source.
4. **Subset:** easier to avoid retrieving an entire dataset.
   - Fewer rows.
   - Fewer columns.

Perks of REDCapR (part 2)

1. **SSL:** provides extra transport security, by default.
   - Assumes responsibility for updating certificates.

2. **Unit & Integration Tested:** 100+ checks before release.
   - Corner cases are being added every month.

3. **Wider Adoption:** Library is used across multiple projects.
   - More assurances than evolving code that's copy & pasted.
   - Builds on experience within and between libraries (eg, `PyCap` Python package and `redcap` R package).

knitr

- Executes R code, and presents results in a coherent document.
- Eliminates the need to repeatedly copy & paste:
  - Multiple descriptives, graphs, and model results.
  - Updated results after more data trickles in.
- Can produce **markdown** reports that can be quickly produced internal audiences.
- Can produce **LaTeX** reports that can be beautifully crafted for external audiences.

knitr Examples

- Descriptives & graphs in markdown.
- Descriptives & graphs in LaTeX.
- Inferential model results in LaTeX.
- markdown report on GitHub.
  https://github.com/OuhscCcanMiechvEvaluation/MReporting/blob/master/OhcaReports/OhcaReport1/OhcaReport1.md

Shiny

Web interface for interactive graphs, stats & tables.

https://wibeasley.shinyapps.io/SdtThreshold/

- Currently our BBMC server is configured only for public information, not PHI data.
- Consequently, it can’t pull data from REDCap, but csv data can be pushed to it.

Pattern List (part 1)

- **Extractor**: exports/pulls REDCap data to R, and lightly munge, such as
  - calculate timespans,
  - convert categories levels into `factor`s, and
  - clean up missing values.
  - It is called by reports and sanitizers.

- **Gateway**: exports/pulls SQL Server data to R and lightly munge. (Subset of Fowler 2002)
  - It is called by called by reports and other gateways.

Pattern List (part 2)

- **Transfer 1**: moves external data (eg, sent from OSDH) to SQL Server, where it is audit-able.
  - read CSV into R
  - verify structure is the same as the previous import.
  - lightly munge.

- **Transfer 2**: moves data from SQL Server to REDCap
  - eg, so recruiters view in REDCap, instead of SQL Server. It's a lot cheaper/quicker to set up a two-way bound interface in REDCap than in other databases.

¿Any recommendations for distinguishing names?

Pattern List (part 3)

- **Sanitizer**: pulls from a gateway or extractor, and removes PHI and anything else that we don't want publicly exposed. Sanitizers are required before copying data to Shiny.

- **Data Access Layer (DAL)**: collection of the previous 5 patterns. (Fowler 2002)

Pattern List (part 4)

- **Code Behind**: R file that does all the heavy lifting in an analysis
- **Report**: Rmd file that's a thin layer on top of the 'Code Behind' file. It only presents info, and doesn't manipulate or calculate.
- **Common Report Components?**: Contains code and graph templates that are used by multiple reports. The results are typically more consistent, and higher quality.

Goals

- Reproducible research.
  - Facilitates scientific replication.
  - Disseminates techniques to other subfields.
  - Promotes cumulative research.
- Literate programming.
  - Evaluated programs need fresh & frequent feedback.
- Collaborative Development.

Beyond REDCap

- Today’s mechanics were specific to REDCap, but the goals aren’t.

- Regardless of the database chosen,
  - Minimize the repetition code within & between projects.
  - Use tested code where possible.
  - Use code instead of manual operations.
  - Use automated reports instead of ad-hoc reports.