PDF Reports with MCC Scripts

Objective

Come up with a solution involving MCC Scripts functionality and AwReporting to generate and send performance reports to client accounts under the partners' MCC.
The main idea is to use the least amount of APIs possible, and demand as little work as possible from partners.

Scenario

For this solution, it was considered a partner that does not have a lot of technological expertise, and has at least one person that has some tech background.
The *tech person* needs to have some knowledge of command line usage, and access to the MCC that will be queried in order to generate the reports.

Solution

Due to the success of AwReporting, and because it is a tool that is very easy for partners to install and run, AwReporting will be used as the application that will gather all the data required to generate the PDF reports, and expose the files to the partner via Google Drive, so they can be sent to the clients.

The solution consists on using MCC Scripts to go through all the client accounts, then accessing the PDF files in Google Drive and create and send an email for each client account.

To get all the client accounts, we will be using a spreadsheet, also in Google Drive with the account ID, account email and dates information. This data will be used by the MCC script when creating the email to be sent to partners.

Steps for the Solution

First thing to do, is to install AwReporting and configure the *properties* file properly to access the MCC. All the instructions on how to do that can be found on the GitHub page of the project.
It is super important to configure AwReporting to upload the PDF files to Google Drive. The instructions on how to do that, can be found on the same page as the installation guide.

With AwReporting installed and configured properly, now we need to schedule **two** different runs of the tool: the first one downloads and stores all the data into the database; the second one (must run after the first one) will generate all the PDFs and upload to Google Drive.

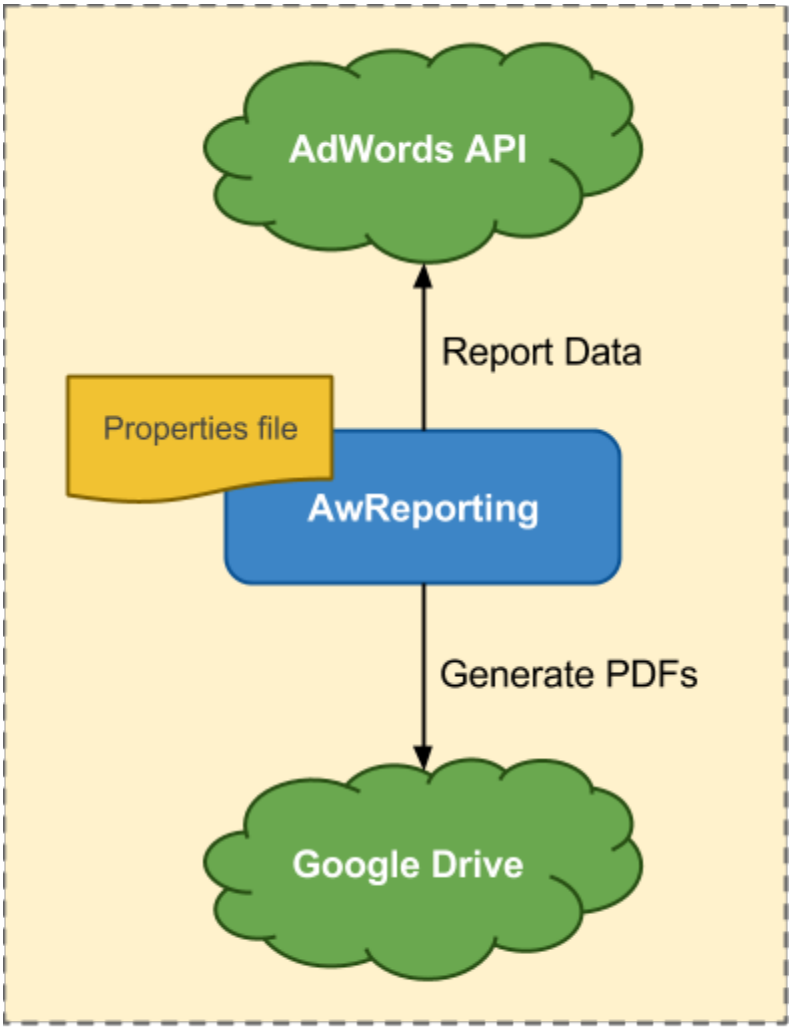
NOTE: *The Google Drive account has to be the same that will be used to run the MCC Scripts. This is mandatory, otherwise the script won't be able to access the PDF files.*

Next step is to create a spreadsheet in the same Google Drive account, that will contain all the clients emails and IDs. This spreadsheet has to be **populated by the partner**, because AdWords does not guarantee that all the client accounts will have a valid email. The spreadsheet will contain **one** client account per line, and a column with the account ID and a column with the email to send to reports.

Now that all the data providers are in place, we just need the MCC Script to go through the client accounts and send out the emails. (*We will provide the scripts*)

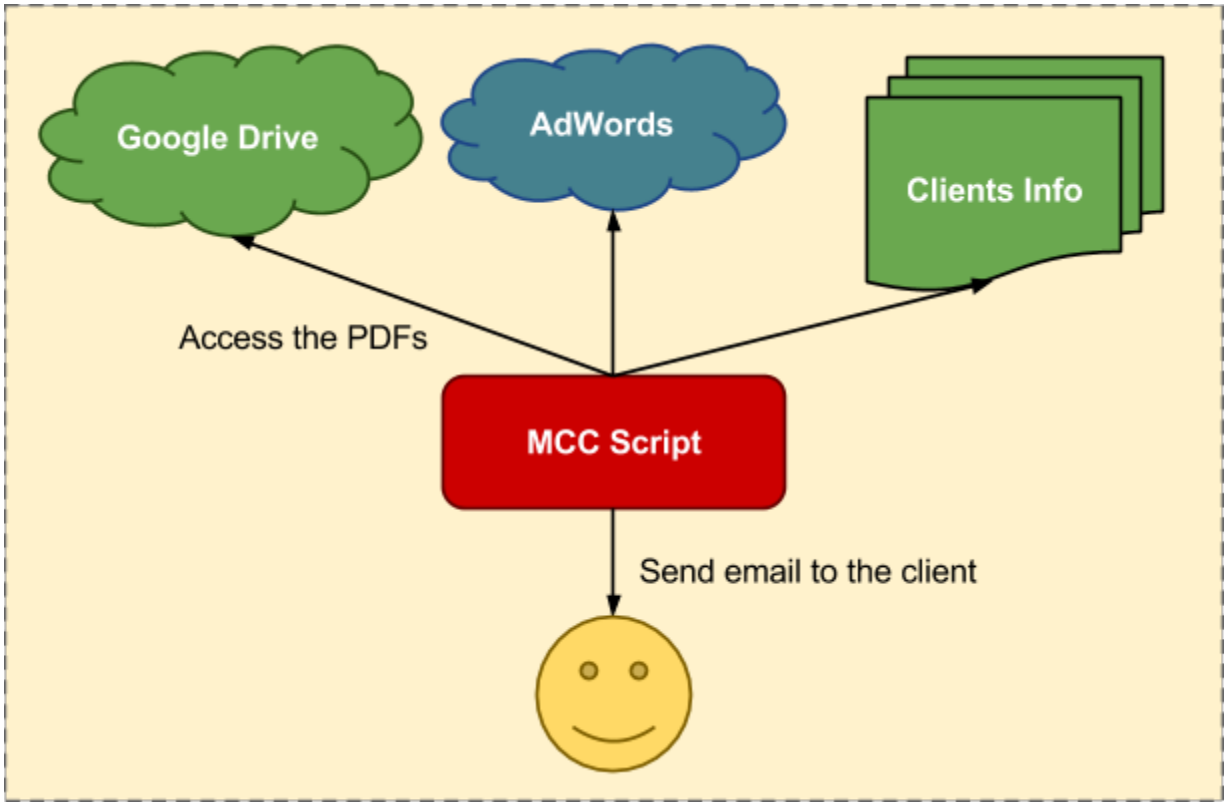
Overview of the Solution

AwReporting running overnight:



This can all be done in a local machine at the partner.

Accessing the MCC and running the script:



The MCC Script will access all the reports that were generated by AwReporting, and combine with the client information that is in the spreadsheet to send out the email to clients.

Effort and Maintenance

The only work required from the partner is to install AwReporting, create the spreadsheet, and schedule the script to run for all the accounts.

It is possible to have the whole solution running within a day, and there is very little maintenance involved. The only part that really needs attention in order to keep things updated is the spreadsheet with all the client contacts.

Adwords Scripts code

```
// SpreadSheet with two columns A=Email B=AccountId
var SPREADSHEET_URL =
'https://docs.google.com/a/google.com/spreadsheets/d/1vMmbp7g5HVKRvcy
qpzw_2F4MYthFqkIV4yIgNbiCxRI/edit#gid=0';

// Names example:
ACCOUNT_PERFORMANCE_REPORT_1231231234_20140601_20140630.pdf
// We add the AccountID in the middle of PREFIX + AccountId + SUFIX
var REPORT_NAME_PREFIX = "ACCOUNT_PERFORMANCE_REPORT_"
var REPORT_NAME_SUFIX = "_20140601_20140630.pdf"

function main() {

  var spreadsheet = SpreadsheetApp.openByUrl(SPREADSHEET_URL);
  var emailAccountIdData = getSheetData(spreadsheet, 0);

  for (var i = 0; i < emailAccountIdData.length; i++) {
    var email = emailAccountIdData[i][0];
    var accountId = emailAccountIdData[i][1];
    Logger.log( "Sending " + email + " " + accountId);
    var file = getFileFromDrive(accountId);
    sendEmailWithAttachments(email, accountId, file);
  }
}

// Gests the Report File from Google Drive
function getFileFromDrive(accountId) {
  var filesIterator = DriveApp.getFilesByName(REPORT_NAME_PREFIX +
accountId + REPORT_NAME_SUFIX);
  while (filesIterator.hasNext()) {
    var file = filesIterator.next();
    return file;
  }
}

// Emails report file to email address
function sendEmailWithAttachments(email, accountId, file) {

  MailApp.sendEmail(
```

```
    email, 'AwReporting Account Report for account ' + accountId,
'Account Performance Report attached.',
    {
      name: 'AwReporting Automatic eMailer', attachments:
[file.getBlob()]
    });
}

/**
 * Retrieves the data for a worksheet.
 * @param {Object} spreadsheet The spreadsheet.
 * @param {number} sheetIndex The sheet index.
 * @return {Array} The data as a two dimensional array.
 */
function getSheetData(spreadsheet, sheetIndex) {
  var sheet = spreadsheet.getSheets()[sheetIndex];
  var range =
      sheet.getRange(2, 1, sheet.getLastRow() - 1,
sheet.getLastColumn());
  return range.getValues();
}
```