MEMO

Dein Erstes Google Ads Script

Von der Idee zur ersten selbst erstellten Lösung

Erfahre anhand eines konkreten Beispiels, wie Du Dir auch ohne fundierte Programmierkenntnisse mit überschaubarem Zeitaufwand eine individuelle Automatisierungslösung für Deine Google Ads Kampagnen selbst bauen kannst. Grundkenntnisse in JavaScript solltest Du idealerweise mitbringen.

Inhaltsverzeichnis

MEMO

Vorwort

Google Ads Scripts

Jenseits von Filtern, Regeln und Bulk-Uploads bieten Dir Google Ads Scripts eine ungleich größere Auswahl an Möglichkeiten, die Prozesse bei der Überwachung und Optimierung Deiner Google Ads Kampagnen zu automatisieren. Auf der zugehörigen Developers Seite findest Du eine Einführung (Guides) und die Referenz (Reference) zu Google Ads Scripts, sowie viele Code-Beispiele (Examples) für einzelne Funktionen bzw. Abfragen und sogar einige fertige Komplettlösungen (Solutions). Letztere sind in der Regel einfach konfigurierbar und lassen sich so relativ leicht den jeweiligen Bedürfnissen anpassen.

Dein individuelles Problem

Die vorgenannten Beispiele decken schon eine ganze Reihe an Anwendungsfällen ab oder lassen sich durch mehr oder weniger geringfügige Änderungen entsprechend anpassen. Oft sind die Vorlagen aber doch zu weit weg von der eigentlich gewünschten Lösung. Das ist der Zeitpunkt, selbst Hand anzulegen und ein neues, eigenes Google Ads Script - sozusagen auf der grünen Wiese - zu erstellen.

Unser konkretes Beispiel

Google legt die Anzeigenausrichtung nach Standort und Sprache bekanntlich relativ flexibel aus:

> *“Das Language Targeting erlaubt es, User zu targeten, die der jeweiligen Sprache mit sehr hoher Wahrscheinlichkeit mächtig sind. Hierzu verwenden wir unter anderem die gewünschte Sprache des Users bei Youtube, den Standort und die Browsereinstellung.”*

Dadurch werden selbst bei sehr eingeschränkter regionaler Ausrichtung die Anzeigen auch auf fremdsprachigen Seiten mit ausländischen Top-Level-Domains ausgespielt, die in unserem Fall erwartungsgemäß schlechte Performance-Daten aufweisen.

MEMO

Die Idee

Der Impuls für die Entwicklung einer Automatisierung kommt naturgemäß meistens aus dem operativen Account Management. Unser Kollege Robert erklärt im Interview, wie sich aus seinem Wunsch nach einer Lösung für sein konkretes Problem das Skript aus diesem Artikel entwickelt hat.

***Richard**:* Hi Robert, kannst Du noch einmal kurz zusammenfassen, wie die Idee für eine automatisierte Lösung zum Ausschluss von Placements zustande gekommen ist?

***Robert**:* Die Antwort ist relativ einfach: aus dem täglichen Doing heraus. Als Account-Manager war mir beim Reporting einiger neuer Display-Kampagnen in einem unserer Kunden-Accounts aufgefallen, dass besonders viele und auf den ersten Blick doch eher unpassende Placements im Report “Wo Anzeigen ausgeliefert wurden” aufgeführt wurden. So hatten sich auf der ersten Seite des Berichts eine Vielzahl an Placements mit Endungen wie .ru, .pl oder .hu angehäuft - Placements bzw. Domainendungen also, die zumindest zu einem guten Teil an der eigentlich adressierten Zielgruppe vorbeigehen dürften.

***Richard**:* Hattest Du noch weitere Anhaltspunkte dafür, dass die besagten Placements eher ungeeignet sind, einmal abgesehen vom Umstand mit den unpassenden Top-Level-Domains?

***Robert**:* Ja, tatsächlich war mir bei einem genaueren Blick auf einige der Placements auch deren konkrete Performance negativ aufgefallen. Dazu ein Beispiel:

Placement	Typ	Anzeigengruppe	Klicks	Impr.	CTR	Durchschn. CPC	Kosten
Summe: Wo Anzeigen ausgeliefert wurden			795	63.853	1,25 %	0,19 €	150,48 €
inc-news.ru	Website	Relevante Zielgruppen	53	97	54,64 %	0,17 €	9,03 €

Dieses Placement ist auch in anderen Display-Kampagnen immer wieder aufgetaucht und hatte dabei ähnlich “verdächtige” Kennzahlen ausgewiesen. Bei einer CTR von über 50% im Display-Netzwerk hatten hier bei mir als verantwortlicher Account-Manager sofort die Alarmglocken geläutet und das Placement musste raus.

***Richard**:* Wie ging es dann weiter?

***Robert**:* Ausgehend vom manuellen Placement-Ausschluss und der doch relativ großen Menge potenziell ungeeigneter Placements kam relativ bald die Frage auf, ob sich dieser Prozess denn nicht - zumindest in Teilen - automatisieren ließe. In meiner naiven Vorstellung konnte es doch nicht so schwer sein, Google mitzuteilen, dass wir gerne nur Placements mit Endungen wie beispielsweise .de, .org oder .com drinnen haben und andere eben ausschließen möchten. Der bis zu diesem Zeitpunkt manuell durchgeführte Placement-Ausschluss gestaltete sich vergleichsweise aufwändig, da hier jedes einzelne Placement einzeln eingetragen und bestätigt werden musste. Leider musste ich aber relativ schnell feststellen, dass ein pauschales Ausschließen von ungewünschten TLDs mit Bordmitteln von Google so nicht zu bewerkstelligen war - vermutlich weil auch nicht unbedingt gewünscht. 😉

***Richard**:* Und dann ging es als Nächstes auch schon an die Recherche und ans gemeinsame “Scripten”?

***Robert**:* Richtig. Die Idee war relativ schnell geboren und musste jetzt “nur” noch in ein entsprechendes Google Ads-Script gepackt werden. Dass sich dieses Skript, wenn einmal fertig, dann mit hoher Wahrscheinlichkeit auch gewinnbringend für andere Kunden-Accounts einsetzen ließe, war während des gesamten Prozesses eine schöne, zusätzliche kleine Motivationsspritze.

***Richard**:* Wie zufrieden bist Du mit dem gemeinsamen Ergebnis?

***Robert**:* Sehr zufrieden. Insbesondere auch deshalb, weil sich an die erste Version des Skripts sehr schnell eine zweite angeschlossen hat, die die erste direkt noch um eine Funktion für ein Whitelisting bestimmter, erwünschter Placements erweitert hat.

***Richard**:* Könntest Du noch kurz erklären, was es damit auf sich hat?

***Robert**:* Nach ca. einer Woche Arbeit mit der ersten Version des Skripts hat sich - erneut aus dem täglichen Doing heraus - schnell noch ein relativ wichtiger Punkt herauskristallisiert: es gibt eben doch auch Placements, die nicht auf die vertrauten TLDs enden, die aber dennoch bespielt werden sollen (ein schönes Beispiel dazu: “bedienungsanleitu.ng” - vermutlich schon einmal dem ein oder der anderen begegnet). So war also relativ schnell klar - wir brauchen eine Whitelisting-Funktion, um bestimmte Placements dauerhaft von einem automatischen Ausschluss ausnehmen zu können.

***Richard**:* In drei Stichpunkten - was waren Deine wichtigsten Learnings aus dem gesamten Prozess?

***Robert**:* (1) Die Erstellung von Google Ads-Scripts folgt einer interessanten Mischung aus gesundem Menschenverstand und purer Logik. (2) Grundlegende JavaScript-Kenntnisse vereinfachen die Erstellung zweifellos, man ist aufgrund der umfangreichen und ausführlichen Google-Dokumentation aber auch ohne solche Kenntnisse nicht komplett aufgeschmissen. (3) Google Ads-Scripts möchten sich von sich aus gerne “weiterentwickeln”, man muss ihnen nur ein wenig dabei behilflich sein.. 😉

***Richard**:* Vielen Dank für das Interview!

MEMO

Der manuelle Prozess

Wie Robert im Interview beschrieb, wurden unerwünschte Placements bisher in der Ansicht “Wo Anzeigen ausgeliefert wurden” mittels Filter ausgewählt und als Placement-Ausschlüsse in der jeweiligen Kampagne ergänzt. Ein direktes Hinzufügen dieser Placements aus dieser Ansicht in eine gemeinsame Ausschlussliste ist nicht möglich - sie müssten händisch kopiert oder abgetippt werden.

Der Wunsch

Weil diese marktfremden Placements sinnvolle Ausschlüsse für mehrere, wenn nicht alle Kampagnen sind, sollten diese automatisiert in einer gemeinsamen Placement-Ausschlussliste gepflegt werden, die für alle betreffenden Kampagnen angewendet wird.

Die Lösung

Um die konkreten Anforderungen für unser eigenes Google Ads Script zu definieren, überlegen wir zunächst, wie wir den Prozess in einzelne Schritte zerlegen können:

Vorbereitung: Placement Ausschlussliste erstellen und für Kampagne(n) anwenden

Auch das könnte man automatisieren, wir haben uns aber dazu entschieden, diese Einmalaufgabe manuell zu erledigen.

1. Schritt: Unerwünschte Placements erkennen

Das Script soll zuerst eine Liste mit den Placements erstellen, die als Ausschluss in Frage kommen.

2. Schritt: In Ausschlussliste ergänzen

Sofern die unerwünschten Placements noch nicht ausgeschlossen sind, sollen sie in der Ausschlussliste ergänzt werden.

3. Schritt: Bestätigung per E-Mail verschicken

Nach der Ausführung soll das Script eine E-Mail mit den ausgeschlossenen Placements an den zuständigen Account Manager verschicken.

Die Recherche

Für jeden Prozessschritt werden wir eine Funktion oder Abfrage benötigen. Im Vorfeld recherchieren wir deshalb, welche Objekte innerhalb der Google Ads Script API dafür in Frage kommen:

1. Schritt: Unerwünschte Placements erkennen

Zur Abfrage von Daten aus Google Ads hält die API die Methode search(query, optArgs) bereit.
Die Abfragen kann man in der Google Ads Query Language schreiben.
Damit können eine ganze Menge an Datenquellen und -feldern abgefragt werden.
Und es gibt einen Google Ads Query Builder, um diese Abfragen zu bauen.
Auf den ersten Blick kommen drei Ressourcen für unsere Aufgabe in Frage:

- Detail Placement View
- Group Placement View
- Managed Placement View

2. Schritt: In Ausschlussliste ergänzen

In der Google Ads Script API gibt es ein offenbar passendes Objekt mit der Bezeichnung AdsApp.Excluded Placement List, mit dessen Methoden addExcludedPlacement und addExcludedPlacements einzelne oder auch mehrere Placements gleichzeitig zu einer Placement-Ausschlussliste hinzugefügt werden können.

3. Schritt: Bestätigung per E-Mail verschicken

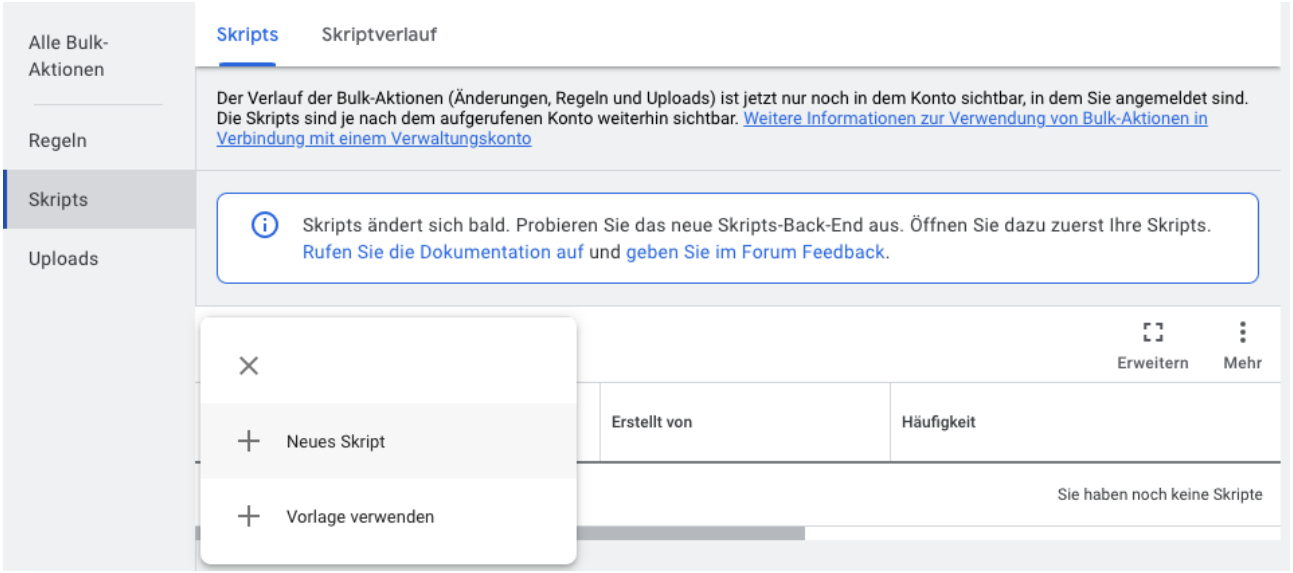
Unter den Code-Beispielen auf der Google Ads Scripts Seite findet sich unter Utilities > MailApp das Snippet “Send a simple email”, das offenbar genau diesen Zweck erfüllt.

Die Erstellung

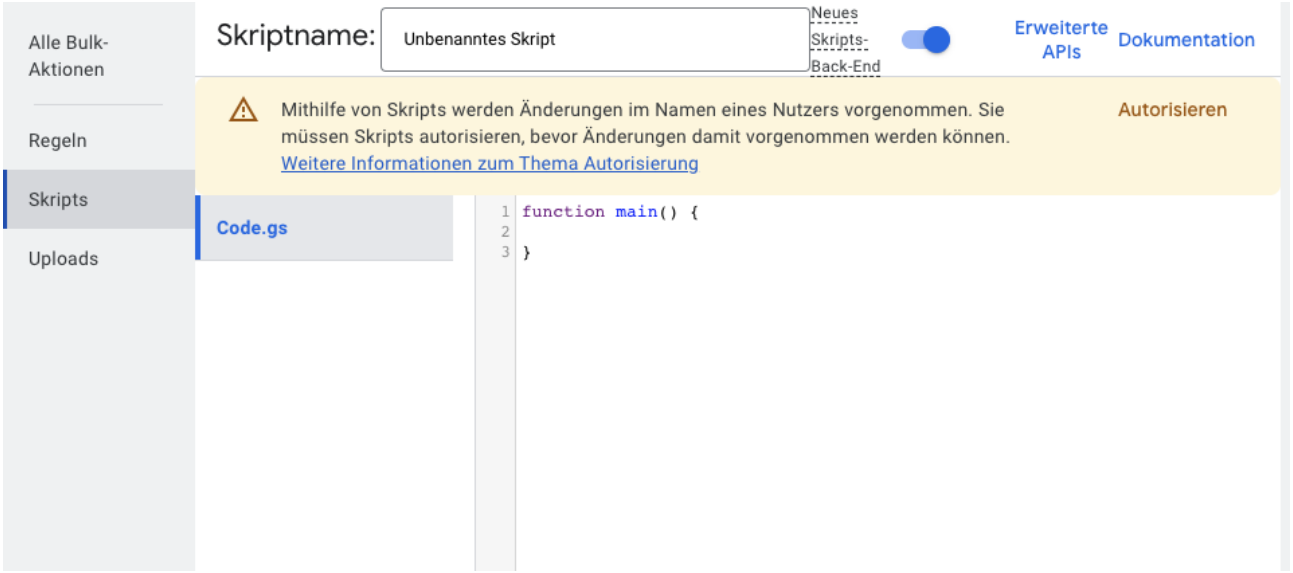
Nach Abschluss unserer Recherche können wir direkt mit der Erstellung des Script-Codes anfangen.

Neues Script anlegen

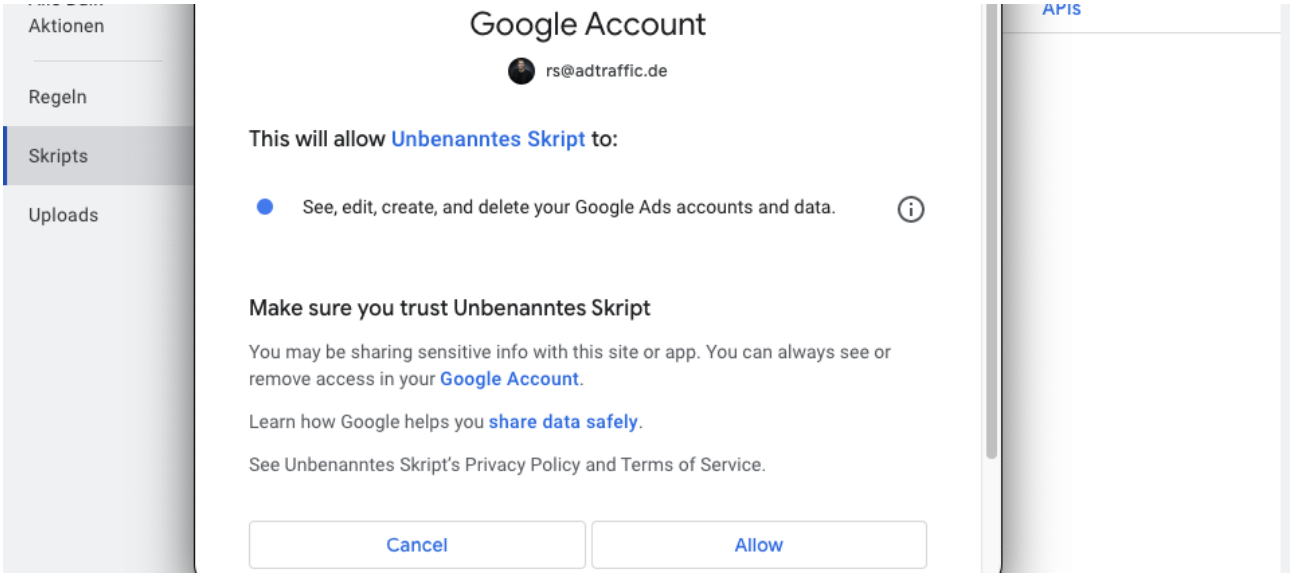
Dazu legen wir uns zuerst ein neues Skript im betreffenden Google Ads Account an. Zum Bereich “Skripts” gelangt man in Google Ads über die Header-Navigation durch Klicken auf “Tools und Einstellungen > Bulk-Aktionen > Skripts”. Ein neues leeres Skript wird durch Klick auf den blauen Plus-Button und die anschließende Auswahl der Option “Neues Skript” erstellt.



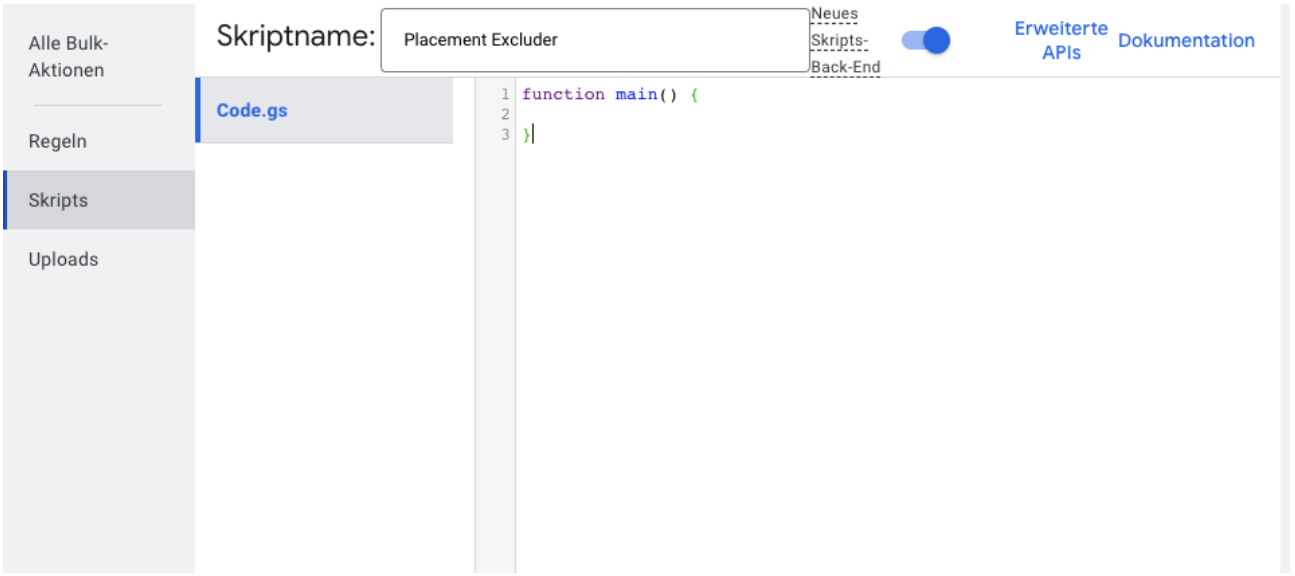
Das neu erstellte Skript fordert zunächst eine Autorisierung an, um später Änderungen im Namen des jeweiligen Nutzers vornehmen zu können.



Nach Klick auf “Autorisieren” kann man zunächst das gewünschte Konto auswählen und dem Skript dann die benötigten Zugriffe erlauben.



Dem Skript kann man auch gleich einen passenden Namen geben.



Jetzt kann es losgehen mit der Erstellung des Codes…

Funktion queryPlacementUrls()

Im ersten Schritt möchten wir unerwünschte Placements erkennen. Dazu schreiben wir uns eine Funktion, die uns ein sortiertes Array zurückgibt.

```
function queryPlacementUrls(){
  var placementUrls = [];
  return placementUrls.sort();
}
```

Um das Array zu füllen, erstellen wir uns mithilfe des Google Ads Query Builders zunächst den Code zur Abfrage der unerwünschten Placements in unserer/n Display-Kampagne/n.

MEMO

In unserem Beispiel durchsuchen wir dazu alle aktiven und pausierten Kampagnen vom Typ “DISPLAY”, deren Name mit “GDN_” beginnt und fragen alle Placements vom Typ “WEBSITE” ab, die nicht auf eine der im Suchstring angegebenen Top-Level-Domains enden.

```
  var query =
    "SELECT " +
    "  group_placement_view.target_url " +
    "FROM group_placement_view " +
    "WHERE " +
    "  campaign.status IN ('ENABLED', 'PAUSED') " +
    "  AND campaign.advertising_channel_type IN ('DISPLAY') " +
    "  AND group_placement_view.placement_type IN ('WEBSITE') " +
    "  AND group_placement_view.target_url NOT REGEXP_MATCH
'.*\.(de|at|ch|com|net|org|info)' " +
    "  AND campaign.name REGEXP_MATCH '^GDN_.*$' " +
    ""
    ;
  var result = AdsApp.search(query, {apiVersion: 'v12'});
```

Das Ergebnis der Abfrage verarbeiten wir anschließend in einer Schleife und schreiben die URL jedes einzelnen Placements in das vorbereitete Array:

```
  while (result.hasNext()) {
    var row = result.next();
    if ( placementUrls.indexOf(row['groupPlacementView']['targetUrl']) == -1
) {
      placementUrls.push(row['groupPlacementView']['targetUrl']);
    }
  }
```

Die fertige Funktion sieht in unserem Beispiel wie folgt aus:

```
function queryPlacementUrls(){
  var placementUrls = [];
  var query =
    "SELECT " +
    "  group_placement_view.target_url " +
    "FROM group_placement_view " +
    "WHERE " +
    "  campaign.status IN ('ENABLED', 'PAUSED') " +
    "  AND campaign.advertising_channel_type IN ('DISPLAY') " +
    "  AND group_placement_view.placement_type IN ('WEBSITE') " +
    "  AND group_placement_view.target_url NOT REGEXP_MATCH
'.*\.(de|at|ch|com|net|org|info)' " +
    "  AND campaign.name REGEXP_MATCH '^GDN_.*$' " +
    ""
    ;
  var result = AdsApp.search(query, {apiVersion: 'v12'});
  while (result.hasNext()) {
    var row = result.next();
    if ( placementUrls.indexOf(row['groupPlacementView']['targetUrl']) == -1
) {
      placementUrls.push(row['groupPlacementView']['targetUrl']);
    }
  }
  return placementUrls.sort();
}
```

Um die Funktion im Skript auszuführen, ergänzen wir den Aufruf innerhalb der Hauptfunktion main(). Den Rückgabewert der Funktion schreiben wir zur späteren Verwendung die Variable “placementUrls”:

```
function main(){
  var placementUrls = queryPlacementUrls();
}
```

Funktion getExcludedPlacements()

Bevor wir neue Placements in unserer Placement-Ausschlussliste ergänzen, möchten wir zuerst eine Liste der bereits ausgeschlossenen Placements erstellen, um diese nicht (noch einmal) zu verarbeiten.

Auch diese Funktion bereitet zu Beginn ein leeres Array vor, das anschließend befüllt und danach zurückgegeben wird. Mit der Variable “queryListName” übergeben wir später aus der Hauptfunktion main() heraus den Namen der Placement-Ausschlussliste, die das Skript verwenden soll.

```
function getExcludedPlacements(queryListName){
  var excludedPlacements = [];
  var sharedExcludedPlacementList = getExcludedPlacementList(queryListName);
  var sharedExcludedPlacementIterator =
sharedExcludedPlacementList.excludedPlacements().get();
  while (sharedExcludedPlacementIterator.hasNext()) {
    var sharedExcludedPlacement = sharedExcludedPlacementIterator.next();
    excludedPlacements.push(sharedExcludedPlacement.getUrl());
  }
  return excludedPlacements.sort();
}
```

Die Funktionsweise ähnelt der vorausgegangen Funktion queryPlacementUrls(), allerdings ist die Datenquelle für die Stapelverarbeitung offenbar keine Abfrage, die mit dem Google Ads Query Builder gebaut wurde, sondern entstammt einer separaten Funktion getExcludedPlacementList().

Diese Hilfsfunktion haben wir uns gebaut, um sie später für die Ergänzung der Placement-Ausschlüsse wiederverwenden zu können. Schauen wir uns kurz die Funktionsweise dieser Hilfsfunktion an:

Funktion getExcludedPlacementList()

Die Funktion bedient sich der Methode excludedPlacementLists() der Google Ads API, mit deren Hilfe man Placement-Ausschlusslisten mit bestimmten Eigenschaften zur weiteren Verwendung innerhalb des Skripts abrufen kann.

In unserem Beispiel holen wir die erste gefundene Ausschlussliste ab und übergeben sie zurück an die aufrufende Funktion.

```
function getExcludedPlacementList(queryListName){
  var sharedExcludedPlacementList = AdsApp.excludedPlacementLists()
      .withCondition("Name = '" + queryListName + "'").get().next();
  return sharedExcludedPlacementList;
}
```

In der Hauptfunktion main() setzen wir nun den Namen für die zu verwendende Placement-Ausschlussliste und schreiben das Ergebnis der Funktion getExcludedplacements in die Variable “excludedPlacements”:

```
function main(){
  var placementUrls = queryPlacementUrls();
  var queryListName = 'Globale Placement-Ausschlüsse';
  var excludedPlacements = getExcludedPlacements(queryListName);
}
```

Funktion loadWhitelist()

Damit wir bestimmte Placements explizit erlauben können, auch wenn sie nicht auf eine der vordefinierten TLDs enden, haben wir uns eine Whitelist als Google Sheet angelegt, die wir manuell befüllen können. Diese Whitelist laden wir vor der weiteren Verarbeitung ebenfalls in ein Array.

Unschwer erkennbar sind die beiden Funktionsparameter, mit denen wir später die URL unseres Google Sheets und den Namen des darin enthaltenen Sheets mit der Whitelist an die Funktion übergeben.

```
function loadWhitelist(spreadsheetUrl,sheetName){
  const ss = SpreadsheetApp.openByUrl(spreadsheetUrl);
  const sheet = ss.getSheetByName(sheetName);
  const lastRow = sheet.getLastRow();
  const range = sheet.getRange(1, 1, lastRow);
  const values = range.getValues();
  var returnValues = [];
  for (i=0;i<values.length;i++){
    returnValues.push(values[i][0]);
  }
  return returnValues;
}
```

Den Ladevorgang ergänzen wir ebenfalls in der Hauptfunktion main() und definieren zuvor zwei globale Variablen mit den benötigten Werten:

```
const SPREADSHEET_URL = 'https://docs.google.com/spreadsheets/d/----------/';
const SHEET_NAME = 'placements';
```

```
function main(){
  var placementUrls = queryPlacementUrls();
  var queryListName = 'Globale Placement-Ausschlüsse';
  var excludedPlacements = getExcludedPlacements(queryListName);
  var whitelist = loadWhitelist(SPREADSHEET_URL,SHEET_NAME);
}
```

Funktion getToBeExcludedPlacements()

Im nächsten Schritt ermitteln wir die auszuschließenden Placements. Diese Funktion verarbeitet alle drei zuvor vorbereiteten Listen bzw. Arrays:

- placementUrls - alle vermeintlich unerwünschten Placements
- excludedPlacements - alle bereits ausgeschlossenen Placements
- whitelist - alle explizit gewünschten Placements

Die im Stapel “placementUrls” enthaltenen Placements werden zunächst einzeln gegen die Whitelist geprüft. Ist das jeweilige Placement darin enthalten, wird es ignoriert. Für unsere Bestätigungsmail zählen wir außerdem die in der Whitelist gefundenen Placements mit.

Wenn das Placement nicht auf der Whitelist steht, wird als nächstes geprüft, ob es bereits in der Placement-Ausschlussliste enthalten ist. Falls nicht, wird es im Rückgabe-Array “toBeExcludedPlacements” ergänzt.

Sobald alle Placements gegen die Whitelist geprüft wurden, wird das Rückgabe-Array nebst Whitelist-Zähler an die aufrufende Funktion zurückgegeben.

```
function
getToBeExcludedPlacements(placementUrls,excludedPlacements,whitelist){
  var toBeExcludedPlacements = [];
```

```
  var numWhitelisted = 0;
  for (var i=0;i<placementUrls.length;i++){
      if (whitelist.indexOf(placementUrls[i]) != -1){
        numWhitelisted++;
      } else if (excludedPlacements.indexOf(placementUrls[i]) == -1){
          toBeExcludedPlacements.push(placementUrls[i]);
      }
  }
  return [toBeExcludedPlacements.sort(),numWhitelisted];
}
```

Nach Ergänzung der neuen Funktion in der Hauptfunktion main() sieht der Kopfbereich des Skripts wie folgt aus:

```
const SPREADSHEET_URL = 'https://docs.google.com/spreadsheets/d/----------/';
const SHEET_NAME = 'placements';

function main(){
  var placementUrls = queryPlacementUrls();
  var queryListName = 'Globale Placement-Ausschlüsse';
  var excludedPlacements = getExcludedPlacements(queryListName);
  var whitelist = loadWhitelist(SPREADSHEET_URL,SHEET_NAME);
  var toBeExcludedPlacements =
getToBeExcludedPlacements(placementUrls,excludedPlacements,whitelist);
}
```

Funktion addExcludedPlacement()

Nachdem wir nun alle zu ergänzenden Placement-Ausschlüsse zusammen haben, brauchen wir sie nur noch zu der bestehenden Placement-Ausschlussliste hinzufügen. Die Funktion dazu ist wieder relativ einfach gehalten.

Sie empfängt neben dem Namen der zu verwendenden Placement-Ausschlussliste auch das Array mit den auszuschließenden Placements, ruft die Ausschlussliste auf und hängt die Placements an.

```
function addExcludedPlacement(queryListName,toBeExcludedPlacements){
  var sharedExcludedPlacementList = getExcludedPlacementList(queryListName);
  sharedExcludedPlacementList.addExcludedPlacements(toBeExcludedPlacements);
}
```

Eigentlich wäre unser Skript damit schon fertig. Der Kopfbereich sieht jetzt wie folgt aus:

```
const SPREADSHEET_URL = 'https://docs.google.com/spreadsheets/d/----------/';
const SHEET_NAME = 'placements';

function main(){
  var placementUrls = queryPlacementUrls();
  var queryListName = 'Globale Placement-Ausschlüsse';
  var excludedPlacements = getExcludedPlacements(queryListName);
  var whitelist = loadWhitelist(SPREADSHEET_URL,SHEET_NAME);
  var toBeExcludedPlacements =
getToBeExcludedPlacements(placementUrls,excludedPlacements,whitelist);
  addExcludedPlacement(queryListName,toBeExcludedPlacements[0]);
}
```

Funktion sendConfirmationEmail()

Damit wir aber auch mitbekommen, welche Placements neu ausgeschlossen wurden, lassen wir uns vom Skript eine Mail mit einer Liste der neu ausgeschlossenen URLs schicken.

Diese Funktion empfängt als Parameter ebenfalls den Namen der Ausschlussliste und das Rückgabe-Array mit der Liste der auszuschließenden Placements und dem Whitelist-Zähler.

Damit bereitet es den Output für die zu erstellende E-Mail vor. Zuerst wird die String-Variable “outputPlacements” zeilenweise mit den URLs der ausgeschlossenen Placements befüllt. Es folgt die Definition der Empfängeradresse(n) und des Betreffs, sowie des E-Mail-Inhalts. An dessen Ende wird die URL-Liste angehängt.

```
function sendConfirmationEmail(queryListName,toBeExcludedPlacements){
  var outputPlacements = "";
  for (i=0;i<toBeExcludedPlacements[0].length;i++){
      outputPlacements += toBeExcludedPlacements[0][i] + "\n";
  }
  var eMailAddress = "rs@adtraffic.de";
  var eMailSubject = "Placement Excluder // Mein Account-Name";
  var eMailContent = toBeExcludedPlacements[0].length.toLocaleString("de") +
" Placement(s) in Placement-Ausschlussliste " + queryListName + " ergänzt." +
"\n" +
      toBeExcludedPlacements[1] + " Placement(s) übersprungen (whitelisted)."
+ "\n\n" + outputPlacements;
  console.log(eMailContent);
  sendSimpleTextEmail(eMailAddress,eMailSubject,eMailContent)
}
```

Funktion sendSimpleTextEmail()

Den eigentlichen Versand der fertigen Bestätigungsmail übernimmt diese Hilfsfunktion aus der Google Ads Scripts Bibliothek.

```
function sendSimpleTextEmail(eMailAddress,eMailSubject,eMailContent) {
  MailApp.sendEmail(eMailAddress,eMailSubject,eMailContent);
  Logger.log("Mail sent.");
}
```

Und damit ist unser Skript bereit für den Einsatz. Der Kopfbereich sieht jetzt so aus:

```
const SPREADSHEET_URL = 'https://docs.google.com/spreadsheets/d/----------/';
const SHEET_NAME = 'placements';

function main(){
  var placementUrls = queryPlacementUrls();
  var queryListName = 'Globale Placement-Ausschlüsse';
  var excludedPlacements = getExcludedPlacements(queryListName);
  var whitelist = loadWhitelist(SPREADSHEET_URL,SHEET_NAME);
  var toBeExcludedPlacements =
getToBeExcludedPlacements(placementUrls,excludedPlacements,whitelist);
  addExcludedPlacement(queryListName,toBeExcludedPlacements[0]);
  sendConfirmationEmail(queryListName,toBeExcludedPlacements);
}
```

MEMO

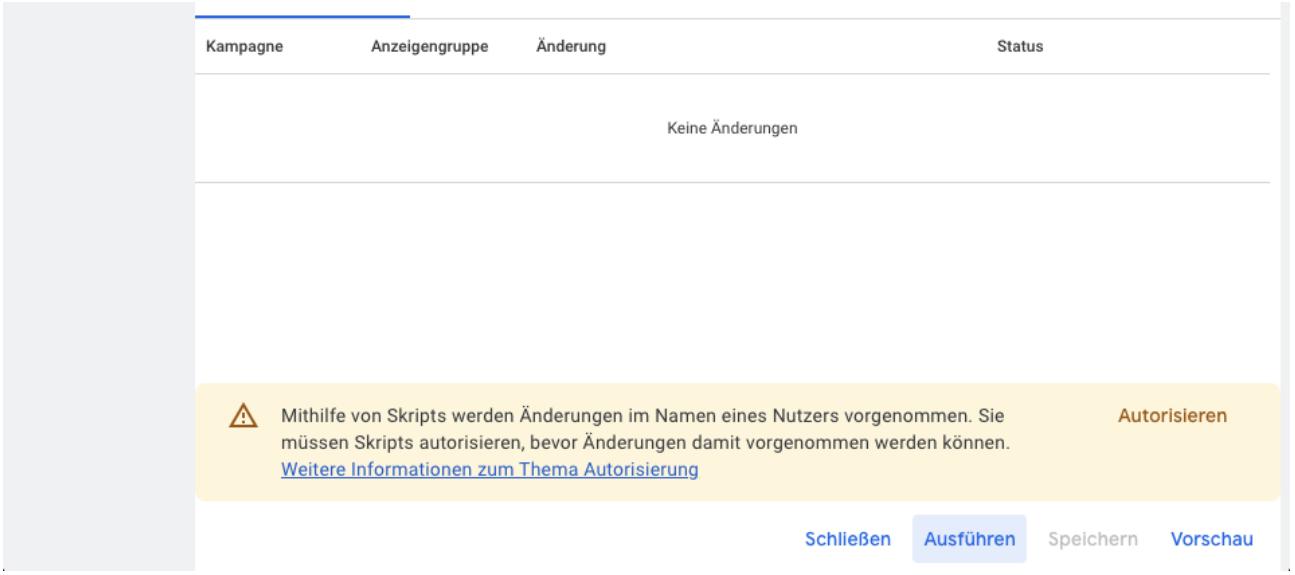
Author/License Header hinzugefügt

Der Form halber haben wir in unserer Version noch einen Info-Header hinzugefügt.

```
//            _ _              __  __ _
//   __ _  __| | |_ _ __ __ _ / _|/ _(_) ___
//  / _` |/ _` | __| '__/ _` | |_| |_| |/ __|
// | (_| | (_| | |_| | | (_| |  _|  _| | (__
//  \__,_|\__,_|\__|_|  \__,_|_| |_| |_|\___|
//
// Placement Excluder 23.1 (c) 2023 by R. Stinauer & R. Frank
//
// E: info@adtraffic.de | W: www.adtraffic.de
```

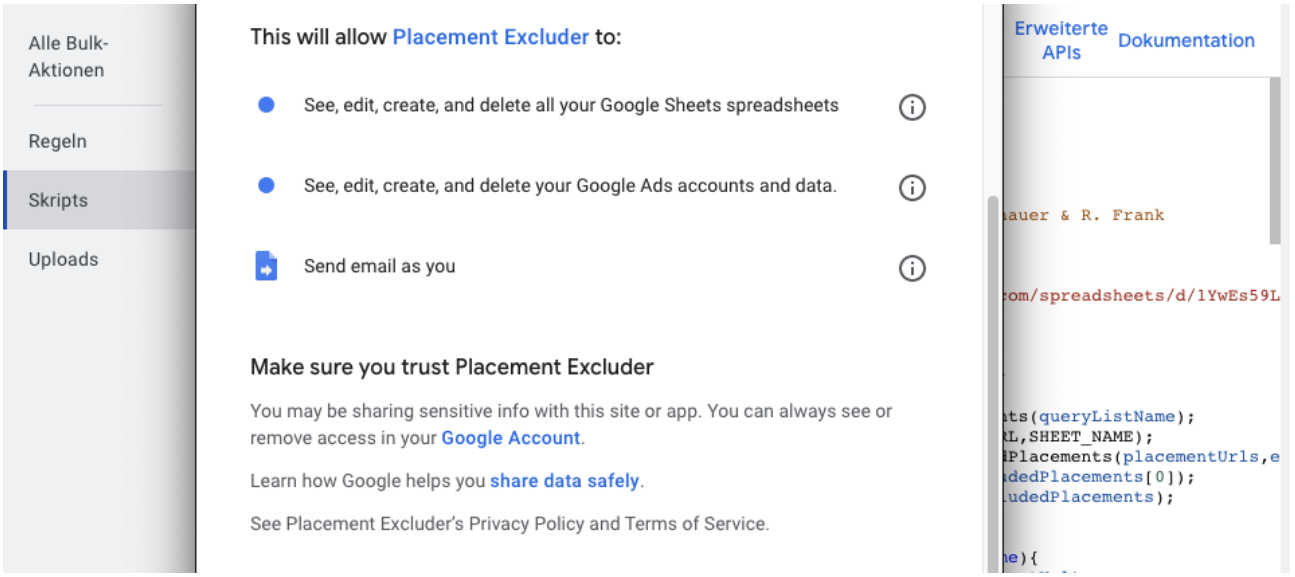
Weitere Berechtigungen

Das Script benötigt für den Zugriff auf das Google Sheet mit der Whitelist und den Versand der Bestätigungsmail noch weitere Berechtigungen, die es spätestens bei der ersten Ausführung abfragt. Die entsprechende Meldung taucht am unteren Rand auf:



MEMO

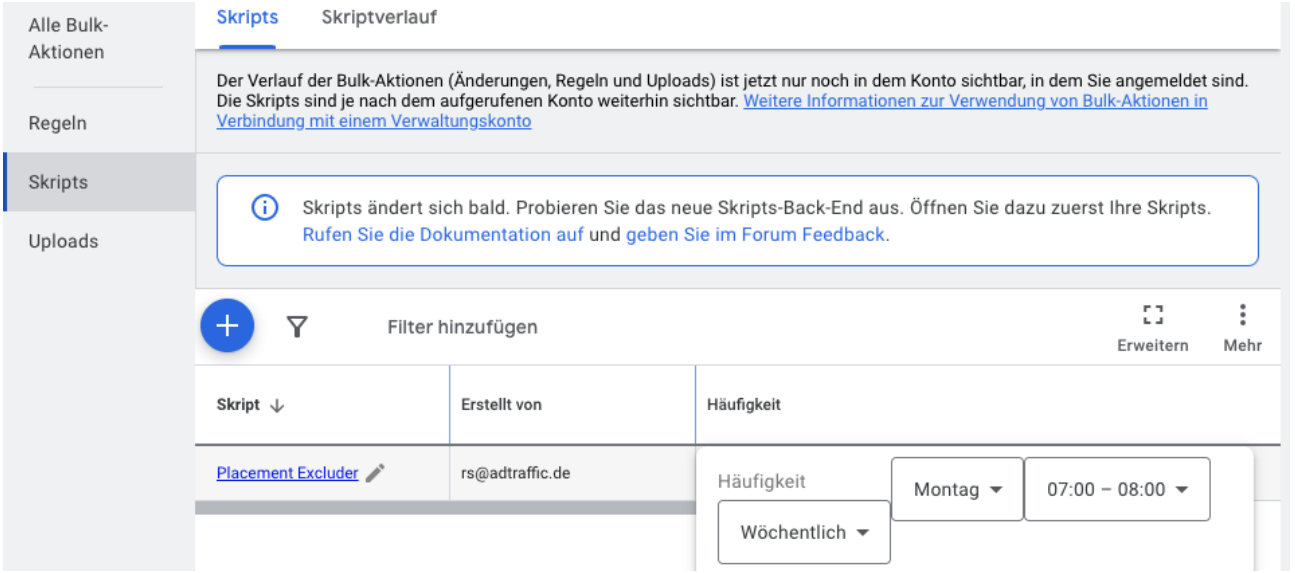
Folgende zusätzliche Berechtigungen werden benötigt:



Zeitplan

Um das fertige Skript automatisch laufen zu lassen, stellen wir auf der Übersichtsseite die Zeitplanung in der Spalte “Häufigkeit” ein.

In unserem Beispiel lassen wir das Skript jeden Montag zwischen 7 und 8 Uhr laufen.



MEMO

Zusammenfassung

Automatisierungsmöglichkeiten für Google Ads Kampagnen schlummern hinter den meisten operativen Handgriffen. Wir hoffen, Du hast anhand unseres konkreten Beispiels eine genaue(re) Vorstellung davon bekommen, wie Dich Google Ads Scripts bei Deinen täglichen Aufgaben unterstützen können. Unser Artikel hat Dich hoffentlich auch dazu ermuntert, Dich näher mit Google Ads Scripts zu beschäftigen. Vielleicht hast Du auch schon eine konkrete Idee zum Bau Deiner skript-basierten Lösung für ein individuelles Problem. Wir wünschen Dir viele Freude beim Experimentieren und viel Erfolg mit Deinem ersten Google Ads Script!

ChatGPT

Vielleicht hast Du Dich gerade gefragt: “Kann ich mir das nicht einfach von ChatGPT schreiben lassen?”

Auf die Frage “Could you create Google Ads scripts using the latest Google Ads API version?” antwortete uns das KI-Netzwerk hilfsbereit, aber zurückhaltend:

“As an AI language model, I can provide you with an overview of how Google Ads scripts could be written using the latest Google Ads API version. However, writing complete and functional scripts is beyond my capabilities as it requires knowledge of specific programming languages and expertise in Google Ads scripting.”

Tatsächlich wendet ChatGPT zur Erstellung von Code-Snippets für Google Ads Scripts teils veraltete API-Syntax an und ergänzt bei Bedarf auch gerne mal erfundene Methoden.

Lass Dir also gerne von ChatGPT helfen, aber probier es doch vorher einmal ohne KI…

Happy Scripting!