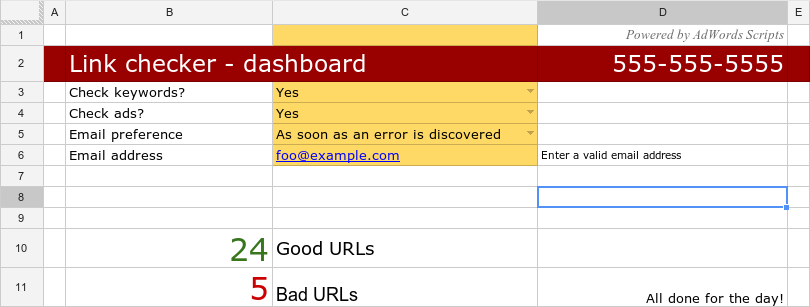
Link Checker



As a website evolves, new pages get added, old pages are taken down, links get broken and fixed. Keeping an AdWords campaign in sync with the website is an ongoing battle for many advertisers. Live advertisements may be pointing to non-existent pages, and the advertiser ends up paying for clicks that yield 404 pages.

Link Checker addresses this problem by iterating through all ads and keywords in your account and making sure their URLs do not produce "Page not found" or other types of error responses. Whenever an issue is encountered, Link Checker will send you an email about it. Alternatively, you can opt out for daily summary emails. Link Checker also maintains a spreadsheet tracking all URLs checked so far today and their statuses.

Link Checker is controlled by a spreadsheet-based dashboard:



Scheduling

The script will not check the same URLs twice on the same day. A single execution of a script checks up to 800 URLs, so that you don't exceed the 20,000 daily quota. Schedule it to run **Daily** if your account has fewer than 800 URLs, or **Hourly** if it's larger than that.

How it works

The script creates a label "link\_checked" in your account and uses it to track the ads and keywords that it already tested so far today. The label gets removed and re-created every day.

The script pulls in keywords and ads that a) have a destination URL, and b) were not yet checked today. It then uses UrlFetchApp to check the URLs, and records the results into a spreadsheet.

Link Checker is limited to 20,000 URLs a day due to quota limitations of UrlFetchApp. In order not to overwhelm your webserver, the script makes at most one URL fetch per second.

Setup

* Set up a spreadsheet-based script with the source code below. Use the Link Checker [template spreadsheet](http://goo.gl/6tvzrb).
* Don't forget to update SPREADSHEET\_URL in code.
* Schedule the script **Daily** or **Hourly**.

[Setting up a spreadsheet-based script](https://developers.google.com/adwords/scripts/docs/solutions/link-checker)

Source code

var SPREADSHEET\_URL = "[YOUR\_URL]";  
  
var LABEL\_NAME = "link\_checked";  
  
var shelper = new SHelper();  
var badUrls = 0;  
  
function main() {  
  dealWithFirstRunOfTheDay();  
  if (shelper.config.email.length == 0 && shelper.config.emailPreference != "Never") {  
    Logger.log("WARNING: no email specified, proceeding...");  
  }  
  if (!shelper.config.checkAds && !shelper.config.checkKeywords) {  
    Logger.log("WARNING: requested no keywords and no ads checking. Exiting.");  
    return;  
  }  
  createLinkCheckerLabel();  
  
  var anythingChanged = shelper.config.checkKeywords && checkUrls(AdWordsApp.keywords());  
  anythingChanged = (shelper.config.checkAds && checkUrls(AdWordsApp.ads())) || anythingChanged;  
  
  if (anythingChanged) {  
    shelper.flush();  
    if (badUrls > 0 && shelper.config.email.length > 0 && shelper.config.emailPreference == "As soon as an error is discovered") {  
      var bad = shelper.spreadsheet.getRangeByName("bad").getValue();  
      var good = shelper.spreadsheet.getRangeByName("good").getValue();  
      sendReportWithErrors(good, bad);  
    }  
  } else {  
    shelper.spreadsheet.getRangeByName("finished").setValue("All done for the day!");  
  }  
}  
  
function dealWithFirstRunOfTheDay() {  
  var date = new Date();  
  var lastCheckDate = shelper.dataSheet.getRange(1, 3).getValue();  
  if (lastCheckDate.length == 0 || date.getYear() != lastCheckDate.getYear() ||  
      date.getMonth() != lastCheckDate.getMonth() || date.getDay() != lastCheckDate.getDay()) {  
    // kill the label.  
    var labels = AdWordsApp.labels().withCondition("Name='" + LABEL\_NAME + "'").get();  
    if (labels.hasNext()) {  
      labels.next().remove();  
    }  
    // send out yesterday's report  
    if (shelper.config.email.length > 0 && (shelper.config.emailPreference == "Once a day" ||  
        shelper.config.emailPreference == "Once a day if there are errors")) {  
      var bad = shelper.spreadsheet.getRangeByName("bad").getValue();  
      var good = shelper.spreadsheet.getRangeByName("good").getValue();  
      if (shelper.config.emailPreference == "Once a day") {  
        if (bad == 0) {  
          MailApp.sendEmail(shelper.config.email,  
              "AdWords Link Checker verified " + good +  
              " URLs on account " + AdWordsApp.currentAccount().getCustomerId() + ", all looking good!", "");  
        } else {  
          sendReportWithErrors(good, bad);  
        }  
      } else if (shelper.config.emailPreference == "Once a day if there are errors" && bad > 0) {  
        sendReportWithErrors(good, bad);  
      }  
    }  
    // reset the spreadsheet  
    shelper.spreadsheet.getRangeByName("account\_id\_dashboard").setValue(AdWordsApp.currentAccount().getCustomerId());  
    shelper.spreadsheet.getRangeByName("account\_id\_report").setValue(AdWordsApp.currentAccount().getCustomerId());  
    shelper.spreadsheet.getRangeByName("date").setValue(date);  
    shelper.spreadsheet.getRangeByName("finished").setValue("Checking links...");  
    shelper.dataSheet.getRange(4, 1, shelper.dataSheet.getMaxRows() - 3, 6).clear();  
  }  
}  
  
function sendReportWithErrors(good, bad) {  
  var emailBody = [];  
  emailBody.push("Summary for account " + AdWordsApp.currentAccount().getCustomerId() +  
      ": " + good + " good URLs, " + bad + " bad ones\n");  
  emailBody.push("Full report available at " + shelper.spreadsheet.getUrl() + "\n");  
  shelper.reset();  
  var row = shelper.readRow();  
  while (row != null && emailBody.length < 200) {  
    if (row[1] >= 300) {  
      var entityType = row[4].length > 0 ? "Keyword: " : "Ad: ";  
      var entityText = row[4].length > 0 ? row[4] : row[5];  
      emailBody.push("Campaign: " + row[2] + ", Ad Group: " + row[3] + ", " + entityType + entityText);  
      emailBody.push(row[0] + " - " + row[1] + " response code.\n");  
    }  
    row = shelper.readRow();  
  }  
  if (emailBody.length >= 200) {  
    emailBody.push("Further URLs omitted. Check the report at " + shelper.spreadsheet.getUrl());  
  }  
  shelper.reset();  
  MailApp.sendEmail(shelper.config.email,  
    "AdWords Link Checker verified found " + bad +  
    " bad URLs on account " + AdWordsApp.currentAccount().getCustomerId() + "",  
    emailBody.join("\n"));  
}  
  
function checkUrls(selector) {  
  var iterator = selector  
    .withCondition("DestinationUrl STARTS\_WITH\_IGNORE\_CASE 'h'")  
    .withCondition("LabelNames CONTAINS\_NONE['" + LABEL\_NAME + "']")  
    .orderBy("DestinationUrl")  
    .withLimit(800)  
    .get();  
  if (!iterator.hasNext()) {  
    return false;  
  }  
  var lastUrl = "";  
  while (iterator.hasNext()) {  
    var entity = iterator.next();  
    if (entity.getDestinationUrl() != lastUrl) {  
      lastUrl = entity.getDestinationUrl();  
      var now = new Date().getTime();  
      var response = UrlFetchApp.fetch(lastUrl, { muteHttpExceptions: true} );  
      var then = new Date().getTime();  
      Utilities.sleep(then - now);  
      if (response.getResponseCode() < 300) {  
        shelper.writeRow(lastUrl, response.getResponseCode());  
      } else {  
        badUrls ++;  
        if (typeof(entity['getHeadline']) != "undefined") {  
          var adText = entity.getType() == "TEXT\_AD" ?  
              entity.getHeadline() + "\n" + entity.getDescription1() + "\n" + entity.getDescription2() :  
              entity.getType();  
          shelper.writeRow(lastUrl, response.getResponseCode(),  
              entity.getCampaign().getName(), entity.getAdGroup().getName(), null, adText);  
        } else {  
          shelper.writeRow(lastUrl, response.getResponseCode(),  
              entity.getCampaign().getName(), entity.getAdGroup().getName(), entity.getText());  
        }  
      }  
    }  
    entity.applyLabel(LABEL\_NAME);  
  }  
  return true;  
}  
  
function createLinkCheckerLabel() {  
  var labels = AdWordsApp.labels().withCondition("Name='" + LABEL\_NAME + "'").get();  
  if (!labels.hasNext()) {  
    AdWordsApp.createLabel(LABEL\_NAME, "Managed by Link Checker, please don't modify!", "#60e020");  
  }  
}  
  
// Spreadsheet helper  
function SHelper() {  
  this.MAX\_ROWS = 20000;  
  this.BATCH\_SIZE = 50;  
  this.spreadsheet = SpreadsheetApp.openByUrl(SPREADSHEET\_URL);  
  this.dataSheet = this.spreadsheet.getSheets()[1];  
  this.config = {  
    checkAds: this.spreadsheet.getRangeByName("check\_ads").getValue() == "Yes",  
    checkKeywords: this.spreadsheet.getRangeByName("check\_keywords").getValue() == "Yes",  
    email: this.spreadsheet.getRangeByName("email\_address").getValue(),  
    emailPreference: this.spreadsheet.getRangeByName("email\_preference").getValue(),  
  }  
  this.globalRow = 4;  
  this.cells = null;  
  this.localRow = 0;  
  
  this.reset = function() {  
    this.globalRow = 4;  
    this.cells = null;  
    this.localRow = 0;  
  };  
  this.readRow = function() {  
    initCells(this);  
    if (this.localRow == this.cells.length) {  
      this.globalRow += this.cells.length;  
      if (this.globalRow >= this.dataSheet.getMaxRows()) {  
        return null;  
      }  
      this.cells = this.dataSheet.getRange(this.globalRow, 2, this.BATCH\_SIZE, 6).getValues();  
      this.localRow = 0;  
    }  
    if (this.cells[this.localRow][0].length > 0) {  
      return this.cells[this.localRow++];  
    } else {  
      return null;  
    }  
  };  
  this.writeRow = function() {  
    fetchCells(this);  
    for (var i = 0; i < arguments.length; i ++) {  
      this.cells[this.localRow][i] = arguments[i];  
    }  
  };  
  this.flush = function() {  
    if (this.cells) {  
      this.dataSheet.getRange(this.globalRow, 2, this.cells.length, 6).setValues(this.cells);  
      this.dataSheet.getRange(1, 1).copyFormatToRange(this.dataSheet, 3, 3, this.globalRow, this.globalRow + this.cells.length);  
    }  
  };  
  function initCells(instance) {  
    if (instance.cells == null) {  
      instance.globalRow = 4;  
      instance.cells = instance.dataSheet.getRange(instance.globalRow, 2, instance.BATCH\_SIZE, 6).getValues();  
      instance.localRow = 0;  
    }  
  }  
  function fetchCells(instance) {  
    initCells(instance);  
    while (!findEmptyRow(instance) && instance.globalRow < instance.MAX\_ROWS) {  
      if (instance.dataSheet.getMaxRows() < instance.globalRow + this.BATCH\_SIZE) {  
        instance.dataSheet.insertRowsAfter(instance.dataSheet.getMaxRows(), instance.BATCH\_SIZE);  
      }  
      instance.flush();  
      instance.globalRow += instance.cells.length;  
      instance.cells = instance.dataSheet.getRange(instance.globalRow, 2, instance.BATCH\_SIZE, 6).getValues();  
      instance.localRow = 0;  
    }  
    if (instance.globalRow >= instance.MAX\_ROWS) {  
      Logger.log("WARNING: maximum length of the spreadsheet exceeded. Exiting.");  
      throw "";  
    }  
  }  
  function findEmptyRow(instance) {  
    for (; instance.localRow < instance.cells.length && !(instance.cells[instance.localRow][0] == null ||  
        instance.cells[instance.localRow][0].length == 0); instance.localRow++);  
    return instance.localRow < instance.cells.length;  
  }  
}