REDCap technical guide

Contents

[Azure environment 1](#_Toc81832634)

[Azure resources 1](#_Toc81832635)

[1. Web App 1](#_Toc81832636)

[2. MySQL database server 1](#_Toc81832637)

[3. SMTP server 1](#_Toc81832638)

[PHP configuration 2](#_Toc81832639)

[Gmail account 3](#_Toc81832640)

[Troubleshooting 3](#_Toc81832641)

[REDCap 3](#_Toc81832642)

[Software 3](#_Toc81832643)

[Configuration 3](#_Toc81832644)

[Authentication 3](#_Toc81832645)

# Azure environment

# Azure resources

## Web App

## MySQL database server

## SMTP server

REDCap requires an email server to function properly. Specifically, for our purposes, it needs to send emails in order for user accounts to be validated, and for passwords to be reset periodically.

An email server was not configured when REDCap was first installed. Russell did not know how to configure an SMTP server. All of the instructions and advice I found on the REDCap Community forum and elsewhere referenced the use of SendGrid. SendGrid is an SMTP (email server) which can be installed and configured in an Azure resource group. Unfortunately, it turned out that due to the nature of the Azure subscription (apparently because SendGrid is a SaaS resource), we would have needed assistance – and probably the assignment of administrator privileges – by the Department of the Premier and Cabinet. This was an unattractive prospect, so I decided to make a further attempt at configuring the SMTP settings in PHP while utilising a different SMTP server. After many hours of trawling StackOverflow, testing, and troubleshooting, I got it working. Below is a summary of the configuration details in the form of instructions (should they have to be repeated in the future).

### PHP configuration

REDCap uses PHP to send emails. Here we have to configure PHP to use our desired SMTP settings, so PHP can successfully send emails when requested.

1. First, install Sendmail on the web server. Sendmail is an application that allows you to [send emails from a Windows machine](https://php.tutorials24x7.com/blog/how-to-use-sendmail-on-windows-to-send-email-using-php) without installing an email server.

Download Sendmail from [this Github repository](https://github.com/sendmail-tls1-2/main), unzip it and load the contents into C:\home\site\repository\sendmail on the Web App using Kudu.

1. Now we configure Sendmail with our SMTP settings. In Kudu, open the sendmail.ini file that you just uploaded and modify it with the following values:

smtp\_server=smtp.gmail.com

smtp\_port=465

auth\_username=cdsircredcap@gmail.com

auth\_password=Cdsirc01!

These are the settings for the Gmail account that was created for this purpose (details below). This allows us to use Gmail’s email server instead of using SendGrid or installing and configuring our own email server (no thanks!).

1. Next, we need to configure PHP to use Sendmail to send emails. Everything else is handled by Sendmail with the settings above. Azure does not allow you to modify source code in most directories (including Program Files), so instead of modifying the php.ini file directly, we have to tell PHP to look for additional configuration settings.

To do so, add a PHP\_INI\_SCAN\_DIR app setting to the Web App by running the [following command](https://docs.microsoft.com/en-in/azure/app-service/configure-language-php?pivots=platform-windows) in the Cloud Shell (Bash):

az webapp config appsettings set --name DfE-REDCap --resource-group CDSIRC-REDcap-RG01 --settings PHP\_INI\_SCAN\_DIR="C:\home\site\ini"

You can confirm that this was successful in Azure Portal. Navigate to the App Service (DfE-REDCap), then to the ‘Configuration’ page under Settings. You should see the above application setting (PHP\_INI\_SCAN\_DIR) listed.

1. Now, using Kudu, navigate to C:\home\site and create the ini folder we referenced above. Inside C:\home\site\ini, create a file called settings.ini and include in it the following line:

sendmail\_path = C:/home/site/repository/sendmail/sendmail.exe

Thanks to the app setting we created in the previous step, PHP will scan this folder and include this setting when it runs. PHP will then invoke sendmail.exe, which includes our SMTP settings, when the mail function is called by REDCap. Great success!

### Gmail account

A Gmail account was created for the purpose of sending emails from REDCap.

The email address is: cdsircredcap@gmail.com

The password is: Cdsirc01!

The recovery email address is: educationcdsirc@sa.gov.au

Note that to send emails from REDCap (e.g., when creating user accounts), the associated email account (cdsircredcap@gmail.com) needs to allow external app connections. The account will automatically disable this feature after some period of disuse. To re-enable this:

1. Login to the account
2. Navigate to ‘Manage your Google Account’ under the gear icon
3. Go to the Security tab
4. Enable ‘Less secure app access’

REDCap should now be able to send emails again.

### Troubleshooting

Some issues can be troubleshooted using PHP itself. To run PHP commands in the Kudu Console, create a PHP file containing your PHP code and run php <filename> in the console.

To check the PHP SMTP settings you can run the phpinfo() command. phpinfo(INFO\_GENERAL) will output basic configuration info, including where (or whether) PHP is searching for additional configuration settings.

You may also want to send a test email. This can be performed by running the following code in a PHP file:

<?php

$from = "cdsircredcap@gmail.com";

$to = "<recipient\_email>";

$subject = "Hello Sendmail";

$message = "This is an test email to test Sendmail. Please do not block my account.";

$headers = [ "From: $from" ];

mail( $to, $subject, $message, implode( '\r\n', $headers ) );

Just replace <recipient\_email> with your own email address.

If the configuration settings above were applied correctly, this email should be sent successfully.

If it is not sent, check the crash.txt file in the directory in which you ran the above code, or check the error.log file in C:\home\site\repository\sendmail for error messages.

# REDCap

## Software

## Configuration

## Authentication

Table-based authentication is in use. User accounts can be added using the admin account. Note that the Gmail account’s setting may need to be altered for this to work. See the Gmail account section on the previous page.

### Admin account

The administrator account (CDSIRC-Admin) has elevated privileges and essentially has complete control over the system. For example, it can be used to suspend other user accounts and delete records. This account must be used with caution and should only be used for administration and configuration purposes.

See the Password Log in Objective for login details.

## Hooks (custom functionality)

### Overview

As per the [REDCap Developer Tools documentation](https://dfe-redcap.azurewebsites.net/redcap/redcap_v11.4.0/Plugins/index.php), a hook is a PHP function that is executed in a predetermined location inside REDCap. This location is determined by the specific hook function used; for example, a redcap\_data\_entry\_form hook will be executed when an instrument is accessed.

Other functions – including user-defined functions – can be placed inside a hook in order to modify either the data in or appearance of that form. Functions can be written in PHP, which will be executed by the web server, or in JavaScript, which will be executed by the browser. HTML and CSS can also be implemented.

Because hooks are executed by REDCap itself (on the web server), they can utilise REDCap’s resources, including the database connection to the mySQL backend, and all available PHP constants and functions used by REDCap.

### Configuration

The REDCap Hooks setting points REDCap to the appropriate location to find the hooks file, currently C:\home\site\wwwroot\redcap\hooks\hooks.php. This setting can be found in REDCap under Control Panel -> General Configuration.

Add hook functions to this file to add functionality to REDCap. The file can be modified using Kudu.

### SEIFA, region, and ARIA assignment

The SEIFA, ARIA (Remoteness Area), and SA Government Region fields are automatically updated with the appropriate value based on the record’s year of death and postcode. The information for these fields is stored in nested dictionaries (JavaScript objects). The values are assigned by passing the (census) year and postcode as keys to the appropriate dictionary; the values are returned and passed to the REDCap fields.

The script is executed inside **function** redcap\_data\_entry\_form($project\_id, $record, $case\_information, $event\_id, $group\_id, $repeat\_instance){} each time the case\_information instrument is modified (the instrument is passed as a parameter).

Note the syntax for accessing or modifying REDCap field values using JavaScript is $('input[name=”<fieldname> “]').val(), passing the field’s name to the name parameter and, if setting the value, passing that value to the val() method.