Sale Countdown



Displaying up-to-date and dynamic values in ad text can be very compelling for viewers. Ad parameters allow you to update numeric values in text ads without having to delete and re-add the ad or wait for policy checks. This script demonstrates how to update your ad parameters hourly to count down until a sale you'd like to promote.

How it works

Suppose you have an upcoming sale:



This script calculates how many days and hours remain until the target date, then finds the specific ad group set up for this promotion, and updates the ad parameters of all keywords in this ad group with the results of the calculation.

**Which ad group?**

The script assumes you have a single ad group for which you want to run this promotion. But what if you wanted to run this promotion for more than one ad group? You have a few options.

Multiple ad groups

If you want to do it for multiple specific ad groups, you could change the AD\_GROUP\_NAME to an array like this:

var AD\_GROUP\_NAMES = ['Widget Sale', 'Widget Sale 2'];

and then modify the getAdGroup to getAdGroups like so:

function getAdGroups(names) {
  var predicateValues = "['" + names.join("','") + "']";
  Logger.log(predicateValues);
  return AdWordsApp.adGroups()
      .withCondition('Name IN ' + predicateValues)
      .withLimit(names.length)
      .get();
}

Next, modify the main() function to use multiple ad groups:

var adGroups = getAdGroups(AD\_GROUP\_NAMES);
while (adGroups.hasNext()) {
  var adGroup = adGroups.next();
  var keywords = adGroup.keywords().get();
  // ...
}

All ad groups in a campaign

You could also run this script for a single campaign by using a similar function to getAdGroup for Campaigns, invoking campaign.adGroups().get() and structuring your main() function in the same manner as the example script.

**What about keyword bids?**

We're updating the ad parameterss tied to our keywords, which makes the ad copy more dynamic. But what if we wanted to modify our keyword bids at the same time? That's fairly straightforward, since we're already looping over the target keywords.

while (keywords.hasNext()) {
  var keyword = keywords.next();
  // We want to update {param1} to use our calculated days and {param2} for hours.
  keyword.setAdParam(1, timeLeft['days']);
  keyword.setAdParam(2, timeLeft['hours']);
  if (timeLeft['days'] < 5) {
    keyword.setMaxCpc(keyword.getTopOfPageCpc());
  } else {
    keyword.setMaxCpc(keyword.getFirstPageCpc());
  }
}

This code updates the max CPC of the keyword to match the TopOfPageCpc for that keyword if we have less than 5 days remaining until our deadline, making sure we are strongly promoting this campaign right before our deadline. It otherwise sets the bid to the first page CPC, making sure that the ad is seen, but not paying for top slot.

**Our sale has started, now what?**

Once we've finished the countdown period and our sale has started, our ad doesn't make as much sense. It would be much more relevant to promote the sale with different verbiage. Let's enhance our example to better promote the sale while it is on.

First, let's imagine that we've already created a second ad group that is promoting the ongoing sale. We'll call it "Widget Sale In Progress". Let's add some new variables at the top of the script, like so:

// Ad Group that has text ads with AdParams for use while the sale is ongoing.
var AD\_GROUP\_NAME\_IN\_PROGRESS = 'Widget Sale In Progress';
var END\_DATE\_OF\_SALE = new Date('December 31, 2012');

Next, let's alter the rest of the main function body to incorporate this new logic:

function main() {
  var timeLeft = calculateTimeLeftUntil(END\_DATE, new Date());
  var adGroup = getAdGroup(AD\_GROUP\_NAME);
  if (timeLeft['days'] <= 0 && timeLeft['hours'] <= 0) {
    adGroup.pause();
    adGroup = getAdGroup(AD\_GROUP\_NAME\_IN\_PROGRESS);
    adGroup.enable();
    timeLeft = calculateTimeLeftUntil(END\_DATE\_OF\_SALE, new Date());
  }
  var keywords = adGroup.keywords().get();
  // ...
}

We've added some logic to check if we've passed the start of the sale. If we have, we pause the (countdown) adGroup, get the adGroup for the in-progress portion, enable it and calculate a new timeLeft based on the end of the sale.

Scheduling

Scheduled to run hourly, the script creates dynamic ad text that counts down the hours to the sale.

Setup

* Create a new ad group with a text ad. On one of the ad's description lines, add some ad params, e.g.:
	+ Description: Only {param1:a few} days {param2:and} hours left!

Use any value you like for the other fields.

* Add a few relevant keywords to this ad group.
* Create a new AdWords script with the source code below.
* At the top of the script replace the value of the AD\_GROUP\_NAME variable with the name of the ad group you created.
* Upon previewing, you'll see that it will set the ad parameters to the days and hours left until your target date for each keyword in the ad group you specified. You can preview how your ads with the [Preview Tool](https://adwords.google.com/d/AdPreview).

[Creating an AdWords script](https://developers.google.com/adwords/scripts/docs/solutions/sale-countdown)

Source code

// Date to use to find out how many days are remaining.
var END\_DATE = new Date('December 21, 2012');
// Change this to the Ad Group you set up with text ads with AdParams.
var AD\_GROUP\_NAME = 'Widget Sale';

function main() {
  var timeLeft = calculateTimeLeftUntil(END\_DATE);
  var adGroup = getAdGroup(AD\_GROUP\_NAME);
  var keywords = adGroup.keywords().get();
  while (keywords.hasNext()) {
    var keyword = keywords.next();
    // We want to update {param1} to use our calculated days and {param2} for hours.
    keyword.setAdParam(1, timeLeft['days']);
    keyword.setAdParam(2, timeLeft['hours']);
  }
}

var DAY\_IN\_MILLISECONDS = 1000\*60\*60\*24;

function calculateTimeLeftUntil(end) {
  var current = new Date();
  var timeLeft = {};
  var daysFloat = (end - current) / (DAY\_IN\_MILLISECONDS);
  timeLeft['days'] = Math.floor(daysFloat);
  timeLeft['hours'] = Math.floor(24 \* (daysFloat - timeLeft['days']));
  return timeLeft;
}

function getAdGroup(name) {
  var adGroupIterator = AdWordsApp.adGroups()
      .withCondition('Name = "' + name + '"')
      .withLimit(1)
      .get();
  if (adGroupIterator.hasNext()) {
    return adGroupIterator.next();
  }
}