connecting your business

Integration Google Calendar

LANCOM Wireless ePaper Server

LANCOM Systems

Contents

1 Objectives and requirements

1.1 Objectives

This document aims to show you how to successfully integrate the LANCOM Wireless ePaper solutions into an environment that makes use of the Google Calendar. This is done by means of a Java program, the corresponding configuration file and an XSL template.

The Java program, the configuration files and the template are examples, which are designed to make it easy for you to create your own files. If the code is to be used productively, you need to amend some information in the configuration file accordingly. Further templates can be created or used to achieve an optimal adjustment to the required operations. The Java program can be newly created or rewritten if the existing version does not meet your requirements.

1.2 Requirements

To successfully combine LANCOM Wireless ePaper solutions with a Google Calendar, the available software needs to meet certain requirements and probably requires some adaptation.

1. LANCOM Wireless ePaper Server

   The following requirements must be met by the LANCOM Wireless ePaper Server:

   - The template must be stored under `\\<Installation-directory>\data\templates\`.
   - Images referenced from the template must be stored under `\\<Installation-directory>\data\images\`.
   - Each ePaper Display must be given a tag, which contains the name of the room. Use uppercase letters only. If there are spaces in the room name (calendar names), these must be replaced by underscore characters. Example: If the room name in Google Calendar is `Room AACHEN`, the corresponding tag is `ROOM_AACHEN`.

2. Google Developers Console & Google Calendar

   The following conditions must be met in the Google Developers Console and the Google Calendar of the associated Google account:

   - In the Google Developers Console, create a project for the Wireless ePaper management (see *Chapter 2.1*), in which the necessary API is enabled (see *Chapter 2.2*).
   - For the project created In the Google Developers Console, create a Service Account for the Wireless ePaper management. The access credentials for this account are used by the Java program to access the Google Calendar (see *Chapter 2.3*).
   - In Google Calendar, the service account set up for each calendar needs to have read access (see *Chapter 2.4*).

3. Java program: WePService

   The following requirements must be met for the use of the Java program WePService:

   - A system with the Java JDK (version 8 or higher) to run the Java program must be available on the network.

     openJDK is not supported.

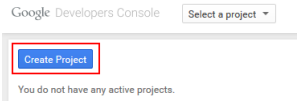
   - The configuration files need to be adapted to the requirements of the intended use.

2 Google Developers Console & Google Calendar

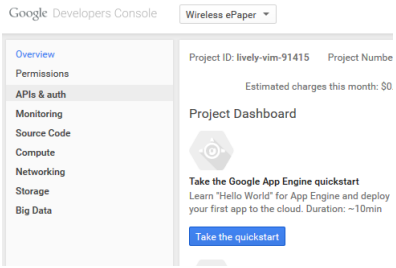
The first step is to create a project in the Google Developers Console, which is used to configure the APIs and the access data. To do this, access the web page *https://console.developers.google.com* in your browser. Log on with the access data of the Google account that you are using for calendar management.

2.1 Creating a project

A new project is created by clicking the **Create Project** button. You can choose the name of the project freely.

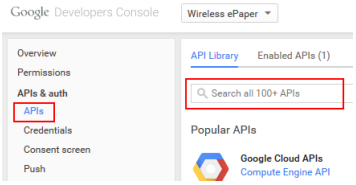


Click on the project name in the browser to switch to the project dashboard. The menu item **APIs & auth** displays the configuration items for defining the APIs and creating service accounts.

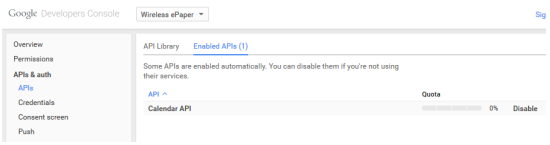


2.2 Defining APIs

Under **APIs** > **API Library**, enter **Calendar API** as a search term and enable the API of that name.

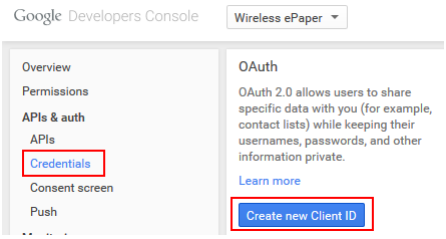


In the menu item **APIs** > **Enabled APIs**, disable all of the APIs except for the **Calendar API**.



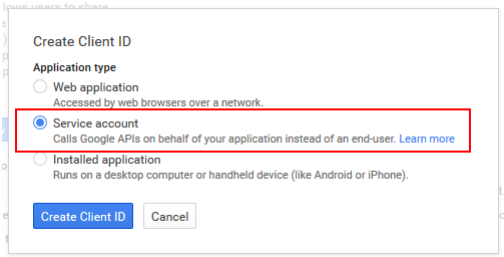
2.3 Setting up a service account for calendar access

The menu item **Credentials** displays the accounts you created already, and you can create new ones here. Click on the button **Create new Client ID** to create a new account.

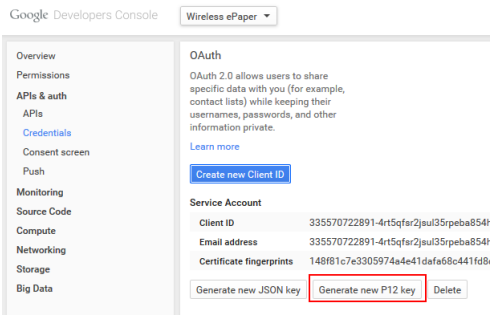


Select the option **Service account** and confirm the selection by clicking the button **Create Client ID**. This automatically starts the download of a key in the JSON format.

As this key is not required, you can cancel the download.



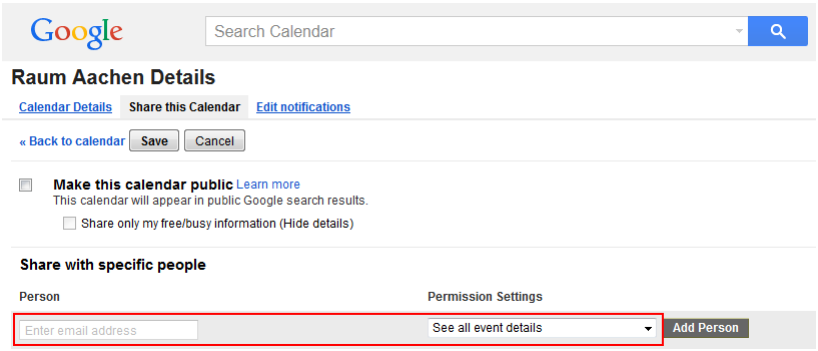
The service account consists of the client ID, e-mail address and public-key fingerprints. Create a new PKCS12 file by clicking on the button **Create new P12 key**. This starts the download of the PKCS12 file. It is used by the Java program WePService to authenticate itself to the Google Calendar. You then delete the public-key fingerprint that was available before you created the new one. This is linked to the auto-generated keys in JSON format and is not required.



2.4 Configuring Google Calendar

The service account must be linked to the individual calendars of rooms in Google Calendar. To do this, open *https://www.google.com/calendar* to access Google Calendar with your browser. Log on with the access data of the Google account that you are using for calendar management. Enter the e-mail address of the service account for the

respective calendar under **Calendar settings** > **Share this Calendar**. Set the Permission Settings to **See all event details**.



You will find the e-mail address of the service account listed in the Google Developers Console in the corresponding project under **APIs and authentication** > **Credentials** > **0Auth** > **Service account**.

3 Java program WePService

3.1 Installation

All you have to do is store the folder "WePService" on the system where it is to run. The PKCS12 file that was downloaded when you created the service account (see *Chapter 2.3*) must be copied into this folder and renamed as `calendar.p12`.

3.2 Contents of the WePService folder

File	Purpose
WePService.jar	The Java program for querying calendar entries and the configuration file for calendar properties.
calendar.properties	Configuration file for the parameters of Google Calendar.
client.properties	Configuration file for the parameters of the LANCOM Wireless ePaper Server.
service.properties	Configuration file for the template and the update interval.
run.bat	Batch file to start the Java program under Windows.
run.sh	Shell file to start the Java program under Linux.
log4j2.xml	XML file as the basis for creating the log.
calendar.p12	Key for the service account with access to Google Calendar. The key is created in *Chapter 2.3* and is not part of the delivery.

3.3 WePService.jar

This is the Java program that accesses Google Calendar. The program retrieves the information from each calendar and sends any necessary updates to the LANCOM Wireless ePaper Server. No detailed explanation of the code is provided here, because all of the changes necessary for use are made to the configuration files.

3.4 calendar.properties

The `calendar.properties` file is a configuration file containing the parameters regarding necessary for Google Calendar.

```
# the used timezone when computing dates
time_zone=Europe/Berlin
# the number of events to be read from each calendar
number_events=2
# the google service account id used for fetching calendar data
serviceaccountid=
```

```
# list of resources (rooms) that will be read

# Room 0
resource0=
# Room 1
resource1=
# Room 2
resource2=
# Room 3
resource3=
# Room 4
resource4=
```

Parameter description:

Parameter	Description
`time_zone`	Sets the time zone. **Default:** Europe/Berlin
`number_events`	Sets the number of queried events per calendar. **Default:** 2
`serviceaccountid`	Contains the e-mail address of the service account created in *Chapter 2.3* and used to query the calendar information. Place the corresponding P12-key (PKCS12 file) in the folder "WePService" and rename the file to `calendar.p12`.
`resource [x]`	Contains the calendar address for room x, the information for which is to be queried. The room numbering starts at 0. The calendar address can be viewed in Google Calendar under **Calendar settings** > **Calendar details** > **Calendar address**.
`run.bat`	Batch file for starting the Java program under Windows.
`run.sh`	Shell file to start the Java program under Linux.
`log4j2.xml`	XML file as the basis for creating the log.
`calendar.p12`	Key for the service account with access to Google Calendar. The key is created in *Chapter 2.3* and is not part of the delivery.

3.5 client.properties

The file `client.properties` is a configuration file containing the parameters necessary for communicating with the LANCOM Wireless ePaper Server.

```
# the WeP-Server's hostname or ip address
host = 192.168.1.1
# the WeP-Server's port
port = 8001
# the WeP-Server's username
user = admin
# the WeP-Server's password
password = admin
# the annotation the client send its data (XML or JSON - XML is recommended)
annotation = XML
```

Parameter description:

Parameter	Description
`host`	Contains the IPv4 address of the LANCOM Wireless ePaper Server.
`port`	Contains the port used to communicate with the LANCOM Wireless ePaper Server.
	Default: 8001 (We recommend that you keep the default setting.)
`user`	Contains the user name for the communication with the LANCOM Wireless ePaper Server.
	Default: admin (We recommend that you keep the default setting.)
`password`	Contains the password for the `user` specified above.
	Default: admin (We recommend that you keep the default setting.)
`annotation`	Contains the format of the communication with the LANCOM Wireless ePaper Server.
	Default: XML (We recommend that you keep the default setting.)

3.6 service.properties

The file `service.properties` is a configuration file which defines additional parameters.

```
# defines the xsl template file on the WeP server used to generate the image on the WeP
template = lcsconference_landscape.xsl

# display text aliases for today and tomorrow instead of numeric dates
use.date.alias = false

# the update interval of the server in milliseconds
# 30 sec.
#update.interval = 30000
# 1 min
#update.interval = 60000
# 5 mins
update.interval = 300000
```

Parameter description:

Parameter	Description
`template`	Sets the template used by the Wireless ePaper Server to create the images for the Wireless ePaper Displays. You need to store the template on the Wireless ePaper Server under `\\<Installation-directory>\data\templates\`.
	Default: `lcsconference_landscape.xsl` (We recommend that you keep the default setting.)
`use.date.alias`	This setting controls whether a text alias (today, tomorrow) should be used instead of the numeric date.
	Default: `false` (The numeric date is displayed.)
`update.interval`	Specifies the interval in milliseconds in which the Java program queries the information on Google Calendar.

3.7 run.bat

The file `run.bat` is a batch file, which is used to start the Java program "WePService" under Windows. The program queries the calendar information on Google Calendar (see *Chapter 3.3*).

3.8 run.sh

The file `run.sh` is a shell file, which is used to start the Java program "WePService" under Linux. The program queries the calendar information on Google Calendar (see *Chapter 3.3*).

Note that the file delivered to you will not run without being configured first.

4 Template for room signage

The template is stored on the Wireless ePaper Server as an XSL file. This specifies the format used by the Wireless ePaper Server to render the updates send by the Java program "WePService". The resulting image is then delivered to the corresponding Wireless ePaper Displays.

4.1 Example of a template

This XSL template is based on a potential meeting room signage concept for the LANCOM Systems GmbH.

Line numbering is not a part of the code and is for purposes of clarity only.

```
001 <?xml version="1.0" encoding="UTF-8"?>
002 <!—The template is provided for 7.4" Displays mounted in the landscape orientation.-->
003 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
004
005 <xsl:template match="Record">
006
007  <!-- Rendering information for 7.4” Displays -->
008  <image height="480" width="800" rotation="90" font-family="Verdana">
009
010   <!-- Room -->
011   <field height="108" width="780" x="10" y="20">
012    <text align="center" font-size="40" font-weight="bold">
013     <utils method="toUpperCase">
014      <xsl:value-of select="room/@roomName"/>
015     </utils>
016    </text>
017
018     <!-- Date -->
019    <text align="center" font-size="35" font-weight="bold" padding-top="10">
020     <xsl:value-of select="room/field[@key='date']/@value"/>
021    </text>
022   </field>
023
024   <line thickness="2" x-from="0" x-to="800" y-from="130" y-to="130"/>
025
026   <!-- Time1 -->
027   <field height="50" width="780" x="20" y="150">
028    <text align="left" font-weight="bold" font-size="40">
029     <xsl:value-of select="room/field[@key='time1']/@value"/>
030    </text>
031   </field>
032
033   <!-- Purpose1 -->
034   <field height="50" width="780" x="20" y="200">
035    <text align="left" font-size="40" condense="1, 0.8, 0.6, 0.5">
036     <xsl:value-of select="room/field[@key='purpose1']/@value"/>
037    </text>
038   </field>
039
040   <!-- Chair1 -->
041   <field height="40" width="770" x="20" y="250">
042    <text align="left" font-weight="bold" font-size="30">
043     <xsl:value-of select="room/field[@key='chair1']/@value"/>
```

```
044    </text>
045   </field>
046
047    <!-- Time2 -->
048   <field height="35" width="780" x="20" y="320">
049    <text align="left" font-size="28">
050     <xsl:value-of select="room/field[@key='time2']/@value"/>
051    </text>
052   </field>
053
054   <!-- Purpose2 -->
055   <field height="35" width="780" x="20" y="355">
056    <text align="left" font-size="28" condense="1, 0.8, 0.6, 0.5">
057     <xsl:value-of select="room/field[@key='purpose2']/@value"/>
058    </text>
059   </field>
060
061    <!-- Chair2 -->
062   <field height="30" width="770" x="20" y="390">
063    <text align="left" font-size="20">
064     <xsl:value-of select="room/field[@key='chair2']/@value"/>
065    </text>
066   </field>
067
068   <!-- LANCOM Logo -->
069   <field align="right" height="60" width="780" x="10" y="410">
070    <img src="../images/LANCOM_Logo_200.png"></img>
071   </field>
072  </image>
073 </xsl:template>
074 </xsl:stylesheet>
```

4.2 Explanations of individual sections of code

This chapter explains the various sections of code in the example XSL template.

Display information

```
007   <!-- Rendering information for 7.4” Displays -->
008   <image height="480" width="800" rotation="90" font-family="Verdana">
```

This section specifies which Wireless ePaper Display the following code applies to. In this case, it is a 7.4" Display with a resolution of 800 x 480 pixels.

`height`	Specifies the height of the image in pixels.
`width`	Specifies the width of the image in pixels.
`rotation`	Specifies the rotation of the image in degrees. 0 corresponds to vertically mounted Displays and 90 to horizontally mounted Displays.
`font-family`	Specifies the font to be used.

Definition of text fields

```
010   <!-- Room -->
011   <field height="108" width="780" x="10" y="20">
012    <text align="center" font-size="40" font-weight="bold">
013     <utils method="toUpperCase">
014      <xsl:value-of select="room/@roomName"/>
015     </utils>
```

```
016    </text>
017
018    <!-- Date -->
019    <text align="center" font-size="35" font-weight="bold" padding-top="10">
020     <xsl:value-of select="room/field[@key='date']/@value"/>
021    </text>
022   </field>
```

This area specifies the position of the room name and date.

First, a field is defined that specifies the height, width and position on the Display:

```
<field height="108" width="780" x="10" y="20">.
```

`height`	Specifies the height of the field in pixels.
`width`	Specifies the width of the field in pixels.
`x`	Specifies the distance from the left edge of the Display in pixels.
`y`	Specifies the distance from the top edge of the Display in pixels.

The layout of the various texts is defined in this field:

```
<text align="center" font-size="40" font-weight="bold">
```

`align`	Specifies how the text is justified. (e.g., center, left, right).
`font-size`	Specifies the font size.
`font-weight`	Specifies the font style (e.g., bold, italic).
`y`	Specifies the distance from the top edge of the Display in pixels.

```
<utils method="toUpperCase">
```

Invoking this method forces the text to appear in the uppercase.

```
<xsl:value-of select="room/@roomName"/>
```

Here, the XML information provided by the script library is accessed and the text is inserted at the appropriate position. So far, the definitions have only set out the layout.

The next step uses the same principle to specify the text that shows the transmitted date. Additional fields contain the layout for the rest of the information that relates to the individual meetings.

Integration of graphics

```
068    <!-- LANCOM Logo -->
069    <field align="right" height="60" width="780" x="10" y="410">
070     <img src="../images/LANCOM_Logo_200.png"></img>
071    </field>
```

In addition to text fields, images can be integrated too. Similar to the above, a field is specified that sets the position of the image.

```
<img src="../images/LANCOM_Logo_200.png" ></img>
```

The storage location and the name of the image file is specified here.

4.3 Interaction between template, XML information, and Wireless ePaper Displays

This chapter uses an example to describe the interaction between the template, XML information and the actual representation on the Wireless ePaper Display.

Step 1:

If the Java program "WePService" triggers an update, the Wireless ePaper Server receives XML information with the following content:

Line numbering is not a part of the code and is for purposes of clarity only.

```
001 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
002 <TaskOrder title="Refresh D1001BF6 for Aachen B2.100">
003    <TemplateTask labelId="D1001BF6" externalId="4711"
template="lcsconference_landscape.xsl">
004        <room roomName="Aachen B2.100">
005            <field key="date" value="16.09.2014"/>
006            <field key="time1" value="10:00 - 11:30"/>
007            <field key="purpose1" value="Marketing Project Coordination"/>
008            <field key="chair1" value="Alec Coad"/>
009            <field key="time2" value="11:30 - 13.00"/>
010            <field key="purpose2" value="Team Meeting Controlling"/>
011            <field key="chair2" value="Emily Kirby"/>
012        </room>
013    </TemplateTask>
014 </TaskOrder>
```

Step 2:

The Wireless ePaper Server processes the received information line by line. This sets the relevant Display and the template required. Based on the template, the Server now creates the data set used to render the image. The layout is set according to the template and then the information from the XML is inserted.

Two content elements are described here for illustrative purposes: The chairperson of the next meeting and the company logo.

The chairperson for the subsequent meeting is formatted according to the template as follows:

```
061   <!-- Chair2 -->
062   <field height="30" width="770" x="20" y="390">
063    <text align="left" font-size="20">
064     <xsl:value-of select="room/field[@key='chair2']/@value"/>
065    </text>
066   </field>
```

The information necessary for the representation can be seen in the XML below:

```
011    <field key="chair2" value="Emily Kirby"/>
```

Access to the image specified in the template does not require access to the XML information. The path is evaluated instead.

```
068    <!-- LANCOM Logo -->
069    <field align="right" height="60" width="780" x="10" y="410">
070     <img src="../images/LANCOM_Logo_200.png"></img>
071    </field>
```

Step 3:

The image is rendered and sent to the Wireless ePaper Display. The Display appears as shown below. The chairperson of the subsequent meeting **(1)** and the image **(2)** are highlighted in color.

5 Source code of the Java program WePService

See the source code for the Java program "WePService" is contained by the ZIP file in the subfolder "SourceCode".