Multi Bidder



Optimizing keyword bids is one of the most important activities when managing AdWords accounts. Keyword bids have a direct impact on the placement of an advertisement (how high does it rank in Google Search) as well as its cost. Advertisers often develop their own unique customized bidding strategies that help them meet their goals.

Automated Rules is a powerful AdWords feature that allows advertisers to modify bids of keywords meeting a certain criteria, and do so on a schedule. Often, advertisers end up creating a large number of Automated Rules, and they become quite difficult to manage.

Multi Bidder offers functionality similar to that of Automated Rules based on a spreadsheet. Each row in a spreadsheet is in effect equivalent to an entire Automated Rule. Managing 100 rules—a difficult task in AdWords—becomes a lot easier.



The spreadsheet above demonstrates a single rule that

* Look at statistics for **THIS\_WEEK\_SUN\_TODAY**.
* Finds all keywords in **Campaign #1** that received more than 1 impressions and whose Ctr is greater than 0.25%
* Increases their bids by 10%, while not exceeding $1.40.

**Warning:** Multi Bidder is a powerful script. It has capacity to cause damage if misused. Proceed with caution.

How it works

The following columns in the spreadsheet must stay in place: **Action**, **Argument**, and **Stop limit**. Other columns can be added or removed as needed. A large number of columns may be used. The following are all valid examples of column names, and corresponding cell values:

* Column: CampaignName STARTS\_WITH '?', cell value: Indonesia\_
* Column: Conversions >= ?, cell value: 4
* Column: Status IN [?], cell value: 'ENABLED', 'PAUSED'

? symbol in the column name gets replaced with the value in the corresponding row. Check documentation for [KeywordSelector](https://developers.google.com/adwords/scripts/docs/reference/adwordsapp/adwordsapp_keywordselector) for a complete set of supported columns.

**Action** is one of the following:

* **Multiply by** - multiplies the keyword bid by the **Argument**. 1.1 will increase the bid by 10%. 0.8 will decrease it by 20%.
* **Add** - adds the **Argument** to the keyword bid. 0.3 will increase the bid by $0.30 (assuming the account is in USD). -0.14 will decrease it by $0.14.
* **Set to First Page Cpc** - sets the keyword bid to the first page cpc. **Argument** is ignored.
* **Set to Top of Page Cpc** - sets the keyword bid to the tope of page cpc. **Argument** is ignored.

**Stop Limit** is used to cap the bid changes that the script is making. If the changes to the bid is positive, the bid won't become larger than **Stop Limit**. If the change is negative, the bid won't become smaller than **Stop Limit**. For instance, if the keyword bid is presently $2.00, and you're increasing it by 25% with a **Stop Limit** of $3.00, the bid will become $2.50. If however the **Stop Limit** is set at $2.30, the bid will become $2.30 as well.

**Results** column is auto-generated by the script. It will contain any errors an execution may have encountered, or an indication of the number of keywords the script fetched and attempted to change. Note that "attempted change" may not mean "actual change": if you try to give a keyword a negative bid, it won't work. Make sure to check the execution logs to see exactly what the script did.Note that **Results** column is the last column in the spreadsheet that will be read in. Should you add any conditions after the **Results** column, they won't apply.

Scheduling

Most commonly used scheduling options for bidding rules are **Daily** or **Weekly**. Be mindful of how the schedule frequency will interplay with the statistics date range you have selected. Since AdWords statistics can be delayed by [up to 3 hours](https://support.google.com/adwords/answer/2544985), avoid scheduling your script **Hourly**.

In some cases, it may not make sense to schedule the script at all. Do what makes sense.

Setup

* Setup a spreadsheet-based script with the source code below. Use the Multi Bidder [template spreadsheet](http://goo.gl/yaR0lp).
* Don't forget to update SPREADSHEET\_URL in code.
* Make sure to preview the script before executing it.
* Consider whether to schedule the script.

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

Source code

var SPREADSHEET\_URL = "[YOUR\_URL]";

var spreadsheetAccess = new SpreadsheetAccess(SPREADSHEET\_URL, "Rules");

var totalColumns;

function main() {
  spreadsheetAccess.spreadsheet.getRangeByName("account\_id").setValue(AdWordsApp.currentAccount().getCustomerId());

  var columns = spreadsheetAccess.sheet.getRange(5, 2, 5, 100).getValues()[0];
  for (var i = 0; i < columns.length; i ++) {
    if (columns[i].length == 0 || columns[i] == 'Results') {
      totalColumns = i;
      break;
    }
  }
  if (columns[totalColumns] != 'Results') {
    spreadsheetAccess.sheet.getRange(5, totalColumns + 2, 1, 1).setValue("Results");
  }
  // clear the results column
  spreadsheetAccess.sheet.getRange(6, totalColumns + 2, 1000, 1).clear();

  var row = spreadsheetAccess.nextRow();

  while (row != null) {
    var argument;
    var stopLimit;
    try {
      argument = parseArgument(row);
      stopLimit = parseStopLimit(row);
    } catch (ex) {
      logError(ex);
      row = spreadsheetAccess.nextRow();
      continue;
    }
    var selector = AdWordsApp.keywords();
    for (var i = 3; i < totalColumns; i ++) {
      var header = columns[i];
      var value = row[i];
      if (!isNaN(parseFloat(value)) || value.length > 0) {
        if (header.indexOf("'") > 0) {
          value = value.replace(/\'/g,"\\'");
        } else if (header.indexOf("\"") > 0) {
          value = value.replace(/"/g,"\\\"");
        }
        var condition = header.replace('?', value);
        selector.withCondition(condition);
      }
    }
    selector.forDateRange(spreadsheetAccess.spreadsheet.getRangeByName("date\_range").getValue());

    var keywords = selector.get();

    try {
      keywords.hasNext();
    } catch (ex) {
      logError(ex);
      row = spreadsheetAccess.nextRow();
      continue;
    }

    var fetched = 0;
    var changed = 0;

    while (keywords.hasNext()) {
      var keyword = keywords.next();
      var oldBid = keyword.getMaxCpc();
      var action = row[0];
      var newBid;

      fetched ++;
      if (action == 'Add') {
        newBid = addToBid(oldBid, argument, stopLimit);
      } else if (action == 'Multiply by') {
        newBid = multiplyBid(oldBid, argument, stopLimit);
      } else if (action == 'Set to First Page Cpc' || action == 'Set to Top of Page Cpc') {
        var newValue = action == 'Set to First Page Cpc' ? keyword.getFirstPageCpc() : keyword.getTopOfPageCpc();
        var isPositive = newValue > oldBid;
        newValue = applyStopLimit(newValue, stopLimit, isPositive);
      }
      if (newBid < 0) {
        newBid = 0.01;
      }
      newBid = newBid.toFixed(2);
      if (newBid != oldBid) {
        changed ++;
      }
      keyword.setMaxCpc(newBid);
    }
    logResult("Fetched " + fetched + "\nChanged " + changed);

    row = spreadsheetAccess.nextRow();
  }
  var now = new Date(Utilities.formatDate(new Date(),
      AdWordsApp.currentAccount().getTimeZone(), "MMM dd,yyyy HH:mm:ss"));
  spreadsheetAccess.spreadsheet.getRangeByName('last\_execution').setValue(now);
}

function addToBid(oldBid, argument, stopLimit) {
  return applyStopLimit(oldBid + argument, stopLimit, argument > 0);
}

function multiplyBid(oldBid, argument, stopLimit) {
  return applyStopLimit(oldBid \* argument, stopLimit, argument > 1);
}

function applyStopLimit(newBid, stopLimit, isPositive) {
  if (stopLimit) {
    if (isPositive && newBid > stopLimit) {
      newBid = stopLimit;
    } else if (!isPositive && newBid < stopLimit) {
      newBid = stopLimit;
    }
  }
  return newBid;
}

function parseArgument(row) {
  if (row[1].length == 0 && (row[0] == 'Add' || row[0] == 'Multiply by')) {
    throw("\"Argument\" must be specified.");
  }
  var argument = parseFloat(row[1]);
  if (isNaN(argument)) {
    throw "Bad Argument: must be a number.";
  }
  return argument;
}
function parseStopLimit(row) {
  if (row[2].length == 0) {
    return null;
  }
  var limit = parseFloat(row[2]);
  if (isNaN(limit)) {
    throw "Bad Argument: must be a number.";
  }
  return limit;
}
function logError(error) {
  spreadsheetAccess.sheet.getRange(spreadsheetAccess.currentRow(), totalColumns + 2, 1, 1)
  .setValue(error)
  .setFontColor('#c00')
  .setFontSize(8)
  .setFontWeight('bold');
}
function logResult(result) {
  spreadsheetAccess.sheet.getRange(spreadsheetAccess.currentRow(), totalColumns + 2, 1, 1)
  .setValue(result)
  .setFontColor('#444')
  .setFontSize(8)
  .setFontWeight('normal');
}

function SpreadsheetAccess(spreadsheetUrl, sheetName) {
  this.spreadsheet = SpreadsheetApp.openByUrl(spreadsheetUrl);
  this.sheet = this.spreadsheet.getSheetByName(sheetName);
  this.cells = this.sheet.getRange(6, 2, this.sheet.getMaxRows(), this.sheet.getMaxColumns()).getValues();
  this.rowIndex = 0;

  this.nextRow = function() {
    for (; this.rowIndex < this.cells.length; this.rowIndex ++) {
      if (this.cells[this.rowIndex][0]) {
        return this.cells[this.rowIndex++];
      }
    }
    return null;
  }
  this.currentRow = function() {
    return this.rowIndex + 5;
  }
}