```
// Gmail2GDrive
// https://github.com/ahochsteger/gmail2gdrive

/**
 * Returns the label with the given name or creates it if not existing.
 */
function getOrCreateLabel(labelName) {
  var label = GmailApp.getUserLabelByName(labelName);
  if (label == null) {
    label = GmailApp.createLabel(labelName);
  }
  return label;
}


/**
 * Returns the GDrive folder with the given name or creates it if not existing.
 */
function getOrCreateFolder(id) {
  var folder = DriveApp.getFolderById(id);
  console.log(folder);
  if (!folder) {
    throw new Error( "folder not found." );
  }
  return folder;
}

/**
 * Processes a message
 */
function processMessage(message, rule, config) {
  Logger.log("INFO:       Processing message: "+message.getSubject() + " (" + message.getId() + ")");
  var messageDate = message.getDate();
  var attachments = message.getAttachments();
  for (var attIdx=0; attIdx<attachments.length; attIdx++) {
    var attachment = attachments[attIdx];
    Logger.log("INFO:         Processing attachment: "+attachment.getName());
    var match = true;
    if (rule.filenameFromRegexp) {
    var re = new RegExp(rule.filenameFromRegexp);
      match = (attachment.getName()).match(re);
    }
    if (!match) {
      Logger.log("INFO:           Rejecting file '" + attachment.getName() + " not matching" +
rule.filenameFromRegexp);
      continue;
    }
    try {
      var folder = getOrCreateFolder(rule.folder);
      var file = folder.createFile(attachment);
      var filename = file.getName();
      if (rule.filenameFrom && rule.filenameTo && rule.filenameFrom == file.getName()) {
        filename = Utilities.formatDate(messageDate, config.timezone,
rule.filenameTo.replace('%s',message.getSubject()));
        Logger.log("INFO:           Renaming matched file '" + file.getName() + "' -> '" + filename + "'");
        file.setName(filename);
      }
```

```
    else if (rule.filenameTo) {
      filename = Utilities.formatDate(messageDate, config.timezone,
rule.filenameTo.replace('%s',message.getSubject()));
      Logger.log("INFO:           Renaming '" + file.getName() + "' -> '" + filename + "'");
      file.setName(filename);
    }
    file.setDescription("Mail title: " + message.getSubject() + "\nMail date: " + message.getDate() + "\nMail link:
https://mail.google.com/mail/u/0/#inbox/" + message.getId());
    Utilities.sleep(config.sleepTime);
  } catch (e) {
    Logger.log(e);
  }
 }
}
/** Email Weiterleitung
*
*/
function autoForward(emailadresse, message) {

message.forward(emailadresse);

}

/**
 * Generate HTML code for one message of a thread.
 */
function processThreadToHtml(thread) {
 Logger.log("INFO:   Generating HTML code of thread '" + thread.getFirstMessageSubject() + "'");
 var messages = thread.getMessages();
 var html = "";
 for (var msgIdx=0; msgIdx<messages.length; msgIdx++) {
  var message = messages[msgIdx];
  html += "From: " + message.getFrom() + "<br />\n";
  html += "To: " + message.getTo() + "<br />\n";
  html += "Date: " + message.getDate() + "<br />\n";
  html += "Subject: " + message.getSubject() + "<br />\n";
  html += "<hr />\n";
  html += message.getBody() + "\n";
  html += "<hr />\n";
 }
 return html;
}

/**
* Generate a PDF document for the whole thread using HTML from .
 */
function processThreadToPdf(thread, rule) {
 Logger.log("INFO: Saving PDF copy of thread '" + thread.getFirstMessageSubject() + "'");
 var folder = getOrCreateFolder(rule.folder);
 var html = processThreadToHtml(thread);
 var blob = Utilities.newBlob(html, 'text/html');
 var pdf = folder.createFile(blob.getAs('application/pdf')).setName(thread.getFirstMessageSubject() + ".pdf");
 return pdf;
}

/**
 * Main function that processes Gmail attachments and stores them in Google Drive.
```

```
 * Use this as trigger function for periodic execution.
 */
function Gmail2GDrive() {
  if (!GmailApp) return; // Skip script execution if GMail is currently not available (yes this happens from time to
time and triggers spam emails!)
  var config = getGmail2GDriveConfig();
  var label = getOrCreateLabel(config.processedLabel);
  var end, start, runTime;
  start = new Date(); // Start timer

  Logger.log("INFO: Starting mail attachment processing.");
  if (config.globalFilter===undefined) {
    config.globalFilter = "has:attachment -in:trash -in:drafts -in:spam";
  }

  // Iterate over all rules:
  for (var ruleIdx=0; ruleIdx<config.rules.length; ruleIdx++) {
    var rule = config.rules[ruleIdx];
    var gSearchExp  = config.globalFilter + " " + rule.filter + " -label:" + config.processedLabel;
    if (config.newerThan != "") {
      gSearchExp += " newer_than:" + config.newerThan;
    }
    var doArchive = rule.archive == true;
    var doPDF = rule.saveThreadPDF == true;
    var skipMail = rule.noEmail == true;
    var emailadresse = rule.email;

    // Process all threads matching the search expression:
    var threads = GmailApp.search(gSearchExp);
    Logger.log("INFO:   Processing rule: "+gSearchExp);
    for (var threadIdx=0; threadIdx<threads.length; threadIdx++) {
      var thread = threads[threadIdx];
      end = new Date();
      runTime = (end.getTime() - start.getTime())/1000;
      Logger.log("INFO:     Processing thread: "+thread.getFirstMessageSubject() + " (runtime: " + runTime + "s/" +
config.maxRuntime + "s)");
      if (runTime >= config.maxRuntime) {
        Logger.log("WARNING: Self terminating script after " + runTime + "s");
        return;
      }

      // Process all messages of a thread:
      var messages = thread.getMessages();
      for (var msgIdx=0; msgIdx<messages.length; msgIdx++) {
        var message = messages[msgIdx];
        processMessage(message, rule, config);
        if (skipMail) {

      }
      else
      {  autoForward(emailadresse, message);}
      }
      //Email weiterleiten


      if (doPDF) { // Generate a PDF document of a thread:
        processThreadToPdf(thread, rule);
```

```
      }

      // Mark a thread as processed:
      thread.addLabel(label);

      if (doArchive) { // Archive a thread if required
        Logger.log("INFO:     Archiving thread '" + thread.getFirstMessageSubject() + "' ...");
        thread.moveToArchive();
      }
    }
  }
  end = new Date(); // Stop timer
  runTime = (end.getTime() - start.getTime())/1000;
  Logger.log("INFO: Finished mail attachment processing after " + runTime + "s");
}
```