18 Newsletters 164
  18.1 Create lists ......................................................... 164
  18.2 Compose ............................................................. 164
  18.3 Templates ........................................................... 164
19 Document templates 169
20 Files 171
  20.1 Assistant .......................................................... 171
  20.2 LibreOffice Online ................................................ 171
21 Notes 173
22 Tickets 175
  22.1 Types ............................................................... 176
  22.2 Statuses ............................................................. 178
  22.3 E-mail templates .................................................. 178
  22.4 Options .............................................................. 180
  22.5 Rates ................................................................. 181
  22.6 External ticket page ............................................... 182
  22.7 Example customer ticket page ................................... 182
23 Projects 185
  23.1 Administration ..................................................... 185
  23.2 Reports ............................................................. 191
  23.3 Time tracking ....................................................... 191
  23.4 Jobs ................................................................. 193
  23.5 Hours approval ..................................................... 193
  23.6 Billing ............................................................... 193
24 Holidays 195
  24.1 Working week ....................................................... 195
  24.2 Manager ............................................................. 197
25 Billing 199
  25.1 Languages .......................................................... 199
  25.2 Book properties .................................................... 199
  25.3 ODF Templates ..................................................... 202
  25.4 Statuses ............................................................. 203
  25.5 Cost codes .......................................................... 205
  25.6 Purchase order books .............................................. 205
  25.7 Custom fields ....................................................... 206
26 File search 207
  26.1 Indexing ............................................................. 207
27 Rocket Chat 209
28 Studio 213
  28.1 Prerequisites ....................................................... 213
  28.2 Installation ........................................................ 214
  28.3 Workflow .......................................................... 214
  28.4 The Generated Screens .......................................... 221
  28.5 Deleting a Generated Module ................................... 221
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 Translations</td>
<td>225</td>
</tr>
<tr>
<td>29.1 Contribute</td>
<td>226</td>
</tr>
<tr>
<td>29.2 Customize</td>
<td>226</td>
</tr>
<tr>
<td>30 Developer</td>
<td>227</td>
</tr>
<tr>
<td>30.1 Building a server module</td>
<td>227</td>
</tr>
<tr>
<td>30.2 Building a webclient module</td>
<td>243</td>
</tr>
<tr>
<td>30.3 Building a webclient module, part 2</td>
<td>276</td>
</tr>
<tr>
<td>30.4 Server API</td>
<td>287</td>
</tr>
<tr>
<td>30.5 Webclient API</td>
<td>304</td>
</tr>
<tr>
<td>30.6 JMAP API</td>
<td>309</td>
</tr>
</tbody>
</table>
Group-Office is an open-source groupware and CRM application. It’s our mission to simplify your daily office tasks. Easy to use modules such as the Calendar, Address Book, Projects and time tracking come by default. But it’s also possible to customize Group-Office with your own custom modules.

Visit https://www.group-office.com for more information.

Fig. 1: The Group Office start page
This section is for system administrators that want to install Group-Office on a self hosted server. There are various ways to install Group-Office. We recommend to install our Debian packages on the latest stable Debian Linux OS.

1.1 Installation on Debian or Ubuntu

1.1.1 Core system

Our preferred way of installing is using our Debian packages:

1. On Debian login as the root server and on Ubuntu become root by running:

   ```
   sudo -s
   ```

2. Only on Ubuntu make sure the “universe” repository is enabled:

   ```
   sudo add-apt-repository universe
   ```

3. First add our repository to the package management system.
   If you run PHP 7.1 or higher (Debian 10+, Ubuntu 18.04+) add:

   ```
   echo "deb http://repo.group-office.com/ 64-php-71 main" > /etc/apt/sources.list.d/groupoffice.list
   ```

   If you run PHP 7.0 (Debian 9) run:

   ```
   echo "deb http://repo.group-office.com/ 64-php-70 main" > /etc/apt/sources.list.d/groupoffice.list
   ```

4. Make sure “dirmngr” is installed for adding the public key:
apt-get install dirmngr

5. Add our public key:

```
apt-key adv --recv-keys --keyserver pool.sks-keyservers.net 0758838B
```

6. Update APT:

```
apt-get update
```

7. Then install Group-Office by running:

```
apt-get install groupoffice
```

8. Optionally you can install php-apcu for better performance of the cache:

```
apt-get install php-apcu
```

9. If you purchased Group-Office Professional licenses then make sure the Ioncube loader is installed and place the license files in “/usr/share/groupoffice/”. For example “/usr/share/groupoffice/groupoffice-pro-6.3-license.txt”. You might want to use our script: https://github.com/Intermesh/groupoffice/blob/6.4.x/scripts/install-ioncube.sh

10. Then visit http://yourserver/groupoffice and the installer should appear:

![Group-Office installer](image)

11. Follow the instructions on screen and enjoy Group-Office!
1.1.2 Mailserver

You can also use Group-Office as a complete e-mail platform. It’s based on:

1. Postfix
2. Dovecot
3. Group-Office module to manage mailboxes in the database

At the moment this is only possible with the Debian / Ubuntu packages.

When Group-Office is already installed you can run:

```
apt-get install groupoffice-mailserver
```

When this command is finished login to Group-Office as admin and install the “E-mail domains” module. In this module you can manage the domains, mailboxes and aliases.

![Manage e-mail domains in Group-Office](image)

Serverclient module

The server client allows you to:

1. Create mailboxes when you create a new user
2. Synchronize mailbox passwords when you set a new Group-Office password.

Install the module at Start menu -> modules

Then edit `/etc/groupoffice/globalconfig.inc.php` or create it if it doesn’t exist:
<?php
$config = [
    // GO will connect to this installation to add a mailbox. It is the full url to the Group-Office installation with the postfixadmin module installed.
    'serverclient_server_url' => 'http://localhost/groupoffice/',
    // A token to authenticate. The token has to be identical on the web and mail server. By default they are the same server so you can just set anything here.
    'serverclient_token' => 'someSecureTokenOfyourChoice',

    // Comma separated list of mailbox domains
    'serverclient_domains' => 'intermeshdev.nl',

    // The email account properties that will be added for the user
    'serverclient_mbroot' => '',
    'serverclient_use_ssl' => false,
    'serverclient_use_tls' => false,
    'serverclient_novalidate_cert' => '0',
    'serverclient_host' => 'localhost',
    'serverclient_port' => 143,
    'serverclient_smtp_host' => 'localhost',
    'serverclient_smtp_port' => 25,
    'serverclient_smtp_encryption' =>'',
    'serverclient_smtp_username' => '',
    'serverclient_smtp_password' => ''
];

Now when you create a new user you have the option to create:

<username>@intermeshdev.nl

And when you set your password this account will be updated too.

Fig. 3: Option to create mailbox when creating new users
TLS / SSL

It’s required to install SSL certificates for your mailserver to operate properly. So obtain an SSL certificate and take these steps:

1. Configure Dovecot IMAP in file /etc/dovecot/conf.d/10-ssl.conf:

   ```
   ssl = yes
   ssl_cert = </etc/letsencrypt/live/YOURHOSTNAME/fullchain.pem
   ssl_key = </etc/letsencrypt/live/YOURHOSTNAME/privkey.pem
   ```

2. Restart dovecot:

   ```
   invoke-rc.d dovecot restart
   ```

3. You can verify the SSL certificate with this command:

   ```
   printf 'quit\n' | openssl s_client -connect YOURHOSTNAME:143 -starttls imap | openssl x509 -dates -noout
   ```

4. Configure Postfix SMTP with these commands:

   ```
   postconf -e 'smtpd_tls_cert_file =/etc/letsencrypt/live/YOURHOSTNAME/fullchain.pem'
   postconf -e 'smtpd_tls_key_file = /etc/letsencrypt/live/YOURHOSTNAME/privkey.pem'
   ```

5. Restart postfix:

   ```
   invoke-rc.d postfix restart
   ```

6. You can verify the SSL certificate with this command:

   ```
   printf 'quit\n' | openssl s_client -connect YOURHOSTNAME:25 -starttls smtp | openssl x509 -dates -noout
   ```

External IMAP access

By default only local connections are allowed. This means only Group-Office can connect. This is very secure but in some cases you want to allow IMAP access from the outside. You’ll have to configure your firewall or router to allow connections to the server on the necessary ports:

- IMAP: 143
- IMAPS: 993

You’ll also need to uncomment following line in /etc/dovecot/conf.d/99-groupoffice.conf:

```
listen = *
```

Now connect with:

IMAP host: YOURHOSTNAME TLS encryption enabled (Make sure you’ve setup SSL) Username: full email address

External SMTP access

1.1. Installation on Debian or Ubuntu
Note: We recommend to install fail2ban too because spammers will try to abuse your server when you enable SMTP!

1. Configure Postfix SMTP with these commands:
   ```bash
   postconf -e 'smtpd_sasl_auth_enable = yes'
   ```

2. Restart postfix:
   ```bash
   invoke-rc.d postfix restart
   ```

### Anti spam / virus

The package above installs the bare minimum so you can be free to configure your system in your own way. But for your convenience we’ve also prepared an anti spam and anti virus solution based on:

1. rspamd
2. clamav

To install take these steps:

1. Add the rspamd repository because the official Debian repositories contain outdated versions:
   ```bash
   apt-get install -y lsb-release wget # optional
   CODENAME=`lsb_release -c -s`
   wget -O- https://rspamd.com/apt-stable/gpg.key | apt-key add -
   echo "deb [arch=amd64] http://rspamd.com/apt-stable/ $CODENAME main" > /etc/apt/
   →sources.list.d/rspamd.list
   echo "deb-src [arch=amd64] http://rspamd.com/apt-stable/ $CODENAME main" >> /etc/
   →apt/sources.list.d/rspamd.list
   ```

2. Update APT:
   ```bash
   apt-get update
   ```

3. Install groupoffice-mailserver-antispam:
   ```bash
   apt-get install groupoffice-mailserver-antispam
   ```

4. Run the rspamd config wizard:
   ```bash
   rspamadm configwizard
   ```

5. Test if the spam filter works by sending a GTUBE message
6. Test if the anti virus works by sending an EICAR test file
7. Checkout the rspamd Web GUI at http://yourserver/rspamd/

### 1.1.3 Multi Instance

It’s possible to host multiple instances of Group Office on one server. After installing Group Office via the Debian packages or Docker you do the following to enable it:

1. Make sure the main install database user has permissions to create databases by running the following SQL:
2. Create “multi_instance” config folder:

```bash
mkdir /etc/groupoffice/multi_instance && chown www-data:www-data /etc/groupoffice/multi_instance
```

3. Create “multi_instance” data folder:

```bash
mkdir /var/lib/groupoffice/multi_instance && chown www-data:www-data /var/lib/groupoffice/multi_instance
```

4. Login as administrator into the main Group Office instance that will manage the other instances and install the “Multi Instance” module from the “Community” package.

### 1.1.4 Documents

If you purchased the documents package you probably want to install some additional tools required for indexing file contents:

```bash
apt-get install catdoc unzip tar imagemagick tesseract-ocr tesseract-ocr-eng poppler-utils exiv2
```

These tools provide support for:

- Microsoft Office
- Images
- PDF documents

### 1.1.5 Database credentials

The mailserver connects to the “groupoffice” database to lookup mailboxes, aliases and domains. If you need to change the “groupoffice” database password, username or name. Then you also need to change the login details in these files:

- `/etc/dovecot/dovecot-groupoffice-sql.conf.ext`
- `/etc/postfix/mysql_virtual_mailbox_maps.cf`
- `/etc/postfix/mysql_virtual_mailbox_domains.cf`
- `/etc/postfix/mysql_virtual_alias_maps.cf`

Afterwards restart postfix and dovecot:

```
systemctl restart postfix
dsystemctl restart dovecot
```
1.2 Docker

1.2.1 Core System

It’s very easy to install Group-Office with docker. Please follow the instructions on our docker-groupoffice github repository:
https://github.com/Intermesh/docker-groupoffice

If you’d like to setup a developer environment then you should use:
https://github.com/Intermesh/docker-groupoffice-development

1.2.2 Multi Instance

It’s possible to host multiple instances of Group Office on one server. After installing Group Office via the Debian packages or Docker you do the following to enable it:

1. Make sure the main install database user has permissions to create databases by running the following SQL:

```sql
GRANT ALL PRIVILEGES ON *.* TO 'groupoffice'@'%' REQUIRE NONE WITH GRANT OPTION;
```

2. Login as administrator into the main Group Office instance that will manage the other instances and install the “Multi Instance” module from the “Community” package.

1.3 Tarball

We strongly recommend that you use our Debian packages or Docker over this method. But if you really want to use the Tarball source then here is how.

1.3.1 System requirements

You need a Linux server with:

<table>
<thead>
<tr>
<th>Type</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Linux / Docker</td>
</tr>
<tr>
<td>Webserver</td>
<td>Apache 2+</td>
</tr>
<tr>
<td>Database</td>
<td>MySQL 5.7+ / MariaDB 10.0.5</td>
</tr>
<tr>
<td>Programming language</td>
<td>PHP 7.0+</td>
</tr>
</tbody>
</table>

Required PHP Extensions

- pcre
- mbstring
- ctype
- date
• iconv
• curl
• zip
• soap
• gd
• pdo
• pdo_mysql (With mysqlnd, nd_pdo_mysql)
• mysqlnd
• calendar

Recommended:
• acpu (for better caching performance)
• ioncube (For professional version with Intermesh support)

### 1.3.2 Instructions

1. Grab the source from:
   
   https://github.com/Intermesh/groupoffice/releases

   **Note:** For PHP 7.0 use the -php-70.tar.gz file. For all newer PHP version use the php-7.1.tar.gz file

2. Put the unpacked source in apache’s document root.

3. Make sure to make some aliases in the Apache configuration:

   ```
   Alias /public <YOURDOCUMENTROOT>/public.php
   Alias /Microsoft-Server-ActiveSync <YOURDOCUMENTROOT>/modules/z-push/index.php
   #For CalDAV support
   Alias /caldav <YOURDOCUMENTROOT>/modules/caldav/calendar.php
   #For CardDAV support
   Alias /carddav <YOURDOCUMENTROOT>/modules/carddav/addressbook.php
   #For WebDAV support
   Alias /webdav <YOURDOCUMENTROOT>/modules/dav/files.php
   #For WOPI support (Collabora Online and Office Online)
   Alias /wopi <YOURDOCUMENTROOT>/go/modules/business/wopi/wopi.php
   #DAV Service discovery. At least required for iOS7 support
   Redirect 301 /.well-known/caldav /caldav
   Redirect 301 /.well-known/carddav /carddav
   ```

Or if you’re not able to add these aliases you could create a .htaccess file and use mod_rewrite rules. Replace `<YOURDIR>` with the relative URL of where Group-Office is installed:

### 1.3. Tarball
4. If you purchased Group-Office Professional licenses then make sure the IONcube loader is installed and place the license files in the root folder of Group-Office. For example “/usr/share/groupoffice/groupoffice-pro-6.3-license.txt”.

5. Then visit http://yourserver/ and the installer should appear:

![Group-Office installer]

6. Follow the instructions on screen.

7. Finally, create a cron job for the scheduled tasks:

```
* * * * * www-data php <YOURDOCUMENTROOT>/cron.php
```

Optionally you can add the config file location:

```
* * * * * www-data php <YOURDOCUMENTROOT>/cron.php /etc/groupoffice/myoffice/config.php
```
1.3.3 Authentication with CGI or FastCGI

When using PHP through CGI the “Authorization” header might not be passed by default. You can enable this header by adding these “mod_rewrite” rules to your VirtualHost section or .htaccess file:

```
RewriteEngine On
RewriteCond %{HTTP:Authorization} ^(.*
RewriteRule .* - [e=HTTP_AUTHORIZATION:%1]
```

1.4 Extras

1.4.1 LibreOffice Online

With LibreOffice Online you can edit office documents in your browser. You need a working LibreOffice Online server. More info on https://www.libreoffice.org/download/libreoffice-online/

For LibreOffice Online to work you need to setup SSL and allow your Group-Office URL to use it.

Docker

We found the easiest way to set it up is using Docker with Docker compose and Nginx or Apache as reverse proxy. If you run it on the same server as Group-Office you should setup with Apache as the package comes with Apache.

Replace “docs.example.com” everywhere below with your hostname that you’ll use to access LibreOffice Online.
Docker compose

Create a file “docs.example.com/docker-compose.yml”:

```yaml
version: "3.6"
services:
  libreoffice:
    image: libreoffice/online:master
    environment:
      domain: (.+\.example\.com|host\.docker\.internal)
      username: admin
      password: secret
      extra_params: --o:ssl.enable=false --o:ssl.termination=true
      DONT_GEN_SSL_CERT: 1
    volumes:
      - lo_config_volume:/etc/loolwsd
    cap_add:
      - MKNOD
    ports:
      - "127.0.0.1:9980:9980"
    restart:
      unless-stopped
    volumes:
      lo_config_volume:
```

Replace the domain part with a regular expression that allows the Group-Office hosts. For a single domain you can replace this with just “groupoffice.example.com”.

Start docker with the command in the directory “docs.example.com”:

```
docker-compose up -d
```

Nginx

You can use either Nginx or Apache. If you already have Apache installed then skip this section and proceed with Apache. Setup the virtual host in a new text file: /etc/nginx/sites-enabled/docs.example.com:

```nginx
# HTTPS Server
server {
  listen 443 ssl;
  server_name docs.example.com;

  error_log /var/log/nginx/docs_error.log;

  # We use let's encrypt for SSL
  ssl_certificate /etc/letsencrypt/live/docs.example.com/fullchain.pem;
  ssl_trusted_certificate /etc/letsencrypt/live/docs.example.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/docs.example.com/privkey.pem;

  include /etc/letsencrypt/options-ssl-nginx.conf;

  # static files
  location ^~ /loleaflet {
    proxy_pass http://localhost:9980;
    proxy_set_header Host $http_host;
  }
```

(continues on next page)
# WOPI discovery URL
location ^~ /hosting/discovery {
    proxy_pass http://localhost:9980;
    proxy_set_header Host $http_host;
}

# Capabilities
location ^~ /hosting/capabilities {
    proxy_pass http://localhost:9980;
    proxy_set_header Host $http_host;
}

# main websocket
location ~ ^/lool/(.*)/ws$ {
    proxy_pass http://localhost:9980;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header Host $http_host;
    proxy_read_timeout 36000s;
}

# download, presentation and image upload
location ~ ^/lool {
    proxy_pass http://localhost:9980;
    proxy_set_header Host $http_host;
}

# Admin Console websocket
location ^~ /lool/adminws {
    proxy_pass http://localhost:9980;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header Host $http_host;
    proxy_read_timeout 36000s;
}

Check the nginx syntax with:

```
nginx -t
```

It’s OK then reload nginx:

```
systemctl reload nginx
```

Now that Libre Office online is running you can skip to the Group-Office section below to connect it.

**Apache**

Create this virtual host in the text file /etc/apache2/sites-enabled:

```
<VirtualHost *:443>
    ServerName docs.example.com:443
    Options -Indexes
</VirtualHost>
```

(continues on next page)
# SSL configuration, you may want to take the easy route instead and use LetsEncrypt!
SSLEngine on
SSLCertificateFile /path/to/signed_certificate
SSLCertificateChainFile /path/to/intermediate_certificate
SSLCertificateKeyFile /path/to/private/key
SSLProtocol all -SSLv2 -SSLv3
SSLCipherSuite ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-
SHA256:EDH-RSA-DES-CBC3-SHA!:DSS
SSLHonorCipherOrder on
# Encoded slashes need to be allowed
AllowEncodedSlashes NoDecode
# Container uses a unique non-signed certificate
SSLProxyEngine On
SSLProxyVerify None
SSLProxyCheckPeerCN Off
SSLProxyCheckPeerName Off
# keep the host
ProxyPreserveHost On
# static html, js, images, etc. served from loolwsd
# looleaflet is the client part of LibreOffice Online
ProxyPass /looleaflet http://127.0.0.1:9980/looleaflet retry=0
ProxyPassReverse /looleaflet http://127.0.0.1:9980/looleaflet
# WOPI discovery URL
ProxyPass /hosting/discovery http://127.0.0.1:9980/hosting/discovery retry=0
ProxyPassReverse /hosting/discovery http://127.0.0.1:9980/hosting/discovery
# Capabilities
ProxyPass /hosting/capabilities http://127.0.0.1:9980/hosting/
capabilities retry=0
ProxyPassReverse /hosting/capabilities http://127.0.0.1:9980/hosting/capabilities
# Main websocket
ProxyPassMatch "/lool/\.*\//ws" ws://127.0.0.1:9980/lool/$1/ws nocanon
# Admin Console websocket
ProxyPass /lool/adminws ws://127.0.0.1:9980/lool/adminws
# Download as, Fullscreen presentation and Image upload operations
ProxyPass /lool http://127.0.0.1:9980/lool
ProxyPassReverse /lool http://127.0.0.1:9980/lool
</VirtualHost>
Now that Libre Office online is running you can skip to the Group-Office section below to connect it.

**Verify LibreOffice install**

You can verify that the install worked by visiting the URL below in your browser:

https://docs.example.com/hosting/discovery

You should see an XML document. If not then look at the log files:

docker-compose logs

**Debian packages**

**SSL**

We’ve used the Debian packages and setup SSL with Letsencrypt. Then we’ve added this SSL configuration to /etc/loolwsd/loolwsd.xml:

```xml
<ssl desc="SSL settings">
    <enable type="bool" desc="Controls whether SSL encryption is enable (do not disable for production deployment). If default is false, must first be compiled with SSL support to enable." default="true">true</enable>
    <termination desc="Connection via proxy where loolwsd acts as working via https, but actually uses http." type="bool" default="true">false</termination>
    <cert_file_path desc="Path to the cert file" relative="false">/etc/letsencrypt/live/groupoffice.co.cert.pem</cert_file_path>
    <key_file_path desc="Path to the key file" relative="false">/etc/letsencrypt/live/groupoffice.co.privkey.pem</key_file_path>
    <ca_file_path desc="Path to the ca file" relative="false">/etc/letsencrypt/live/groupoffice.co/fullchain.pem</ca_file_path>
    <cipher_list desc="List of OpenSSL ciphers to accept" default="ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH">"</cipher_list>
    <hpkp desc="Enable HTTP Public key pinning" enable="false" report_only="false">
        <max_age desc="HPKP's max-age directive - time in seconds browser should remember the pins" enable="true">1000</max_age>
        <report_uri desc="HPKP's report-uri directive - pin validation failure are reported at this URL" enable="false"></report_uri>
        <pins desc="Base64 encoded SPKI fingerprints of keys to be pinned">
            <pin/></pin>
        </pins>
    </hpkp>
</ssl>
```

**Network**

Change network settings to allow posting from your Group-Office URL. We’ve used a wildcard for all subdomains .example.com* for example:

```xml
<net desc="Network settings">
    <proto type="string" default="all" desc="Protocol to use IPv4, IPv6 or all for both">all</proto>
    <listen type="string" default="any" desc="Listen address that loolwsd binds to. Can be 'any' or 'loopback'.">any</listen>
</net>
```

(continues on next page)
<service_root type="path" default="" desc="Prefix all the pages, websockets, etc. with this path."></service_root>

<post_allow desc="Allow/deny client IP address for POST(REST)." allow="true">

  <host desc="The IPv4 private 192.168 block as plain IPv4 dotted decimal addresses.">192\.168\.[0-9]{1,3}\.[0-9]{1,3}</host>

  <host desc="Ditto, but as IPv4-mapped IPv6 addresses">:fff:192\.[0-9]{1,3}\.0\.[0-9]{1,3}</host>

  <host desc="The IPv4 loopback (localhost) address.">127\.0\.0\.1</host>

  <host desc="Ditto, but as IPv4-mapped IPv6 addresses">:fff:127\.[0-9]{1,3}\.0\.[0-9]{1,3}</host>

  <host desc="The IPv6 loopback (localhost) address.">::1</host>

  <host desc="wildcard" allow="true">.*\.example\.com</host>

</post_allow>

<frame_ancestors desc="Specify who is allowed to embed the LO Online iframe (loolwsd and WOPI host are always allowed). Separate multiple hosts by space."></frame_ancestors>

<net/>

Storage

Change the backend storage to allow your Group-Office URL:

<storage desc="Backend storage">

  <filesystem allow="false" />

  <wopi desc="Allow/deny wopi storage. Mutually exclusive with webdav." allow="true">

    <host desc="wildcard" allow="true">.*\.example\.com</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="true">localhost</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="true">10\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="true">172\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="true">172\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="true">172\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="false">192\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}</host>

    <host desc="Regex pattern of hostname to allow or deny." allow="false">192\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}</host>

  </wopi>

  <webdav desc="Allow/deny webdav storage. Mutually exclusive with wopi." allow="false">

    <host desc="Hostname to allow" allow="false">localhost</host>

  </webdav>

</storage>

After making these changes restart loolwsd:

```
sudo systemctl restart loolwsd
```

Check the status:
sudo systemctl status loolwsd

If anything is wrong view the logs:

sudo journalctl -u loolwsd

**Group-Office connection**

When LibreOffice Online is running you can setup Group-Office to use it. Install the Office Online module from the Business package.

Reload Group-Office and go to System Settings -> Office Online

Then add your LibreOffice Online Service. The default port for LibreOffice online is 9980. But with the reverse proxy setup above we’re using the standard SSL port so it’s not necessary to specify it.

Now every user can go to the files module and use it. Just right click a file and choose “Open with…”.

**Note:** If you get a 404 error when editing because /wopi is not found then you probably are missing the alias in your webserver configuration. The Group-Office Debian package automatically does this but with the tarball package you have to do this manually. *Example configuration can be found here.*
1.4.2 Fail2ban

Fail2ban can be used to block users after a number of failed login attempts. It works by monitoring the apache access log for invalid logins.

1. Make sure fail2ban is installed and enabled on your server.

2. Add the filter definition to /etc/fail2ban/filter.d/groupoffice.conf:

   ```
   [Definition]
   failregex = <HOST> - - .*auth.php.* 401\s
   ignoreregex =
   ```

3. Define the jail in /etc/fail2ban/jail.d/groupoffice.conf:

   ```
   [groupoffice]
   enabled = true
   port = http,https
   filter = groupoffice
   logpath = /var/log/apache2/access.log
   maxretry = 3
   ```

   **Note:** Make sure the “logpath” value is set to the access log of the webserver.

For more information about Fail2ban and the configuration of it visit https://www.fail2ban.org

**Note:** This works from 6.3.62 and up.

**Note:** If you use the Group-Office mailserver then also enable the sasl, dovecot and postfix filters.
1.4.3 IMAP proxy

Webmail systems make lots of imap connections. It’s wise to use imapproxy to cache connections to the IMAP server.

Install imapproxy on the webserver and configure it to proxy the imapservr. Accounts can be configured with “localhost” as the hostname and imapproxy will route this to the real IMAP server.

run:

```
apt-get install imapproxy ca-certificates
```

/etc/imapproxy.conf looks like this (Don’t forget to add “tls_ca_path /etc/ssl/certs/*”):

Note: If you’re running imapproxy on the same host then use “localhost” for “server_hostname” and change “listen_port” to something different than where dovecot is actually listening. 1143 for example. Then use 1143 in Group-Office to use imapproxy

```
## imapproxy.conf
```
```
### This is the global configuration file for imapproxy.
### Lines beginning with a '#' sign are treated as comments and will be
### ignored. Each line to be processed must be a space delimited
### keyword/value pair.

##

#  

## server_hostname

## This setting controls which imap server we proxy our connections to.

server_hostname imap.example.com

##

## connect_retries

## This setting controls how many times we retry connecting to our server.

## The delay between retries is configurable with 'connect_delay'

connect_retries 10
connect_delay 5

##

## cache_size

## This setting determines how many in-core imap connection structures
## will be allocated. As such, it determines not only how many cached
## connections will be allowed, but really the total number of simultaneous
## connections, cached and active.

## cache_size 3072

##

## listen_port

##
```
(continues on next page)
## This setting specifies which port the proxy server will bind to and accept incoming connections from.

```
listen_port 143
```

```
## listen_address

## This setting specifies which address the proxy server will bind to and accept incoming connections to. If undefined, bind to all. Must be a dotted decimal IP address.

##listen_address 127.0.0.1
```

tls_ca_path /etc/ssl/certs/

### 1.4.4 HTTP Proxy

Sometimes it’s handy to use a proxy like nginx to forward requests to an internal server. If you do this then you need to set these headers to make Group-Office operate properly:

- X-Forwarded-Proto: http or https. This is the protocol the proxy server uses.

See for more information: https://linuxize.com/post/nginx-reverse-proxy/

### 1.5 Configuration file

The configuration file can be found in:

- /etc/groupoffice/config.php
- /etc/groupoffice/multi_instance/<DOMAINNAME>/config.php
- The root of the Group-Office installation

Here’s a list of config options:

#### 1.5.1 Recommended PHP settings

For optimal Group-Office performance we recommend these settings. This will allow users to upload files up to 1GB:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory_limit</td>
<td>256M</td>
</tr>
<tr>
<td>post_max_size</td>
<td>1000M</td>
</tr>
<tr>
<td>upload_max_filesize</td>
<td>1000M</td>
</tr>
</tbody>
</table>

---

**Table 2: PHP settings**
### 1.5.2 Locking system settings

Configuration properties configurable in the GUI at System settings can be locked in the config.php or globalconfig.php file:

```php
$config['core'] => [  'title' => 'Pinned title',  'primaryColor' => '2E7D32',  'passwordMinLength' => 6,  'smtpEncryption => 'tls',  'smtpHost => 'localhost',  'smtpUsername => null,  'smtpPassword => null,  'smtpPort' => 587
];
```

### 1.5.3 Branding

If you’d like to brand Group-Office you can easily do this:

Edit /etc/groupoffice/globalconfig.inc.php and add:

```php
$config['product_name'] = 'My Office'; //Will replace the word 'Group-Office' with 'My Office'
$config['custom_css_url'] = '/branding/style.css'; //Loads a custom CSS stylesheet.
$config['support_link'] = 'https://docs.example.com/'; //Changes the URL behind "Help" in the main menu. Can also be an e-mail address
```

In the branding folder create a style.css stylesheet with for example this content:

```css
/**
 * Preferred primary theme colors
 */
:root {
  --c-primary: rgb(27, 100, 139);
  --c-header-bg: rgb(27, 100, 139);
  --c-primary-tp: rgba(27, 100, 139, .16);
  --c-secondary: brown;
  --c-accent: orange;
}

/**
 * Override logo
 */
.go-app-logo, .go-about-logo, .go-settings-logo, #go-logo {
  background-image: url('my-group-office.png');
  width: 240px;
  height: 40px;
}
```

**Note:** It will still leave copyright notices to Intermesh. It’s not allowed to remove those.
You can upgrade any version from Group-Office. For some versions it’s necessary to do it in steps:

- 2.18 can upgrade to 3.7.55
- 3.7.55 can upgrade to 6.2
- 6.2 can upgrade to 6.3
- 6.3 can upgrade to 6.4 and so on. Every major release needs to be upgraded to the next from 6.2 and higher.

Needless to say, but always make a backup before upgrading.

After the upgrade you should consider replacing the “gota” module with the new “assistant” module for better file editing on the desktop.

You can find the change log here:
https://github.com/Intermesh/groupoffice/blob/6.4.x/CHANGELOG.md

You might need to switch to the branch (For example 6.3) to view the latest.

## 2.1 Upgrading on Debian / Ubuntu

### 2.1.1 Core system

**Running 6.3 or higher**

To upgrade minor releases run:

```
apt-get update
apt-get install groupoffice
```
Major release upgrade

When the second digit increases in the version number we call this a major release. For example when upgrading to 6.4 from 6.3. When upgrading to the next major release follow these steps prior to the above:

1. Run the above command first to upgrade to the latest 6.3.
2. Then open your browser to update the database.
3. Then you need to adjust the repository in:

   `/etc/apt/sources.list.d/groupoffice.list`

   For 6.4 change it to:

   `deb http://repo.group-office.com/ 64-php-71 main`

   Or when running PHP 7.0:

   `deb http://repo.group-office.com/ 64-php-70 main`

From older versions than 6.3

- First checkout the blog post about this release
- 6.3 offers quite a lot of changes. It’s strongly recommended to upgrade in a test environment first.
- After the upgrade you should consider replacing the “gota” module with the new “assistant” module for better file editing on the desktop.
- Typically the Custom CSS module was used to replace the logo. To take advantage of the new System Settings -> Appearance features you should remove this CSS code.
- If you customized language then you should convert this to the new language Customize system.
- If your system is older than 6.2 your system probably depends on mcrypt. If you migrate to a new server and you run into email password problems you might need to install https://pecl.php.net/package/mcrypt to decrypt email passwords. If you have plain MD5 hashes in your `pa_mailboxes` database table run this SQL query:

   ```
   update `pa_mailboxes` set `password` = concat("\"PLAIN-MD5\"", `password`) WHERE `password` NOT LIKE '%\%' AND `password` NOT LIKE '$%';
   ```

Steps

1. Make sure you’re on the latest 6.2 version.
2. Make sure you’ve installed the “customfields” and “search” modules as they will become part of the Group-Office core.
3. If you are coming from version 5.0 or lower. Then you must install the projects2 module in 6.2 to migrate your existing data. This can’t be done in a later version!
4. Uninstall the old “groupoffice-com” package but do **NOT** deconfigure the database:

   `apt-get remove groupoffice-com`

5. If you made manual changes inside `/usr/share/groupoffice` (Like installing z-push for example). The the package manager will leave these folders intact. To avoid problems move `/usr/share/groupoffice` away before installing:
mv /usr/share/groupoffice /root/groupofficebak

6. Edit /etc/apt/sources.list and remove:

```plaintext
deb http://repos.groupoffice.eu/ sixtwo main
```

7. Now do a fresh install of the 6.3 Debian package. But note:

- When the installer asks to install a database choose “NO”.
- When the installer asks to replace /etc/groupoffice/config.php, choose “Keep the local version currently installed”.
- Use 63-php-70 or 63-php-71 for the repository instead of the 6.4 one.

Continue at Installation on Debian or Ubuntu.

### 2.1.2 Mailserver

If you’re upgrading from a previous 6.3.x or higher version simply run:

```bash
apt-get install groupoffice-mailserver
```

Or if you also installed the anti spam and virus package:

```bash
apt-get install groupoffice-mailserver groupoffice-mailserver-antispam
```

### Upgrading from 6.2

1. To upgrade from 6.2 you must start with a clean system by removing all previous software and configuration. **Make a backup!**:

```bash
apt-get purge groupoffice-mailserver dovecot* postfix* clamav* spamassassin amavisd-
→new
```

2. Then install the new package:

```bash
apt-get install groupoffice-mailserver
```

3. Move the mail to the new location:

```bash
mv /home/vmail/* /var/mail/vhosts
rmdir /home/vmail
```

4. Remove no longer required packages:

```bash
apt-get autoremove
```

### 2.2 Docker

Please follow the instructions on our docker-groupoffice github repository:

[https://github.com/Intermesh/docker-groupoffice](https://github.com/Intermesh/docker-groupoffice)

Or if you’ve used our developer environment then visit:
2.3 Manual upgrade from the Tarball

We strongly recommend that you use our Debian packages or Docker instead of the tarball. But if you really want use it then follow these steps:

1. Grab the version to install from https://github.com/Intermesh/groupoffice/releases
2. Make sure your system meets the System requirements.
3. Make sure you’re on the latest minor release of your current version (For example 6.2.112 or 6.3.76).
4. When running 6.2 make sure you’ve installed the “customfields” and “search” modules as they will become part of the Group-Office core.
5. If you are coming from version 5.0 or lower. Then you must install the projects2 module in 6.2 to migrate your existing data. This can’t be done in a later version!
6. Move away your old source files to a backup location.

**Warning:** Do not copy the new files over the existing. This will result in a broken system.

7. Put the new files at the right location.
8. If exists copy your old config.php or config.ini and license file to the new files. It is good practice to keep these files one directory higher then the Group-Office source so you have a complete clean code base.
10. Check if you have the right cron job in place:

```
* * * * * www-data php <YOURDOCUMENTROOT>/cron.php
```

**Note:** If you’re upgrading from 6.2 to 6.3 or higher and you are running the CGI version of PHP then you need to add a require rule to add the “Authorization” header. Read more at Authentication with CGI or FastCGI.

2.3.1 Update script

I’ve written a simple bash script that downloads the latest PHP 7.1+ version of Group-Office and replaces all code in the given directory.

You can run it like this:

```
./update-groupoffice.sh <DIR_OF_GROUPOFFICE>
```

**Warning:** Please backup before using!
#!/bin/bash

set -e

TARGET=$1

if [ ! -d "$TARGET" ]; then
    echo "$TARGET doesn't exist!"
    exit 1;
fi

if [ ! -f "$TARGET/version.php" ]; then
    echo "$TARGET is not a Group-Office directory!"
    exit 1
fi

read -r -p "Are you sure you want to update directory '$TARGET'? [y/N]" response;
if [[ "$response" != "y" ]]; then
    exit 0
fi

mkdir -p goupdate

rm -f download

wget https://sourceforge.net/projects/group-office/files/latest/download

tar zxf download

GO=`ls | grep groupoffice`

echo $GO

cd $GO

for f in *; do
    rm -rf ../../$TARGET/$f
    cp -a $f ../../$TARGET
done

cd ../../
You need to backup the following items:

1. The configuration file config.php. Can be found in /etc/groupoffice or in the source directory.
2. The MySQL / MariaDB database. You can find the connection parameters in config.php
3. The data folder. You can find the “file_storage_path” in the config.php file.
4. If you use the Group-Office mailserver, then also backup the e-mail that’s located in /var/mail/vhosts.
Recovery

To recover a backup simply replace the folders you’ve backed up using the information above and restore the database.

4.1 Oops, I accidentally deleted something...

Mistakes happen. Sometimes you’ve accidentally deleted something from Group-Office. You could then restore all missing items from a backup by merging. Please be aware that everything that was deleted will be restored. Also the items that you deleted intentionally.

Steps:

1. Load the backup database in a temporary database called “groupoffice_temp” in this example:

   ```
   mysql -u root -p -e 'create database groupoffice_temp;'
   mysql -u root -p groupoffice_temp < backup.sql
   mysqldump --no-create-info --insert-ignore --complete-insert -u root -p
   <-groupoffice_temp > merge-data.sql
   ```

2. Your file merge-data.sql will contain the command to insert in the database and ignore the insert if the record already exists. Now load this into your live database called “groupoffice_live” in this example:

   ```
   mysql -u root -p groupoffice_live < merge-data.sql
   ```

3. Now all records should be back in the database. To reset the Group-Office cache and sync states run the upgrade procedure in the browser:

   ```
   https://groupoffice.example.com/install/upgrade.php
   ```

4. If you also need to restore missing files the use the command line utility rsync to restore the data.
CHAPTER 5

Migrate

To migrate Group-Office to another server you need to backup all data files and the database.

**Note:** The filesystem paths shown here might be different in your environment.

### 5.1 Backup

Determine where the data files are stored (You can also just open config.php and lookup the value instead of using `cat` and `grep` on linux):

```bash
cat /etc/groupoffice/config.php | grep file_storage_path
```

This outputs:

```plaintext
$config['file_storage_path']='/var/lib/groupoffice/';
```

Now create an archive of this path:

```bash
tar czf groupoffice-files.tar.gz /var/lib/groupoffice
```

This command outputs the database parameters (You can also just open config.php and lookup the values instead of using `cat` and `grep` on linux):

```bash
cat config.php | grep db
```

This outputs:

```plaintext
$config['db_type']='mysql';
$config['db_host']='localhost';
$config['db_name']='groupoffice';
```

(continues on next page)
Now create a dump of the database:

```
mysqldump groupoffice -u groupoffice -p > groupoffice-20190101.sql
```

You might want to compress this file to save bandwidth:

```
tar czf groupoffice-database.tar.gz groupoffice-20190101.sql
```

Now we've packed up all necessary files in archives:

- groupoffice-files.tar.gz
- groupoffice-database.tar.gz

### 5.2 Installing the backup on the new server

1. Install a clean *Installation* following this manual
2. Replace the file folder with the groupoffice-file.tar.gz contents.
3. Replace the database with the mysqldump created in the backup.
4. Run `/install/` to finish.
CHAPTER 6

Logging

Errors on the Group-Office server will be logged through PHP. Normally you can find errors in the webserver error log. When using apache this is:

```
/var/log/apache2/error.log
```

ActiveSync is implemented with the Z-push library. This will log errors to:

```
$file_storage_path/log/z-push-error.log
```

### 6.1 Debugging

When `$config['debug']` = true; is set in config.php. Then debug info will be logged into:

```
$file_storage_path/log/debug.log
```

### 6.1.1 Z-Push

When `$config['zpush2_loglevel']` = 32; is set in config.php. Then additional ActiveSync information will be logged into:

```
$file_storage_path/log/z-push.log
```

If you want to debug specific users you can set:

```
$config['zpush2_special_log_users'] = ['username1', 'username2'];
```
CHAPTER 7

System settings

System settings are only available to administrators. When you click on the avatar icon on the top right you can open the “System settings” page. This is the place to configure everything for Group-Office.

**Note:** When setting up Group-Office we recommend to review each tab and make adjustments where necessary before you start creating users and start using the system.

### 7.1 General

#### 7.1.1 Title

The title is used in the browser tab and as sender name for notifications.

#### 7.1.2 Language

The default system language. The user’s language is detected from the browser first. If that language isn’t available this language will be used.

#### 7.1.3 URL

The full URL to Group-Office. Group-Office must only be reachable via this URL. Try to avoid to use multiple URL’s for Group-Office.

### 7.2 Appearance

On the appearance page you can upload your logo and set a color for the interface. We recommend to use a dark color with a light logo.
Fig. 1: System settings
Fig. 2: System settings - General

7.2. Appearance
If you need further customizations then a theme can be created.

Fig. 3: System settings - Appearance

### 7.3 Notifications

On the notifications page you must configure an SMTP server for outgoing e-mail messages. Use the “Send test message” button to test your settings.

#### 7.3.1 Debug

For debugging purposes you can enable that all e-mail from Group-Office will be sent to the configured e-mail address to avoid spamming of users when testing scripts :)

### 7.4 User groups

The first thing you need to do is create user groups for your organization. Permissions are much easier to handle with groups then with individual users. When you grant the secretary access to all calendars as a user it will be a painful job to reset all permissions to another user when a new secretary joins the company. If you had done it with a group you could simply add the new employee to the secretary user group. So choose your groups wise with user permissions in mind. For example create:

1. A secretary group called “Secretary”
7.4. User groups

Fig. 4: System settings - Notifications
2. A group for the engineers called “Engineering”

### 7.4.1 Special user groups

A few special groups are created by default:

1. **Admins**: Users of this group will have permission to everything. Usually you don’t want to add any user to this group. See above.

2. **Everyone**: All users are in this group. Use carefully when granting permissions to this group.

3. **Internal**: All calendars, address books and task lists are shared to this group with write access by default. New employees are commonly added to this group. It should contain all company users and not your customers. If you don’t want to use this feature you may remove this group.

![Fig. 5: System settings - Groups](image)

### 7.4.2 Edit group

The group edit dialog has 3 tabs:

**Members**

Here you can add group members. The selected members are on top. You can search for new members to add. Add them by ticking the checkbox.
### Group: Internal

<table>
<thead>
<tr>
<th>Name</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merlijn Admin</td>
<td>✅</td>
</tr>
<tr>
<td>Amy Wong</td>
<td>✅</td>
</tr>
<tr>
<td>Bender Bending Rodríguez</td>
<td>✅</td>
</tr>
<tr>
<td>Philip J. Fry</td>
<td>✅</td>
</tr>
<tr>
<td>Hermes Conrad</td>
<td>✅</td>
</tr>
<tr>
<td>Hubert J. Farnsworth</td>
<td></td>
</tr>
<tr>
<td>John A. Zoidberg</td>
<td></td>
</tr>
<tr>
<td>Turanga Leela</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6: System settings - Group members

7.4. User groups
Permissions

Here you can manage who can use (for sharing) / edit this group. By default only the group members can use the group for sharing.

![Fig. 7: System settings - Group permissions](image)

Modules

Here you can manage which modules are available to group members. Alternatively you can also manage this at the modules page.

7.4.3 Group defaults

Here you can configure which groups can use new groups by default. Only members of these groups will be able to share items with users and user groups they are not member of. This default to “Everyone” so everyone can see all users and all groups. This might be a problem for larger organizations.

There’s also an “Apply to all” button to apply the changes to all existing groups.
7.4. User groups

Fig. 8: System settings - Group modules
Fig. 9: System settings - Group defaults
7.5 Modules

In this screen you can install modules and configure module access. To install a module simply check the box in the row.

![System settings screenshot](image)

Fig. 10: Modules

7.5.1 Install

After install a permissions window will pop up where you should select the user groups you want to grant access. To change permissions later double click the row.

7.5.2 Permissions

In the permissions window you can double click the level. You can choose “Use” which grants basic access or “Manage” for users that need administrative functions in the module.

You can also manage this at group level.
Fig. 11: Module permissions
7.6 Authentication

On the authentication page you can manage:

1. The minimum password length of user passwords
2. Logout users automatically if they are inactive and disable the “Remember my login until I press logout on this computer”.
3. Two factor authentication
4. Restrict access based on IP addresses
5. LDAP Authentication
6. IMAP Authentication

Note: The features vary depending on which modules are installed on your system.

7.6.1 Restrict access

To enhance security you can limit the locations (IP Addresses) where users can login to the system. You can set this per group:

Allowed groups

Define which groups are allowed to login from which IP addresses. You can use "*" to match any characters and "?" to match any single character. eg. '192.168.12.*'. Be careful, You can lock yourself out!

<table>
<thead>
<tr>
<th>Group</th>
<th>IP pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admins</td>
<td>*</td>
</tr>
<tr>
<td>Internal</td>
<td>192.168.1.*</td>
</tr>
<tr>
<td>elmer</td>
<td>84.168.1.23</td>
</tr>
</tbody>
</table>

Fig. 12: Allowed groups

Warning: You can easily lock everybody out from the system with this feature. I recommend that you add group “Admins” with pattern "*" first so that Admins can always login. When you’re sure everything is setup correctly you can remove that rule.

Rules

Login is allowed when:

1. There are no rules defined for any group you are a member of.
2. One of the rules of groups you are a member of matches your IP address
Each rule has a group and a pattern for IP addresses. You can use “*” to match any part of an IP address and “?” to match a single character.

Some examples:
- “*” will match any IP address
- “192.168.1.?“ will match 192.168.1.1 till 192.168.1.9.
- “192.168.1.*“ will match 192.168.1 till 192.168.1.255

7.6.2 LDAP

You can use an LDAP server to authenticate and autocreate Group-Office users and e-mail accounts.

Install

1. Install the ldapauthenticator module at Start menu -> Modules
2. Reload Group-Office

Configure

Go to System Settings -> Authentication

There’s a new section for LDAP server profiles. Click on the add button to add a profile. Enter the LDAP server details and domains. The domains are important because users need to login with <LDAPUSER>@<DOMAIN> to trigger the LDAP authenticator.
Note: If you’re using Microsoft ActiveDirectory then you should use “samaccountname” as the username attribute. The users DN is typically CN=Users,DC=example,DC=com
Now check if you can login with the LDAP domain:
It should autocreate the LDAP user. In the user settings it won’t be possible to change the password as that’s not supported yet:

**Synchronization**

It’s also possible to synchronize users and groups from the LDAP server. Simply enable the checkboxes and optionally adjust the queries to fetch the correct users and groups.

When you enable the sync, it will schedule the sync to run the next minute. After that it will sync daily at midnight.
Advanced

When a user or group is synchronized, it fires an event. So it’s possible to build a module that can customize the LDAP synchronization.

If you want to test the sync on the terminal lookup the ID from your configuration in table ‘ldapauth_server’ and run:

```
php cli.php community/ldapauthenticator/Sync/users --id=<ID FROM DATABASE> --dryRun=1 --delete=1 --maxDeletePercentage=50
```

Test environment

For testing I’ve used this docker image: https://hub.docker.com/r/rroemhild/test-openldap/

```
docker pull rroemhild/test-openldap
docker run --privileged -d -p 389:389 rroemhild/test-openldap
```

7.6.3 IMAP

You can use an IMAP server to auto create users and e-mail accounts.

Install

1. Install the imapauthenticator module at Start menu -> Modules
2. Reload Group-Office

Configure

Go to System Settings -> Authentication. There’s a new IMAP Authentication section there.
Click on the add button to add a new IMAP server profile:
Server profile

IMAP Server

Domains: intermesh.localhost

Enter the domains this imap server should be used to authenticate. Users must login with their e-mail address and if the domain matches this profile it will be used.

Hostname: mailserver
Port: 143
Encryption: TLS

- Validate certificate
- Remove domain from username

Users must login with their full e-mail address. Enable this option if the IMAP expects the username without domain.

SMTP Server

Hostname: mailserver
Port: 587

- Use user credentials

Enable this if the SMTP server credentials are identical to the IMAP server.

Username:
Password:
Encryption: TLS

- Validate certificate
Now you should be able to login with the full e-mail address. You can’t change passwords with Group-Office using this authentication method.

### 7.6.4 Google Authenticator

You can use two factor authentication when the Google Authenticator module has been installed.

First install any OTP app on your mobile phone. You can use for example:

1. **Google Authenticator**
2. **OTP Auth** (iOS only)

After installing an OTP app on your phone go to “My Account” -> “Account” and click on the “Enable Google Authenticator” button.

![Google Authenticator](image)

Fig. 13: Google Authenticator

You will be prompted for your Group-Office password. Next you will be asked to scan the QR code with your app to register Group-Office. Enter the code from the phone to confirm you’ve set it up and you’re done.

**Note:** While this module is called Google authenticator it doesn’t need a Google account. It does not interact with Google when logging in in any way.

### 7.7 Users

When clicking users you will see a list of all Group-Office users. Via the “more” menu in the record you can:
Fig. 14: System settings - Authentication
1. Edit the user
2. Login as this user
3. Archive a user
4. Delete the user

**Note:** Instead of deleting or archiving users you can simply disable them. Disabling a user will keep all data and references to the user but the user can’t login or used for new items. To deactivate a user you must “Edit” it and toggle “Login enabled” on the account page.

**Warning:** Archiving a user will not only disable the user. It will also hide any calendars, note books task lists and address books from other users. If for a user is re-enabled, these shares have to be restored manually by either the user or the administrator.

![User list](image)

Fig. 15: User list

### 7.7.1 User defaults

Before adding any user. Check the ‘User settings’ to avoid unnecessary changes to user settings after creating them. Click the settings icon to change default values and manage custom fields.
Fig. 16: User settings
7.7.2 Adding a user

**Note:** Before adding users make sure you’ve setup *User groups* with the right module access. So setting up the user permissions is a simple matter of adding it to the right user group.

To add a user click the plus icon. A short wizard opens with three steps.

Supply the username, display name and e-mail address.

- Username is case insensitive.
- Display name is used in Group-Office
- Provide an account e-mail address.
- Because often Group-Office is used as primary e-mail service you must provide a secondary e-mail address for e-mail recovery. If not available just use your primary e-mail.
Provide a password. You can also use the button in the first field top generate a strong password.

Finally, add the user to the right User groups and click ‘Finish’.

### 7.7.3 Edit user

To edit a user double click or use the more menu. The edit dialog is identical to the ‘Account settings’ page but adds some administrative features:

- Group management
- Disable / enable login
- Set disk quota

#### Disk quota

If you leave this blank then users can use an unlimited amount of storage. If set then the user will be limited to this amount of disk space.

Disk quota applies to all files in the user’s home folder of the files module. Other locations such as projects and address book folders are owned by the “admin” user.

### 7.7.4 Visibility of users

By default all users are visible to eachother. You can see users when you share something with another user for example. If you’d like to change this you need to change the default permissions of a new user group. Because every user get’s it’s own personal group used for permissions. You can change this at Group defaults.

You can change visibility settings per user in the user account page at the “Visible to” tab.

---

**Note:** After an upgrade from 6.2 none of the users are visible. This is a known issue. If you’d like to make all users visible then edit the Default permissions of “Group” and add for example group “Everyone” and click “Add to all”. Now all users can see all groups and users.
Fig. 20: Edit user
7.8 Custom fields

Every organization is different and has different data to store. That’s why most entities in Group-Office can be extended with custom fields.

Custom fields show in the:
- Edit dialogs
- Detail views
- Grid columns
- Advanced search and filters

You can create custom fields via System settings at:
- Notes
- Address book
- Billing
- Projects
- Tickets
- Calendar
- Tasks
- Files
- … and many more!

Note: The above list depends on the modules installed. You may have less or more options.

7.8.1 Creating and editing fields

Navigate to System settings -> Custom fields. You’ll see a list of customizable entities and their corresponding modules.

Click on the pencil icon in the right column to manage custom fields for a certain object. For a non-customized object, the resulting window will look like below:

Click the + icon to add a field set first. A field set is a group of fields that has a title and an optional description that will be shown on the form. You can also hide field set in some cases. This depends on the module. In the address book you can only show it on certain address books for example.

When you’ve created a field set you can add a field via the ‘+’ button on the field set row:

The image below shows the custom field types you can add.

Note: The types of custom fields that you can add depend on the modules you have installed.

Depending on the type of custom field, you have a plethora of options for your customization. For instance, you can conditionally require and/or hide a field:

The custom field will show in the:
Fig. 21: Custom fields overview window

Fig. 22: Custom fields for a non-customized object
7.8.2 Filters

Aside from displaying and storing data, it is a breeze to filter items by custom field values. In the left column, near the Filters header, click on the + and select Filter:

7.8.3 Conditionally required fields

It’s possible to make fields required based on a simple condition. You can also hide the field if the condition does not match.

For example you could create a checkbox called “provideDetails” and create a text field “details” with a condition ‘provideDetails = true’. The result will be that when you check the box the provideDetails field will be shown and made required.

The condition syntax is very simple:

```
<fieldName> <operator> <value>
```

Where:

- fieldName can be any property or custom field
- operator can be: =, !, >, >=, <, <=
- value is a string without quotes. For checkboxes you can use 0,1 or true or false.

**Warning:** The fieldName, operator and value must be separated by spaces.
7.8. Custom fields

Fig. 24: Custom field types
Fig. 25: Example text fields with extra conditions
Fig. 26: Example edit dialog. Please note the separate tab

7.8. Custom fields
Fig. 27: Detail view
7.8. Custom fields

![Diagram showing custom fields in grid columns.](image)

- ID
- Name
- Organizations
- Address Book
- Created at
- Modified at
- Created by
- Modified by
- Job title
- Department
- Registration number
- Debtor number
- IBAN
- VAT number
- Phone numbers
- E-mail addresses
- First name
- Last name
- Birthday
7.8.4 Template field

The template field can render data using *Templates*.

The template holds the “entity” model as variable. Which is the entity the custom field belongs too.

You can for example take the first linked contact and store it’s name:

```plaintext
[assign firstContactLink = entity | links:Contact | first]
{{firstContactLink.name}}
```

Or store the postal code of the first linked contact:

```plaintext
[assign firstContactLink = entity | links:Contact | first]
[assign address = firstContactLink.addresses | first]{{address.zipCode}}
```

Or if you want addresses of the contact in the grid you could create a field with this template:

```plaintext
{{entity.addresses | column:formatted| implode}}
```

7.9 Document templates

Document templates are text or spreadsheet documents with auto data tags. You can fetch data from all the linked items. For example a contact in the address book has links to a project and a companies. When you create a document you can use the contact, project and company data.

You can create document templates for OpenOffice.Org and Microsoft Office 2007 (You can use older versions but then you must install the compatibility pack) and up in Group-Office.

To create a template use OpenOffice.Org or Microsoft office to create a new document (in .docx format) and upload it at:
Address book -> Administration -> Templates -> Add document template

The template variables you can use are the same as for e-mail templates.

### 7.10 Default permissions

At this page you can manage the permissions new items get when they are created. By default the creator and administrator get access. You can also use this to add a group to all existing items or reset all permissions per type.

#### 7.10.1 Update existing items

When you’ve changed the defaults and you’ve already created a lot of calendars for example. You might want to add a group to all calendars. You can do this with the “Add to all” or “Reset all” buttons.

Double click “Calendar” to open the edit default permission screen:

Then click **Add to all** to keep existing permissions but add the missing groups from the defaults. Or click **Reset all** to remove existing permissions and reset it to the defaults.

### 7.11 Rocket Chat

Since version 6.4.145 Group-Office can integrate with Rocket.Chat. With rocket chat you can chat with your team in different channels, share files and even start a **video conference** from your desktop, phone or tablet!

#### 7.11.1 Installation

To install you need a working Rocket.Chat installation. See the Rocket.Chat documentation for that.

Login to Rocket.Chat as administrator and:

1. go to Administration -> Settings -> OAuth.
Fig. 31: Default permissions
7.11. Rocket Chat
Fig. 32: Rocket Chat
2. On the top right choose “Add custom oauth”.

3. Note the callback URL / or redirect URI on top. For some reason they made it grey so it doesn’t stand out. You need that later when adding the client in Group-Office.

4. Fill in your URL to Group-Office plus /api/oauth.php. For example: https://groupoffice.example.com/api/oauth.php

5. Copy the settings from the image below. Instead of the example password choose a strong password:

6. Login to Group-Office and go to System Settings -> Oauth 2.0

7. Click “Add client” and fill in the fields:
   • Match the “Identifier” and “Secret” from step 5.
   • Enable “Is confidential”
   • The redirect URI is the callback URL on top of the Rocket.Chat custom oauth page. It’s marked in the red square in the screenshot below:
     The input should look like this:

8. Click save and test if you can login with Group-Office. You should see a “Login with Group-Office” button.

Note: If you get an error double check these settings:
   • Administration -> General -> Site URL. It has to match for the openid handshake to work.
   • Administration -> Oauth -> Groupoffice: check if id and secret match the parameters in the Group-Office client. Also verify the other fields.
When setting up your OAuth Provider, you'll have to inform a Callback URL. Use http://localhost:3000/_oauth/groupoffice

Enable

URL

https://groupoffice.example.com/api/oauth.php

Token Path

/token

Token Sent Via

Payload

Identity Token Sent Via

Same as "Token Sent Via"

Identity Path

/userinfo

Authorize Path

/authorize

Scope

openid

Param Name for access token

access_token
When setting up your OAuth Provider, you'll have to inform a Callback URL. Use http://localhost:3000/_oauth/groupoffice
CHAPTER 8

Account settings

Click on your avatar in the top right corner and select “My account” to change your account settings.

Fig. 1: My Account
8.1 Account

On this tab you can change your:

- Photo / Avatar
- Username
- Display name
- Password

You can also setup two factor authentication with Google Authenticator.

8.2 Profile

When the address book module is installed. You can set your profile here. A contact is created in the users address book when you enter your profile. The users address book is configurable at System settings -> Users -> Settings.

![Fig. 2: Profile](image)

8.3 Look and Feel

Setup:

- Regional settings
- Notification and sound settings
- Theme and user interface options
8.4 Working week

Setup the number of hours your work each day of the week. This information is used in the holidays and absence module to calculate the number of hours for periods. It’s also used in the time tracking module.

8.5 Synchronization

Configure what will be synchronized to other devices with CalDAV, CardDAV and ActiveSync.

![Synchronization Figure](image)

Fig. 3: Synchronization

8.6 Modules

Depending on the modules that are installed other tabs appear here too.
It’s possible to synchronize your phone, tablet or computer with Group-Office. The following items can be synchronized:

- Contacts
- Calendar
- Notes
- Tasks
- E-mail

On desktop computers you can edit files directly with your desktop applications.

### 9.1 Server hostname

You can connect to Group-Office with various software and devices. For all systems it’s important to know the hostname of your Group-Office. You can find that in the browser as shown in the highlighted part in this figure:

In this example web URL is [https://intermesh.group-office.com/groupoffice](https://intermesh.group-office.com/groupoffice) and the server hostname is “intermesh.group-office.com”.

When you’ve noted your server hostname please proceed to the section of your device.
9.2 Supported Devices

9.2.1 Android

On Android you can connect using ActiveSync. You can synchronize:

- Calendar
- Contacts
- E-mail

To setup an ActiveSync account take the following steps:

1. Navigate to Settings and tap “Users and Accounts”.
2. Scroll down and choose “Add account”.
3. In the “Add an account” screen tap “Exchange”.
4. In the next screen enter your e-mail address and tap the “Manual setup” button in the bottom left corner.
5. Then choose for the “Exchange”.
6. Enter your password.
8. In the next screen you must adjust your server settings. It will prefill the username with the e-mail address and the server name with the domain from your e-mail address. This is most likely incorrect. Please adjust to your Group-Office username and enter the Server hostname.
9. In the next screen you must confirm the remote security administration privileges.
10. For some reason you must confirm this twice but now with more info.
11. Optionally you can give it an account name in the final step.
12. Now you must give it some time to sync everything. Then check your contacts, e-mail and calendar for your Group-Office data!

9.2.2 iOS (iPhone or iPad)

On iOS you can connect using ActiveSync. You can synchronize:

- Notes
- Calendar
- Contacts
- E-mail
- Tasks (Called “Reminders” on iOS)

To setup an ActiveSync account take the following steps:

1. Navigate to Settings and lookup the “Accounts & Passwords” page.
2. Click on “Add account”.
3. Now tap on “Exchange”.
4. Fill in your e-mail address and an account description
9.2. Supported Devices

- **Sound**
  - Volume, vibration, Do not disturb

- **Storage**
  - 59% used - 13.23 GB free

- **Security & location**
  - Play Protect, screen lock, fingerprint

- **Users & accounts**
  - Current user: Merijn

- **Accessibility**
  - Screen readers, display, interaction controls

- **Google**
  - Services & preferences

- **System**
  - Languages, time, backup, updates

- **Support & tips**
  - Help articles, phone & chat, getting started
Chapter 9. Connect a device
9.2. Supported Devices
Add your email address

Enter your email address
mschering@intermesh.nl
mschering@intermesh.nl

What type of account is this?

- Personal (POP3)
- Personal (IMAP)
- Exchange
Incoming server settings

Domain\Username
mschering

Password
******

Client certificate
None

Mobile device ID
androidc1787503597

Server
intermesh.group-office.com

Port
443

Security type
SSL/TLS

NEXT
Incoming server settings

Validating server settings...

⚠️ Remote security administration

The server intermesh.group-office.com requires that you allow it to remotely control some security features of your Android device. Do you want to finish setting up this account?

CANCEL   OK
Activate device admin app?

Email

The server intermesh.group-office.com requires that you allow it to remotely control some security features of your Android device.

Activating this admin app will allow the app Gmail to perform the following operations:

- **Erase all data**
  Erase the phone's data without warning by performing a factory data reset.

- **Set password rules**
  Control the length and the characters allowed in screen lock passwords and PINs.

- **Monitor screen unlock attempts**
  Monitor the number of incorrect passwords typed when unlocking the screen and lock the phone or erase all the phone's data if too many incorrect passwords are typed.

- **Lock the screen**
  Control how and when the screen locks.

- **Set screen lock password expiry**
  Change how frequently the screen lock password, PIN or pattern must be changed.

- **Set storage encryption**
  Require that stored app data be encrypted.

- **Disable cameras**
Your account is set up and emails are on their way!

Account name (optional)
mschering@intermesh.nl

Displayed on sent messages
9.2. Supported Devices
5. In the popup dialog choose “Configure manually”.

6. Now enter your password.

7. iOS now attempts to discover settings on the domain of your e-mail address. In this example “intermesh.nl”. It is likely to fail because this is not the correct ActiveSync server address. It give a certificate error. Just ignore it by clicking “Continue”.

8. Enter your Server hostname and username.

9. In the final screen you can optionally disable some apps.

10. Now you must give it some time to sync everything and check your contacts, e-mail and calendar for your Group-Office data!
9.2. Supported Devices
Chapter 9. Connect a device
9.2.3 Linux

- Files
- Calendar
- Contacts
- E-mail

Thunderbird

You can use Thunderbird with the lightning extension to synchronize e-mail, contacts and calendars with IMAP, CardDAV and CalDAV.

Walkthrough guide coming soon…

Files

The Group-Office Assistant is a small program that you can install on your Windows, MacOS or Linux computer. It will automatically download files opened from Group-Office and monitor it for changes. When the file is saved it automatically uploads it back to Group-Office.

On Debian based distributions you can install the Group Office Assistant.

1. First add our APT repository:

9.2. Supported Devices
Group-Office manual

```bash
echo "deb http://repo.group-office.com/ one main" | sudo tee /etc/apt/sources.list.d/groupoffice-assistant.list
```

2. Add our public key:

```bash
gpg --keyserver pool.sks-keyservers.net --recv-keys 0758838B

gpg --export --armor 0758838B | sudo apt-key add -
```

3. Update APT:

```bash
sudo apt-get update
```

4. Then install Group Office Assistant by running:

```bash
sudo apt-get install groupoffice-assistant
```

5. Now double click a file in Group Office and it can be edited on your desktop instantly.

---

**Note:** For Chrome users. You might be annoyed by the popup dialog every-time you open a file. Here’s a solution for that: https://superuser.com/questions/1481851/disable-chrome-to-ask-for-confirmation-to-open-external-application-everytime

---

### 9.2.4 MacOS

On Mac OS you can synchronize:

- Files
- Calendar
- Contacts
- E-mail

**Files**

The Group-Office Assistant is a small program that you can install on your Windows, MacOS or Linux computer. It will automatically download files opened from Group-Office and monitor it for changes. When the file is saved it automatically uploads it back to Group-Office.

**Click here to download Group-Office Assistant for MacOS.**

After downloading take the following steps to install:

1. Open the DMG file and drag “Group-Office Assistant” in the “Applications” folder.
2. Launch Group-Office Assistant and MacOS will prompt for a security warning.
3. Navigate to System Preferences -> Security and Privacy and click the “Open anyway” button.
4. Confirm.
5. In Group-Office right click on a file and choose “Open with”.
6. Select the “Your desktop application (WebDAV) option to use the assistant.”
9.2. Supported Devices
Note: For Chrome users. You might be annoyed by the popup dialog every-time you open a file. Here's a solution for that: https://superuser.com/questions/1481851/disable-chrome-to-ask-for-confirmation-to-open-external-application-everytime

WebDAV client issues

Unfortunately the MacOS webdav implementation can be rather slow due to a lot of redundant requests. We found that the Mountain Duck WebDAV client is much faster.

Calendar & Contacts

Adding contacts and calendar accounts work identically in MacOS. Just choose “CalDAV” for calendars and “CardDAV” for contacts.

1. Open System Preferences and click “Internet Accounts”.

2. Click the “+” button in the bottom left to add an account. Scroll down and choose “Other account”.

3. Now choose “CalDAV” for calendars or “CardDAV” for contacts.

4. Select “Manual” in the “Account Type” dropdown and enter the username, password and Server hostname.

5. Click “Sign in” to finish and you might be able to select additional data sources in the account settings screen.

6. Now check your Calendar or Contact app for your Group-Office data!

Note: Unfortunately the Contacts app on MacOS will only sync the first address book in CardDAV.
Chapter 9. Connect a device
9.2. Supported Devices
E-mail

E-mail on MacOS works with IMAP and SMTP. These settings vary between providers so please ask your system administrator for the right IMAP and SMTP settings.

Intermesh uses:

<table>
<thead>
<tr>
<th>Username</th>
<th>E-mail address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Group-Office password</td>
</tr>
<tr>
<td>Incoming mail server (IMAP)</td>
<td>imap.group-office.com on port 143</td>
</tr>
<tr>
<td>Outgoing mail server (SMTP)</td>
<td>smtp.group-office.com on port 587</td>
</tr>
<tr>
<td>Encryption</td>
<td>TLS Encryption for both servers</td>
</tr>
</tbody>
</table>

To add a mail account take the following steps:

1. Open System Preferences and click “Internet Accounts”.
2. Click the “+” button in the bottom left to add an account. Scroll down and choose “Other account”.
3. Click on “Mail account”.
4. Enter your e-mail address and password and click “Sign in”.
5. If auto discovery fails enter the server addresses and click “Sign in”.
6. Select the apps you’d like to synchronize and click “Done”.
7. Check your mail!

9.2.5 Windows

On Windows you can connect Outlook using ActiveSync. You can synchronize:
9.2. Supported Devices
Chapter 9. Connect a device
Outlook / Windows 10

Note: This guide describes Outlook but this kind of ActiveSync account also works for the Windows 10 Mail, People and Calendar app.

To setup an ActiveSync account take the following steps:

1. Open Outlook and click “File”.
2. Then click “Add account”.
3. Select “Manual setup” and click “Next”.
4. Choose “Outlook.com or Exchange ActiveSync compatible service” and click “Next”.
5. In the next screen you must adjust your server settings. It will prefill the username with the e-mail address and the server name with the domain from your e-mail address. This is most likely incorrect. Please adjust to your Group-Office username and enter the Server hostname. When done click “Next”.
6. Outlook will test your settings. If all is well you should see the following screen.
7. Now you must give it some time to sync everything. Then check your contacts, e-mail and calendar for your Group-Office data!

Files

The Group-Office Assistant is a small program that you can install on your Windows, MacOS or Linux computer. It will automatically download files opened from Group-Office and monitor it for changes. When the file is saved it automatically uploads it back to Group-Office.
Chapter 9. Connect a device
9.2. Supported Devices

Congratulations! All tests completed successfully. Click Close to continue.
Click here to download Group-Office Assistant for Windows.

After installing it you can use it as follows:

1. In Group-Office right click on a file and choose “Open with”.
2. Select the “Your desktop application” option to use the assistant.

Note: If the Assistant doesn’t start you might need to install the Microsoft Visual Studio Runtime library: https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

Note: For Chrome users. You might be annoyed by the popup dialog everytime you open a file. Here’s a solution for that: https://superuser.com/questions/1481851/disable-chrome-to-ask-for-confirmation-to-open-external-application-everytime

WebDAV

Additionally you can map Group-Office as network drive using WebDAV. The assistant makes this easy for you after editing a file. Then you can right click the icon and choose “Connect network drive”.

WebDAV client issues

Unfortunately there are some known issues with the native Windows WebDAV implementation:

1. When opening office files you have to re-authenticate: https://support.microsoft.com/en-us/help/2019105/authentication-requests-when-you-open-office-documents
2. There’s a path length limit in both windows and office. So long paths will fail. The URL of your Group-Office counts as path too.

We recommend to use the assistant as it doesn’t suffer from these issues and generally works faster.
Group-Office has a very powerful link function. Each item can be linked to another. The link function gives you the ability to organize your data easily. You can for example link:

- E-mail messages
- Tasks
- Projects
- Contacts
- Organizations
- Calendar events
- Notes
- Files
- Invoices
- Quotes

### 10.1 Create a link

Every item that supports links has an “Add” menu button with a + icon.

If you want to search for an existing item in the system choose “Existing item”. All other items will open the create dialog to link a new item.

### 10.2 Viewing links

Links are grouped by type in detail views. Here’s an example of how links are shown in the detail view of a contact: If you’d like to browse, search and filter links without jumping in the system you can use the link browser.
Fig. 1: Add menu button
Fig. 2: Contact with links
Fig. 3: Link browser
Group-Office has a powerful search function that will search through all items. You can find it at the top bar:

### 11.1 Query syntax

When you enter words it will find results that match all of those words. For example when searching for:

```
John Doe
```

It will find anything with John* AND Doe*. Because of the automatic wildcards (*) also “Jonathan” will match for example.

If you want to search for a phrase or without wildcards then you can use quotes:

```
"John Doe"
```
Fig. 1: Search via top bar
Any user can write comments on items that support it.

Items that support it are among other things:

- Contacts
- Events
- Tasks
- Projects
- Invoices
- Files
- Folders
- Notes
All items in Group-Office are protected with permissions. For example address books, task lists, calendars, Note categories, Billing books and project types have access control. When you setup permissions you can add user groups and users to the access control list. There are different levels of permission:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read only</td>
<td>All items are read only</td>
</tr>
<tr>
<td>Create</td>
<td>Users may create new items but can’t change them afterwards.</td>
</tr>
<tr>
<td>Write</td>
<td>Full write access</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete is allowed</td>
</tr>
<tr>
<td>Manage</td>
<td>The user can do everything and can also change the permissions.</td>
</tr>
</tbody>
</table>

Sharing works similar for most items. You can access the share dialog via the menu in the address book for example:

![Share via menu](image)

**Fig. 1: Share via menu**

**Note:** If you can’t find users to share with it’s probably due to them being invisible. Read more about that in the
Fig. 2: Share dialog
13.1 Global address book

If you want to create a global shared address book it is recommended to add this address book as the “admin” user so it won’t be deleted along with a regular user when this employee leaves the company for example.

13.2 Default permissions

By default items are only shared with the administrator and owner. You can change that at the Default permissions. You can also add a group or user to all existing items.
Some modules support custom filters. You can create a reusable filter in which you create your own conditions. They also work great together with Custom fields.

For example you’ve defined a custom field “Education” and you want to show all “Software engineers” you can easily do this with a custom filter.

Other useful examples:
- Find all upcoming birthdays
- Find all contacts with invoices with the “Has links to..” and then select invoice.
- Find contacts with recent comments

Here’s a screenshot of the address book:

In the picture above you can see two custom filters called “Software engineers” and “Users”.

### 14.1 Adding a filter

Click the add button in the filters toolbar and choose “Filter” to open the filter dialog. Add your conditions and click “Save”. If you want to share this filter with other users you can add groups in the permissions tab.

### 14.1.1 Wildcards

You can use % to match 0 or more characters. or _ to match a single character.

There are already some wildcards applied automatically. When filtering on strings you can choose:
- Contains, will put a %...% before and after your phrase.
- Equals, will put use no wildcards
- Starts with, will put a .% after your phrase.
- Ends with, will put a %.. before your phrase.
Fig. 1: Filters
14.1.2 Sub groups

To create complex queries, Group-Office supports sub groups. You can use that to create a query for example:

Select all contacts that have job title “CEO” and work in Germany or the Netherlands. You would have to create a sub group for the countries as they are using an “OR” operator.

It can also be used for inverting one of the queries. For example select all contacts that have job title “CEO” but do NOT work in the Netherlands.

Click the “Add sub group” button to add one.

14.2 Adding an input field

You can also create variable input field filters. You can add an a text filter field for a custom field you have or drop down filter that. You can see some examples in the screenshot on top of this page (Address list, Country and Custom select).

To add one click the add button and select “Input field”. Then choose the filter you’d like to add and click ‘Save’.
CHAPTER 15

E-mail

The administration of the e-mail module is pretty straight forward. You can setup multiple IMAP accounts for each user. There are however some advanced topics that we’ll describe here.

The e-mail module supports:

- Links
- Sharing

15.1 Creating accounts

Only users with manage permissions for the e-mail module can create e-mail accounts. If you don’t have manage permissions then you can only edit your e-mail address, sender name and signature if you have one pre-configured.

Go to:
E-mail -> Administration -> Accounts
to view your accounts. You can double click one to edit it.

To setup a new account you need some information from your e-mail service provider. You need the following values:

- Your e-mail address
- The hostname of the incoming mail server (IMAP).
- The port (usually 143)
- Your username and password
- Outgoing SMTP server hostname and port

With this information you can create an account easily. Go to:
E-mail -> Administration -> Accounts -> Add
Fig. 1: E-mail module
After that fill in the e-mail address and name that should be associated with this account. If you are an administrator you can also set it up for another user by selecting it here. At last fill in the other values you got from your e-mail service provider. If you get a ‘certificate-failure’ error then tick the ‘Don’t validate certificate’ option.

15.2 Synchronize e-mail

You can synchronize your mail to your desktop or mobile device. Read more about connecting your device here.

15.3 Sharing e-mail accounts

If you want to edit the e-mail account permissions to share an account you must go to:
E-mail -> Administration -> Accounts -> Double click account -> Permissions
Here you can add the users and user groups you want to grant access.

15.4 Secretary

If you want a secretary who handles calendar invitations, you may need to share the e-mail accounts too. It’s important that you share the owner’s e-mail account instead of adding a duplicate mail account. Group-Office uses the e-mail account owner to find the right calendar to store the appointments in.

When you share an account there are three levels:

<table>
<thead>
<tr>
<th>Level</th>
<th>Extra privileges with manage permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>User’s can only view messages</td>
</tr>
<tr>
<td>Read and delegate</td>
<td>Users can view, mark messages as read and reply with their own account. The account owner will be cc’d automatically.</td>
</tr>
<tr>
<td>Manage</td>
<td>The user has full access to the account</td>
</tr>
</tbody>
</table>

15.5 Creating e-mail signatures or templates

You can create multiple signatures with e-mail templates. Templates can be shared too. It’s easy to setup a default e-mail template for the whole organization. You can also include automatic data tags that will be replaced with data from the logged in user or recipient.

To edit or create templates, go to:
My Account -> E-mail -> Templates.
Double click on a template or create a new one to edit it. If you drafted an HTML document in another program, make sure this HTML only contains inline style. Style sections in the head are not supported.

15.6 Images

When you insert an image make sure you don’t copy paste it from another template or web page. Always insert it through the insert image toolbar from the template editor. This way the image will be embedded into the HTML e-mail template and will automatically be sent along with your e-mail messages.
Fig. 2: E-mail template
15.7 Choose the right signature feature

You can’t use this feature together with the plain text signature at E-mail -> Administration -> Accounts -> Double click account row. Either use the simple plain text signature here or the e-mail templates.

15.8 Changing the font

You can’t select a font in the template editor. The font is globally defined in config.php. The administrator can change the default font:

$config['$html_editor_font']="font-size:13px; font-family:Arial, Helvetica, sans-serif;";

15.9 Template variables

You can use the following values in the document:

Note: Variables are typically wrapped in {...} signs but you may also use %.%. This is necessary when using variables inside HTML templates. For example <a href="mailto:%user:email"> only works with % signs.

15.9.1 Custom fields

You can use custom fields like this:

- {project:databaseName}
- {contact:databaseName}
- {company:databaseName}
- etc.

15.9.2 Common fields

- {date} Current date
- {filename} The filename of the document.

Fields of the logged in user

- {user:displayName}
- {user:email}
- {user:username}
- {user:*} Any contact field listed below can be used if this user has a profile (6.4+).
- {usercompany:*} Any company field listed below can be used if this user has a profile (6.4+).
15.9.3 Fields of the contact

- `{contact:sirmadam}` Sir or Madam depending on the gender.
- `{contact:salutation}` The salutation
- `{contact:formatted_address}` Get the full address formatted according to the country standards.
- `{contact:beginning}` Dear sir / madam
- `{contact:first_name}` First name
- `{contact:middle_name}`
- `{contact:last_name}`
- `{contact:initials}`
- `{contact:title}`
- `{contact:email}`
- `{contact:email2}`
- `{contact:email3}`
- `{contact:home_phone}`
- `{contact:fax}`
- `{contact:cellular}`
- `{contact:address}`
- `{contact:address_no}`
- `{contact:zip}`
- `{contact:city}`
- `{contact:state}`
- `{contact:country}`
- `{contact:department}`
- `{contact:function}`
- `{contact:work_phone}`
- `{contact:work_fax}`
- `{contact:homepage}`

15.9.4 Fields of the contact

- `{company:formatted_address}` Get the full address formatted according to the country standards.
- `{company:formatted_post_address}` Get the full address formatted according to the country standards.
- `{company:mtime}`
- `{company:ctime}`
- `{company:crn}` Company registration number
- `{company:iban}`
15.9.5 Project fields

- {project:name}
- {project:customer}
- {project:description}
- {project:ctime} Creation time
- {project:mtime} Modification time
- {project:status}
- {project:type}
- {project:start_time}
- {project:due_time}
- {project:units_budget}
- {project:responsibleUser:name} The manager
15.9.6 Example template for standard letter

{company:name} {company:formatted_address}
Date: {date} About: {filename}
{contact:salutation},
Best regards,
{user:displayName}

15.10 Filtering and out of office

15.10.1 With sieve support

To use advanced filter e-mails, your IMAP server must support ManageSieve and the Sieve e-mail filtering module must be installed and enabled for your account. Also check if the correct sieve port is set at the incoming mail advanced settings. The port is usually 4190 or 2000.

To edit or add e-mail filters go to:
E-mail -> administration -> Accounts -> Double click account row -> Filters

By default there are already a spam filter rule present.

Fig. 3: Filters window with sieve support
**Edit a filter**

Double click on a filter set (A new window opens)

In this window you can add criteria and desired actions for the filter.

![Rules window](image)

Fig. 4: Edit filter rule

**Spam filter**

Most e-mail servers are setup with a spam filter. We use Spamassassin on our hosted services. Spam filters flag messages as spam but you need a mail filtering rule to do something with it. Group-Office creates one by default but if for some reason it isn’t there you can create it at:

E-mail -> Administration -> Account -> Filters.

Click on “Add” and match the settings like in this screenshot:
Fig. 5: Spam filter
Whitelist

It’s also possible to create a white list. Make sure this rule is sorted above the spam filter rule alike in the screenshot above.

Click on “Add” and match the settings like in this screenshot:

![Whitelist filter rule](image)

Out of office

Group-Office allows you to configure an out of office message easily. Go to:

E-mail -> Administration -> Accounts -> Double click your account -> Out office.

Select a period and configure a message and activate the filter.
Chapter 15. E-mail

15.10.2 Without sieve support

When sieve is not supported. Group-Office will fallback on a simpler filtering system. It can only move incoming e-mails to specified folders based on the from, to or subject text.

15.11 Default mail client on your computer

You can setup Group-Office as your default mail client on your macOS, Windows or Linux computer. So when clicking an e-mail address in other applications Group-Office will open with the e-mail compose window. To do this you need to install the Assistant. When it’s installed you can select it in your OS:

15.11.1 Windows

Go to Settings -> Apps -> “Default apps” and select “Group-Office” for mail:

15.11.2 macOS

Go to Mail -> Preferences -> General and select “Group-Office Assistant” as default e-mail reader:

15.11.3 Ubuntu or (other Gnome based Linux OS)

Go to Settings -> Details -> “Default applications” and select “Group-Office Assistant” for mail:
### 15.11. Default mail client on your computer

**Fig. 8: Filters window without sieve support**

<table>
<thead>
<tr>
<th>Field</th>
<th>Contains</th>
<th>Move to folder</th>
<th>Mark as read</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td><a href="mailto:idontwanttoknow@example.n">idontwanttoknow@example.n</a>...</td>
<td>Trash</td>
<td>No</td>
</tr>
</tbody>
</table>

**E-mail Account**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Server</th>
<th>Permissions</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Delete</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fig. 9: Windows 10 default apps
15.11. Default mail client on your computer

Fig. 10: macOS default e-mail reader
Fig. 11: Ubuntu default apps
CHAPTER 16

Calendar

Every user gets a personal calendar. You can schedule meetings and share calendars among users via the web interface, your mobile or desktop computer.

The calendar module supports:

- Links
- Custom fields
- Sharing
- Files

16.1 Sharing

Sharing calendars works similar to the default procedure. By default calendars are only visible to the calendar owner. You can change that at the Default permissions.

But there is something special to note with inviting participants. When someone invites another Group-Office user, the event will be scheduled in a tentative state even though this user does not have permissions for the calendar.

Note: Free busy access module

When the free busy access module is installed, the event won’t be scheduled in the user’s calendar when the user has not been granted free busy access in their settings.

16.2 Views

A view is a calendar with multiple calendars visible in it. Either merged in one regular view or each calendar underneath each other. You can create views in Calendar -> Administration -> Views.
Fig. 1: Calendar module
16.3 Resources

It’s possible to create resources in Group-Office to manage meeting rooms or company cars for example. The resources are organized in groups and have can have administrators per group. Administrators must check and approve resource bookings. The resources are part of the calendar module and also work together with custom fields.

For resource administration this option you need manage permissions for the calendar module.

16.3.1 Creating groups

Groups are useful to organize resources. You could have the following groups for example:

- Meeting rooms
- Cars
- Books
- etc.

To open the resource group management go to:

Calendar -> Administration -> Resource groups -> Add

Enter a name for the group and click at “Apply”. You can also select the custom field sets you want to use with this type of resource. For meeting rooms you might want to create a field set with a check box for lunch, a beamer or a text field for the number of persons.

After hitting apply the “Administrators” tab is enabled. It’s required to add at least one administrator to the group. A group administrator must check and approve a booking.

---

**Note: Permissions for administrators**

Administrators will automatically be granted write permissions for the resources in this group when you add them. Permissions will NOT be automatically removed because this might lead to unwanted deletion of permissions.

---

16.4 Creating resources

After you setup a resource group you can add resources. Go to:

Calendar -> Administration -> Resources -> Add

Enter a name and click on “Apply”. The permissions tab is now enabled.

**Note:** Permissions

Give users that need to book this resource **read permissions**. They don’t need write permissions. If you give them write permissions they will be able to accept their own bookings and that’s probably not what you want.

After at least one resource is created the “Resources” panel in the calendar navigation on the left is available. New bookings are shown in red, accepted bookings turn green:
16.5 Booking a resource

When you have access to a resource you can book them from the new event dialog in the calendar. Go to:

Calendar -> Add -> Resources.

Select the resource(s) you want to use with your event (The lunch, beamer and persons input fields are custom fields in this example):

The event in the users’ calendar shows up with a star:

When the event is saved an e-mail goes out to all administrators so they can accept or decline it.

The administrator must approve the booking unless you are an administrator yourself.
16.6 Accepting or declining a booking

When you are a resource administrator and you get an e-mail like displayed above, you can click on the link to open the booking or you can browse to:

Calendar -> Resources -> The resource

You can change the status to “Accepted” or “Declined” in the dialog. After saving it an e-mail will go out to the user.
CHAPTER 17

Address book

Fig. 1: Address book

The address book is a very powerful tool to manage contacts and organizations. It’s easy to share address books with
The address book module supports:

- Custom filters
- Links
- Custom fields
- Comments
- Sharing
- Files

### 17.1 Sharing an address book

Via the more menu you in the tree you can edit an address book. Read more about sharing here.

### 17.2 Import and export

You can import and export a Comma Separated File (.csv) or a vCard (.vcf). The export functions will export what’s currently selected on screen. So you can first apply searches or filters and then choose export.

These functions are accessible via the more menu in the toolbar above the grid:

![Fig. 2: Import and export menu options](image)

#### 17.2.1 VCF format

VCF or vCard format is the easiest to use. You don’t need to map any fields because it’s already defined in the standard. You can download an example VCF file here.
17.2.2 CSV format

Any format can be imported as long as the required columns are available. After selecting the file a dialog will be opened where you can match the fields:

Fig. 3: Import CSV dialog

When the column names match the name of the Group-Office field name or label it will be mapped automatically. It also recognized other common names and the Outlook format.

The best approach for importing would be:

1. Setup your custom fields
2. Create a dummy contact in Group-Office with everything filled in.
3. Export this contact to CSV so you have an example template.
4. Fill this CSV with data
5. Import it.

You can download an example CSV file here.

Updating existing contacts

It’s possible to bulk update your Group-Office contacts with a CSV file. After uploading your CSV you’ll get a mapping dialog. In the first combo box you can select: “Update existing items by” where you can select “ID” or “E-mail”. When selecting the “ID” for example it will update all Group-Office contacts with a matching “ID” in the CSV record.

17.2.3 Labels

You can easily print labels for envelopes. You can enter the rows, columns and margins and generate a PDF to print labels.
In the template you can use the new *template syntax* that’s also used in the Newsletters module.

### 17.3 Groups

Each address books can have groups. They’re useful for organizing your address book.

### 17.4 Advanced search syntax

Next to *Custom filters* you can also use an advanced search syntax in the address book search bar.

If you enter a string of text it will search through the name and debtor number fields by default. But you can also search on other fields. For example:
modified: > -1 week

Will show all contacts that have been modified in the past week.
Ranges are also supported for date filters:
modified: last week..now

Or:

age: > 30 age: < 40

Will show all contacts with an age between 30 and 40.
Or:

birthday: < 1 months

Will show all contacts with a birthday coming up within one month.
Or:

org: intermesh,group-office

Will show all employees of the organizations Intermesh and Group-Office

You can use these fields:

- name
- email
- phone
- org (Organization name)
- createdat (date)
- createdby (User’s display name)
- modifiedat (date)
- modifiedby (User’s display name)
- commentedat (date)
- age (number)
- gender (M, F or null)
- birthday (date)
- isuser (0 or 1)
- Custom fields By using <databaseName>: query. Lookup the database name in system settings.

17.4.1 String search explained

For example when searching for:

name: John Jacks
It will find anything with “John Jacks*”. Because of the automatic wildcards (*) also “John Jackson” will match for example.

If you want to search for a phrase or without wildcards then you can use quotes:

```
name: "John Jacks"
```
Newsletters

The newsletters module is an easy tool to manage address lists and send out newsletters. Using powerful templates you can personalize your messages.

Fig. 1: Newsletters
18.1 Create lists

Create a list by clicking the add button in the “Lists” toolbar. Then click the add button from the “Contents” toolbar in the middle to add contacts.

You can add contacts one by one but you can also add entire search results using filters.

18.2 Compose

To send a newsletter click the add button from the “Sent items” panel.

![Fig. 2: Compose](image)

18.3 Templates

From the composer you can also manage templates. Via the menu shown in the screenshot above. You can use the following variables in these templates:

18.3.1 Contact

Fields of the recipient
• contact.salutation
• contact.name
• contact.prefix
• contact.firstName
• contact.middleName
• contact.lastName
• contact.suffix
• contact.jobTitle
• contact.gender
• contact.organizations: Array(same properties as contact)
• contact.dates: Array (type, date)
• contact.emailAddresses: Array(type, email)
• contact.phoneNumbers: Array(type, number)
• contact.addresses: Array(type, street, street2, zipCode, city, state, country, countryCode)
• contact.urls: Array(type, url)
• contact.debtorNumber
• contact.IBAN
• contact.vatNo
• Custom fields (The database name prefixed with “contact.customFields.”)

18.3.2 Other

• unsubscribeUrl
• now (Date) The time the message is sent

18.3.3 Creator

Fields of the user who created the newsletter

• creator.displayName
• creator.email
• creator.username

18.3.4 Syntax

A variable is written like this:

```plaintext
{{contact.name}}
```

Or a custom field:
Arrays can be written like this:

```text
{{contact.emailAddresses[0].email}}
```

You can also iterator over arrays:

```text
[each address in contact.addresses]
    [if {{address.type}} == "billing"]
        {{address.formatted}}
    [/if]
[/each]
```

And filter arrays by property and only write first match using “eachIndex”:

```text
[each emailAddress in contact.emailAddresses | filter:type:"billing"]
    [if {{eachIndex}} == 1]
        {{emailAddress.email}}
    [/if]
[/each]
```

But this is probably the best way to handle the case where you prefer a type of address but just use the first if that’s not found. It uses [assign] to create a new variable. If it’s empty it will use the first address:

```text
[assign address = contact.addresses | filter:type:"postal" | first]
[if !{{address}}]
    [assign address = contact.addresses | first]
[/if]
{{address.formatted}}
```

An advanced example for printing a custom salutation (Just an example. You can use {{contact.salutation}}):

```text
Dear [if {{contact.prefixes}}]{{contact.prefixes}}[else][if !{{contact.gender}}]Ms./Mr.[else][if {{contact.gender}}=="M"]Mr.[else]Ms.[/if][/if][/if] {{contact.lastName}}
```

A simple example template:

```text
Hi {{contact.salutation}},

Best regards,

{{creator.displayName}}
{{creator.profile.organizations[0].name}}
```

Finding a contact with id = 1

```text
[assign contact = 1 | entity:Contact]
{{contact.name}}
```

Find the first linked contact:

```text
[assign firstContactLink = someEntityVar | links:Contact | first]
{{firstContactLink.name}}
```
Filters

You can use filters to format data. They can be used with a “|” char followed by the filter name. Optionally the filter can take arguments separated by a “:”.

For example a date:

```ebnf
{{contact.dates[0].date|date:d-m-Y}}
```

A number formatted in Dutch style with 2 decimals:

```ebnf
{{contact.customFields.number|number:2,:.}}
```
In Account settings -> Document templates, you can upload documents to be used as templates. Supported are:

Microsoft Office
- Microsoft Word (docx)
- Microsoft Excel (xlsx)

Libre Office / Open Office
- Open Document Text (odt)
- Open Document Spreadsheet (ods)

In these templates you can use Template variables.
The files module supports:

- Links
- Custom fields
- Comments
- Sharing
- Files

### 20.1 Assistant

The Group-Office Assistant is a small program that you can install on your Windows, MacOS or Linux computer. It will automatically download files opened from Group-Office and monitor it for changes. When the file is saved it automatically uploads it back to Group-Office.

It also assists in mounting a network drive.

Download it for:

- Windows
- MacOS
- Linux

### 20.2 LibreOffice Online

With LibreOffice Online you can edit office documents in your browser. You need a working LibreOffice Online server. More info on [https://www.libreoffice.org/download/libreoffice-online/](https://www.libreoffice.org/download/libreoffice-online/)

For installations read the [LibreOffice online installation page](https://www.libreoffice.org/download/libreoffice-online/).
Setup Docker container for Collabora online
Notes

The notes module is a simple note taking modules. The notes module supports:

- **Links**
- Custom fields
- Comments
- Sharing
- Files
CHAPTER 22

Tickets

With the ticket system you can easily manage the support requests from your customers.

Fig. 1: Tickets module

The ticket system will be used by:
1. Users: users who create tickets when they have a problem.

2. Agent: A user who is authorized to claim and respond to tickets.

A user only needs write permission for ticket types. A user will become an agent when the user is granted manage permissions at Administration -> Types -> Type -> Permissions.

The tickets module supports:

- *Links*
- *Custom fields*
- *Comments*
- *Sharing*
- *Files*

**Note:** Anonymous tickets

A customer does not necessarily need to be a user. Read the external ticket page section for more information.

### 22.1 Types

The first thing you need to do after installing the tickets module is defining the types. Types are used for different purposes:

1. Permissions, you can control which agent or user is allowed to create tickets of this type.
2. Reporting, there are reports about how many tickets per type etc.
3. Configuration, each type can have different settings. For example who gets an e-mail when a new ticket of this type is created.

You could define a separate type for each product for example.
22.1.1 Type settings

The following settings are on the properties tab:

Type group
Groups are useful for organizing the ticket types. In the screenshot you can see two defined groups. You can create them by clicking on the “Manage groups” button from the types tab.

Description
The description is shown in the interface when you hold your mouse over the type in the filter.

Show tickets from other users
When you enable this checkbox, all users with access to this type can see other people’s tickets.

“E-mail to” on new checkbox
You can enter a comma separated list of e-mail addresses here that will receive a notification when a new ticket is created by a user.

Alternate sender
By default the system will send an e-mail from the webmaster e-mail with the Group-Office title as name and the sender is always the same for each type. You can configure the default in the “Options” tab. One address for all types might not always be ideal. Use the alternate sender settings to set the from address per type.

IMAP Mailbox
You can specify a mailbox here for incoming tickets.

22.1.2 Permissions

The type permissions are very important. It defines if a user is an agent or a regular user. If you have “Manage permissions” on a ticket type then you are an agent.

Here’s an overview of the permission levels:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read only</td>
<td>The user can only read his own tickets. Probably not very useful in most cases.</td>
</tr>
<tr>
<td>Create</td>
<td>Users may create new tickets.</td>
</tr>
<tr>
<td>Write</td>
<td>Identical to create permission</td>
</tr>
<tr>
<td>Delete</td>
<td>The user may delete his own tickets</td>
</tr>
<tr>
<td>Manage</td>
<td>The user will be an agent that can see all tickets in this type and can reply to and modify the ticket.</td>
</tr>
</tbody>
</table>

Note: Make sure each user has at least write permission for one type. Otherwise this user will get an access denied error when he attempts to create a ticket.

22.1.3 E-mail to tickets

You can easily configure an e-mail address to accept incoming tickets. Create an e-mail account as administrator that is dedicated for tickets. Then select this e-mail account in the type properties where you want the e-mails to be imported too.
**Warning:** All e-mail in the inbox of this account will be imported into the ticket system and the e-mail will be removed from the IMAP server.

The scheduled task is configured to import the tickets every 5 minutes.

### 22.2 Statuses

Statuses are very simple and straightforward. You can create and delete them at Administration -> Statuses. Statuses can be filtered on and there’s a counter with the number of tickets in that status in the filter.

### 22.3 E-mail templates

Here you can configure the messages that will be sent to your customers.

![Figure 3: E-mail template](image)

There are five template types:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default response</td>
<td>This template is used by default when an agent replies to a customer</td>
</tr>
<tr>
<td>Auto reply</td>
<td>This template is used for auto replies when customers create new tickets</td>
</tr>
<tr>
<td>Ticket created for client</td>
<td>This template is used when an agent creates a ticket for a customer.</td>
</tr>
<tr>
<td>Ticket created for agent</td>
<td>This template is used when a customer created a new ticket. This message will be sent to all e-mail addresses listed in the type settings.</td>
</tr>
<tr>
<td>Ticket claimed notification</td>
<td>This template is for the message that is sent to the customer when an agent claims the ticket.</td>
</tr>
</tbody>
</table>

**Note:** All templates are optional. If you leave the template out this message type will be disabled.
22.3.1 Template variables

The following template variables can be used to show information about the contact that is set in the ticket:

- `{contact:sirmadam}` Sir or Madam depending on the gender.
- `{contact:salutation}` The salutation
- `{contact:formatted_address}` Get the full address formatted according to the country standards.
- `{contact:first_name}` First name
- `{contact:middle_name}`
- `{contact:last_name}`
- `{contact:initials}`
- `{contact:title}`
- `{contact:email}`
- `{contact:home_phone}`
- `{contact:fax}`
- `{contact:cellular}`
- `{contact:address}`
- `{contact:address_no}`
- `{contact:zip}`
- `{contact:city}`
- `{contact:state}`
- `{contact:country}`
- `{contact:department}`
- `{contact:function}`
- `{contact:work_phone}`
- `{contact:work_fax}`
- `{contact:homepage}`

If the contact has a company set then you can also use these variables to show the company information:

- `{company:formatted_address}` Get the full address formatted according to the country standards.
- `{company:formatted_post_address}` Get the full address formatted according to the country standards.
- `{company:mtime}`
- `{company:ctime}`
- `{company:crn}` Company registration number
- `{company:iban}`
- `{company:vat_no}`
- `{company:bank_no}`
- `{company:comment}`
- `{company:homepage}`
• {company:email}
• {company:fax}
• {company:phone}
• {company:post_zip}
• {company:post_country}
• {company:post_state}
• {company:post_city}
• {company:post_address_no}
• {company:post_address}
• {company:country}
• {company:state}
• {company:city}
• {company:zip}
• {company:address}
• {company:address_no}
• {company:name2}
• {company:name}
• {company:id}

The following template variables can be used to show information about the ticket itself:
• {AGENT} The current ticket agent that corresponds with the contact.
• {MESSAGE} The current message that is about to be sent.
• {LINK} Hyperlink to the ticket’s page.
• {ticket:ticket_number}
• {ticket:subject}
• {ticket:company}
• {ticket:first_name} Contact’s first name.
• {ticket:middle_name} Contact’s middle name.
• {ticket:last_name} Contact’s last name.
• {ticket:email} Contact’s email address.
• {ticket:phone} Contact’s phone number.

### 22.4 Options

In the options tab you can configure various options:

**Name**

The default sender name for e-mail messages
E-mail
The default sender address for e-mail messages

Subject A template for the e-mail subject so the ticket number is included. Defaults to the ticket subject.

Note: The subject is always prefixed with [ts #{ticket_number}] because the e-mail importer relies on this format for detecting replies.

Language
The language of the templates. Group-Office will switch to this language when sending messages to customers.

Days for close
You can optionally enter a number of days that will be used to automatically close tickets when a customer does not respond within this period. A scheduled task is configured to perform this task.

Never close tickets in this status
An exception for the above setting.

Check boxes
The check boxes in the bottom of this tab speak for itself.

22.5 Rates

In the rates tab you can define multiple support rates with a name and value. When you setup rates it’s possible to register a number of hours worked on each message you create on a ticket. In the screenshot below you can see how agents can enter rates. Rates can be exported or converted in to invoices when you also use the billing module.
22.6 External ticket page

The external page can be used as an iframe of your website or it can be styled like your website. This feature depends on the “site” and “defaultsite” module. When you enable the page these modules will be installed automatically.

The following settings are available:

**Enable external page**

Enable the public external page. This will install a site in the website manager module.

**Enable for anonymous users**

Allow visitors to create tickets without logging in.

**Use alternative URL**

This URL is used for e-mail messages. Group-Office will send users to the external page rather than Group-Office when you enable this. It’s required for anonymous tickets.

**Custom CSS**

Add your own styling to the external page.

22.7 Example customer ticket page

By default the external page style is very basic. But you can of course customize that.

![Fig. 5: External ticket list](image)
Fig. 6: External ticket page
With the projects module you can easily manage project information in one place and keep track of worked hours, material usage and travel costs. You can also share projects with customers and co-workers. With the project tasks it’s easy for employees and the manager to keep track of the project progress.

The managers also have reports about progress and budgeting.

The projects module supports:

- **Links**
- **Custom fields**
- **Comments**
- **Sharing**
- **Files**

**Note:** Having trouble with making time entries?

Make sure the projects are not having a status that is closed. Also make sure the user account is set as an *employee* and project resource.

## 23.1 Administration

### 23.1.1 Permission types

Permission types control who can access projects in the projects module. You don’t set permissions per project but each project can have a permission type. If you’re a small organization just one permission type might be sufficient. But if you have different departments it could be useful to setup a permission type for each department. It just depends on who may see the project information.

The permission levels are straight forward and you can read more about it [here](#).
Fig. 1: Projects module
Note: Root level

Only users with manage permission on the projects module are allowed to create projects on the root level. To create sub projects you need at least create permission on the parent project.

Finance permissions

Here you can control which users can see financial data like budgets and actual costs. Also the billing button to create invoices from time entries is only available to those users.

23.1.2 Templates

Project templates are very powerful. You can control what information the end user sees in the project information panel and edit dialog. The templates control:

1. Which standard fields are enabled on the project dialog.
2. Which custom field categories are enabled on the project dialog
3. Default project files and folders
4. Default project tasks
5. Default permission type and status
6. The icon displayed in the project tree and grid
7. Enable / disable the jobs panel

![Fig. 2: Project template](image)

Properties

The following properties can be configured:
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the template</td>
</tr>
<tr>
<td>Type</td>
<td>The template type. Projects with a container type template have no time tracking and expenses functionality.</td>
</tr>
<tr>
<td>Status</td>
<td>The default status of the project</td>
</tr>
<tr>
<td>Permission type</td>
<td>The default permission type of the project</td>
</tr>
<tr>
<td>Enabled fields</td>
<td>Controls which standard fields are enabled on the project dialog.</td>
</tr>
<tr>
<td>Icon</td>
<td>An icon that will be disabled in the tree and grid</td>
</tr>
</tbody>
</table>

**Jobs**

On this tab you can define default project:

- Tasks
- Sub projects
- Jobs

On the name and description fields you can use template tags like:

- `{project:name}`
- `{customer:name}`
- `{contact:name}`

You can use additional *Template variables*.

**Permissions**

On the permissions tab you can control which user may use the template.

**23.1.3 Office times**

Set the office times here. These times will be used for guessing the default time for new time entries and also for estimating a project task due date.

**23.1.4 Statuses**

Manage project statuses here. Per status you can set if a project is completed or not if it’s visible in the tree.

**Note:** Missing projects in the tree or for time entries? Completed projects won’t be available for time tracking anymore. You can also set if the project is visible in the tree. This is useful for keeping your projects overview organized.
23.1.5 Activity Types

Activity types will be available in a drop down when making time entries. It also has a preset for the description and duration.

It’s also possible to exclude hours from billing or to setup different fees per activity type.

23.1.6 Employees

Every user that needs to be set as a resource for time tracking on projects need to be added here with their default fee. When a user closes it’s time tracking week the closed date will be set forward. In this dialog you can set it back if a mistake was made and the week needs to be opened again.

![Fig. 3: Employees](image)

To make the projects available for the time tracking module of a user you’ll also need to define for each project which employees are working on it. To do that, you’ll need to open the edit dialog of a project and go to the “Employees” tab. Add every employee for that project in this screen.

![Fig. 4: Project resources / employees](image)
23.1.7 CSV Import and Export

It is entirely possible to manage projects by exporting them to CSV or importing them from CSV. Please note that by default, this is only to be done by admin users or other users with the ‘manage’ permission.

CSV Export

Exporting your current projects is done from the menu above the projects grid:

A dialog window is opened with export options and a tab in which to select the columns to export. Upon clicking the save button, a CSV file should be generated and saved.

**Note:** Please note that if you have work with parent projects and child projects, only the projects will be exported that are on the current level in the hierarchy and that are not project folders.

CSV Import

Likewise, you can import projects from CSV from the menu above the projects grid. Upon clinking the import option, a new dialog is opened:

Fig. 5: Projects import dialog with an already uploaded file
In the CSV, the first row should contain the column names. For an example, you can download an example CSV file from the import dialog.

In order for a project import to be successful, the following columns should exist in the CSV file:

1. **name** or **path**. The name can be distilled from the path. If the path is omitted, a project will always be imported into the top level of the projects hierarchy.

2. **type_id** or **type_name**. We should know which project type a project is.

3. **status_id** or **status_name**. Defines the project status.

4. **template_id** or **template_name**. Defines the project template.

If one of the conditions above is not satisfied, the import script will return an error message.

**Note:** In order to import a hierarchical structure of projects, be sure to include the **path** or **parent_project_id** column. Each level consists of the project name by level, concatenated with forward slashes, e.g. `Project/Subproject/Subsubproject`. If you want to import a large number of subprojects, for which the parent project ID fields are known, you can include the column **parent_project_id** instead. If both columns are included, the **parent_project_id** column will be ignored.

### 23.2 Reports

Group-Office comes with some reports that are useful for most businesses. But reports can also be customized if you have PHP knowledge. The Group-Office reports are available in source code. Intermesh also offers report customization. Contact us for more information about this.

The default reports are:

1. Time tracking entries in CSV
2. Project information and planning in PDF
3. Time entries in PDF
4. Overview of all projects in PDF

For more information about customizing reports you may want to read our [blog post about it](http://groupoffice.blogspot.com/2014/02/introduction-with-new-projects-version.html)

### 23.3 Time tracking

For users to be able to use time tracking the following conditions must be met:

1. The user has access to the time tracking module
2. The user is a resource of a project that does not have a [container](#templates) type template.
3. The project is not in a [closed status](#statuses).

Users don’t need access to the projects module to use time tracking.

**Note:** Closing weeks
Fig. 6: Time tracking
Users can close weeks. When a week is closed the manager can approve the hours and the user can’t edit the time entries anymore.

When a user accidentally closed a week the manager can reopen it at Administration -> Employees. Double click an employee and set the close date back in time.

### 23.4 Jobs

In the jobs panel you can define smaller parts of the project. You can also select these in the time entry dialog to keep track of how much time these jobs took.

Before you can use jobs you must setup employees for the project. Otherwise the jobs panel will stay disabled.

![Project jobs](image)

**Fig. 7: Project jobs**

### 23.5 Hours approval

Project managers can approve hours if this module is installed. The “Manager” property of projects is important here. Users can approve all time entries of projects that they are the manager of. Only hours that are closed will show up for approval. So users must first press the “Close week” button in the time tracking module.

Only approved hours will be billed to the billing module when you use this function.

### 23.6 Billing

It’s possible to generate invoices from your time tracking. To do so you have to create a project and set the type to “Post calculation”. When time is recorded now the fees will be stored too.
The default fees can be set at Administration -> Employees. When you create new projects these values are copied to the project employees when you add them.

In the detail view of the project you see a summary of hours:

1. Booked: All the entries.

2. Billable: Billable hours are the ones that have a fee. You can have no fee when the project is not set to “Post calculation”. You can also set “Not billable” at activity types in administration. It’s also not billable when you set zero fee for the employee in the project.

3. Billed: An income was generated for these time entries.

<table>
<thead>
<tr>
<th>Username</th>
<th>Booked</th>
<th>Billed</th>
<th>Billable</th>
<th>Budgeted units</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Administrator</td>
<td>9,25</td>
<td>0,00</td>
<td>9,25</td>
<td>0,00</td>
</tr>
<tr>
<td>Elmer Fudd</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>16,00</td>
</tr>
<tr>
<td>Demo User</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>100,00</td>
</tr>
<tr>
<td>Linda Smith</td>
<td>0,00</td>
<td>0,00</td>
<td>0,00</td>
<td>16,00</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td>9,25</td>
<td>0,00</td>
<td>9,25</td>
<td>132,00</td>
</tr>
</tbody>
</table>

Fig. 8: Time entries summary in project detail view

### 23.6.1 Generate invoices

To generate an invoice go to Projects -> Financial -> Post calculation. Select a time period and projects you want to invoice and click “Continue”. An invoice will be stored in the project.

If you have the billing module installed a popup will appear to also generate an invoice in the billing module.

**Note:** We plan to integrate the billing and projects more in GO 6.5 so this will become one.
Using the holidays module, you can manage the leave hours of employees. Every year, an amount of available leave hours (year credit) can be set per user. In this module, every employee can be given a total year credit of leave hours to spend in a year. Every subsequent month, 1/12th of this available credit is added to the spendable credit.

Specifically, for every user and every year, the following statistics are tracked:

- **Leftover**: The number of leave hours that were not used in previous years.
- **Yearly credit**: The year credit (available leave hours) for the selected year.
- **Used**: The leave hours that are spent in the selected year.
- **Built up**: The built up credit \( = \text{LEFTOVER CREDIT} + \text{YEAR CREDIT} \times \text{(current month’s number/12)} \)
- **Available year**: The total available leave hours for the selected year \( = \text{LEFTOVER CREDIT} + \text{YEAR CREDIT} - \text{SPENT CREDIT} \)
- **Available now**: The available leave hours for the current month \( = \text{LEFTOVER CREDIT} + \text{YEAR CREDIT} \times \text{(current month’s number/12)} - \text{SPENT CREDIT UNTIL NEXT MONTH} \)

To edit the yearly credit, double-click on a row in the main grid ‘Holiday Hour Credits’.

### 24.1 Working week

In the *user’s preferences*, you can now set the standard amount working hours per day for users. This will be used to auto-calculate the number of used leave hours when you edit/create a holiday entry. You can always change the number of used leave hours of a holiday.

Note that the working week is by default for all users in the Holidays module: 8 hours on Monday thru Friday, 0 hours on Saturday and Sunday.
Fig. 1: Holidays module
24.2 Manager

For each employee you can set a manager. The manager will get a request to approve the holidays by e-mail when an employee adds new holidays.

**Note:** Managers will need manage permissions for the module so they can see all employees hours and approve them.
With the billing module you can create quotes, orders and invoices. The documents can either be printed or you can send them by e-mail as a PDF attachment. You can also enter your expenses. With the report tool you can easily see your income and expenses. You can create as many books as you want and you can call them whatever you like.

The billing module supports:

- Links
- Custom fields
- Comments
- Sharing
- Files

### 25.1 Languages

The billing module is multilingual. If you’re going to use more than one language is best that you start with configuring these because then you can input all translations directly when configuring statuses, products and templates.

Enter the languages at:

Administration -> Languages

### 25.2 Book properties

The books contain invoices, quotes or orders. Per book you can configure things like tax, order numbers, statuses and templates. You can set the following book properties:
Fig. 1: Billing module
Fig. 2: Book dialog
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the book</td>
</tr>
<tr>
<td>Order prefix</td>
<td>The prefix of order numbers.</td>
</tr>
<tr>
<td>Order no. length</td>
<td>The length of the number. eg. 1 becomes 00001 if this is set to 5.</td>
</tr>
<tr>
<td>Last ID</td>
<td>The last ID used for an invoice number. You can change this in case you want to start with a higher number</td>
</tr>
<tr>
<td>Tax</td>
<td>The default Tax percentage you apply to invoice items.</td>
</tr>
<tr>
<td>Currency</td>
<td>The currency symbol</td>
</tr>
<tr>
<td>Country</td>
<td>The country of the company. This is important for tax calculations in the European Union.</td>
</tr>
<tr>
<td>BCC</td>
<td>A copy of all e-mails sent to customers will be sent to this address.</td>
</tr>
<tr>
<td>E-mail address</td>
<td>By default the Group-Office webmaster email address will be used. Override that here.</td>
</tr>
<tr>
<td>E-mail name</td>
<td>By default the Group-Office title will be used. Override that here.</td>
</tr>
<tr>
<td>Call after days</td>
<td>When you set a number here a task will be created after x days to remind you with a call.</td>
</tr>
<tr>
<td>Allow deletion of items</td>
<td>Check to prevent deleting of items even by administrators.</td>
</tr>
<tr>
<td>Use fixed address book</td>
<td>Limit customer selection to this address book.</td>
</tr>
</tbody>
</table>

### 25.2.1 Order ID prefix

By default GO creates a number like Q200900001 for a quote and I200900001 for an invoice.

The following automatic tags can be used:
- %y will be replaced by the full 4 digit year.
- %m will be replaced by the 2 digit month number
- %r will be replaced by a random digit between 0 and 9. This variable can be used multiple times.
- {autoid} will be replaced by the automatic generated id in the database. When not used the ID is appended to the number.

### 25.2.2 PDF templates

Here you can setup templates for the invoice, quotes or PDF documents. You can change your address details and setup the invoice items table here.

PDF templates also support a background image or base PDF to use as stationery paper.

Download an example PDF document

**Creating a custom PDF script**

Sometimes the default PDF does not suit your needs. In that case you can program your own PDF creator script. All you need to do is copy modules/billing/Pdf.php and put the path to the new file in config.php like this:

```php
$config['billing_pdf_class'] = '/any/path/to/Pdf.php';
```

### 25.3 ODF Templates

You can also create ODF templates for invoice items. The benefit is that you can edit the documents after generation.
You can also create odt templates so that you can edit invoices after creating them. You can use the same variables as for e-mail templates.

Download an example ODF document

### 25.4 Statuses

Order statuses need to be setup as well. Of course status determine the state of the invoice or quote, but it also determines the layout of the invoice document and the e-mail message to the customer.

On the status properties you have the following options:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the status (for each language).</td>
</tr>
<tr>
<td>PDF template</td>
<td>The PDF template for this status</td>
</tr>
<tr>
<td>ODF template</td>
<td>The ODF template for this status</td>
</tr>
<tr>
<td>E-mail template</td>
<td>The e-mail template for this status</td>
</tr>
<tr>
<td>Required status</td>
<td>If you set status here, the invoice can only get this status when it has this required status in its history.</td>
</tr>
<tr>
<td>Max order age</td>
<td>When the invoice is older than this age it will turn red in the invoice grid.</td>
</tr>
<tr>
<td>Order is not paid in this status</td>
<td>For automatically setting the payment date when the status changes</td>
</tr>
<tr>
<td>Remove from stock</td>
<td>Decrease the product stock when this status is set</td>
</tr>
<tr>
<td>Make order read only</td>
<td>Nobody can change this invoice when you check this</td>
</tr>
<tr>
<td>Color</td>
<td>Color used in the grid</td>
</tr>
<tr>
<td>Bill extra item</td>
<td>You can automatically add a new item to the invoice when you set this status. For example extra costs when it’s over due</td>
</tr>
</tbody>
</table>

#### 25.4.1 Permissions

If you don’t want to allow anyone to set this status. You can configure that here.

#### 25.4.2 E-mail templates

In the e-mail template per status you can use the following variables:

**Order data**

- `{company_id}` The company id
- `{contact_id}` The contact id
- `{id}` The database id of the order
- `{order_id}` The textual order ID
- `{po_id}` The purchase order ID
- `{btime}` The date of the order
- `{due_date}` Due date for payment
- `{due_days}` Days left for payment
• {dtime} Delivery date
• {reference} The order reference
• {total} The gross amount of the order
• {subtotal} The nett amount of the order
• {vat}
• {total_paid} Amount paid
• {total_outstanding} Amount outstanding (Since 6.4.75)

**Custom fields**

Lookup the database name in the custom fields module and use {order:databaseName}.

**Customer**

• {customer_salutation}
• {customer_name}
• {customer_address}
• {customer_address_no}
• {customer_zip}
• {customer_city}
• {customer_state}
• {customer_vat_no}
• {customer_country}
• {customer_countryname}

**Logged in user**

For the logged in user fields see this page: Template variables

**Invoice items**

You can generate a table of the invoice items like this:

```
{items_table_start}
<table style="width: 100%;">
<thead>
<tr>
<td><strong>Description</strong></td>
<td style="text-align: center;"><strong>Amount</strong></td>
<td style="text-align: right;"><strong>Unit price</strong></td>
<td style="text-align: right;"><strong>Price</strong></td>
</tr>
</thead>
<tbody>
</tbody>
</table>
```
Invoice item rows in ODF templates

You can use these tags in item rows:

- `{amount_delivered}`
- `{markup}`
- `{cost_code}`
- `{discount}`
- `{vat}` VAT Percentage. eg. 19% for 19%
- `{amount}`
- `{unit_total}` The unit price including VAT.
- `{unit_list}` The unit price from the catalog.
- `{unit_price}` The unit price without VAT
- `{unit_cost}` The unit cost price without VAT
- `{description}`
- `{item_total}` The localized item total (amount*unit_total) incl. VAT
- `{item_subtotal}` The localized item total (amount*unit_price) excl. VAT

25.5 Cost codes

Cost codes can be set per invoice row. They can be exported with a report for the accountant.

25.6 Purchase order books

Purchase order book makes it behave a different. It enables a “Stock” button where you can purchase new items for products in your catalog. It queries the products that are lower in stock than the minimum stock value of the product. When you order these items it will generate purchase orders for the product suppliers.

It will also enable deliveries in the purchase orders so you can keep track of how many items have been delivered to you.
25.7 Custom fields

You can use custom fields for orders. There’s one special thing to note. When you select a customer from the address book all fields with matching database names will be copied to the invoice.

Note: The match used to be made on the field label. Since 6.4 the match is made by database name.
With the file search module you can deep search the contents of files. This module in combination with custom fields and the quick edit pane makes the perfect E-Discovery solution.

Group-Office can index the following file types:

- Microsoft Office Documents
- Open Document format (Open Office, LibreOffice)
- Saved E-mails including attachments. It does not search an IMAP server.
- PDF
- Plain text
- Scanned images using OCR

Next to the regular search it’s also possible to create complex queries with the advanced search.

## 26.1 Indexing

File uploads are not indexed straight away. A schedule task is defined that will run every night at 1:00 am. If you want to run it more often you can adjust the “Filesearch index” task at Start menu -> Manage system tasks.

If you want to index directly after upload. You can put this in config.php:

```
$config['filesearch_direct_index'] = true;
```

**Warning:** This may cause great delays on uploads. We don’t recommend using this setting.

**Note:** If indexing is not working you might need to install some additional tools. See the installation instructions.
Fig. 1: File search module
Since version 6.4.145 Group-Office can integrate with Rocket Chat. With rocket chat you can chat with your team in different channels, share files and even start a video conference from your desktop, phone or tablet!

For installation see Rocket Chat in the system settings page.

To use Rocket.Chat install a client or use your browser and use the Rocket.Chat URL obtained from your system administrator. On the login screen press the “Login with Group-Office” button.
Fig. 1: Rocket Chat
The studio module allows administrator users to quickly create their own modules. It generates a number of files in the `www/go/modules` subdirectory. This module is specifically aimed at developers and resellers who wish to add non-standard functionality within GroupOffice.

### 28.1 Prerequisites

The studio module has no particular dependencies from other modules. It is part of the professional package and will run in GroupOffice versions 6.4 or newer.

We do encourage developers to tinker with the code. If you wish to do so, you need a bit of Javascript and PHP knowledge. Please refer to the development section for more information on how to program for GroupOffice.

Out of the box, a module will have plenty of functions and options though.

**Note:** Please make sure that the ownership for the `www/go/modules` subdirectory is set to the Apache or nginx user. In the case of Debian or Ubuntu, this is commonly `www-data`. The `www-data` user must have write permissions in the `www/go/modules` subdirectory. This is commonly NOT the case. So you must run:

```
chown www-data:www-data /usr/share/groupoffice/go/modules
```
28.2 Installation

You can find the studio module in Business.

28.3 Workflow

This workflow is to be accompanied by a fictional example of Intermesh lunch management.

Creating a module is done in a few steps.

1. Open the studio module and click on the ‘Plus’ button on top of the grid;
2. Fill in the module data;
3. Add custom fields;
4. Add ACL data;
5. Enter frontend options;
6. Confirm your choices

28.3.1 Create a Module

The following fields need to be entered. This step can be run only once for each module, so make sure that you enter the correct data:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the module</td>
</tr>
<tr>
<td>Package</td>
<td>The package name. This allows you to group your modules.</td>
</tr>
<tr>
<td>Description</td>
<td>A description to be displayed in the modules manager</td>
</tr>
<tr>
<td>Sort order</td>
<td>Makes modules sortable by your own preference</td>
</tr>
<tr>
<td>Entity name</td>
<td>The name of the database object.</td>
</tr>
<tr>
<td>ACL Entity</td>
<td>Enables access control for your entities</td>
</tr>
</tbody>
</table>

Upon finishing this step, a database migration script will be generated for the module, a number of PHP files is generated and the module is saved into the `www/go/modules/PACKAGE/NAME/` subdirectory. Subsequently, the module is installed directly.

**Warning:** Trying to install a studio module into business or community generates an error message.

**Warning:** It is not possible to save two different entities with the same name.

28.3.2 Add Custom Fields

A feature of studio is that almost every field is stored as a custom field. After all, the entire module is custom! This works in exactly the same way as for standard modules.
Fig. 1: Add a new module
### Custom fields for Lunch

<table>
<thead>
<tr>
<th>Name</th>
<th>Database name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dish data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dish</td>
<td>Dish</td>
<td>Text</td>
</tr>
<tr>
<td>Is vegan</td>
<td>is_vegan</td>
<td>Yes or no</td>
</tr>
<tr>
<td>Needs cooling</td>
<td>Needs_cooling</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Preparation date</td>
<td>Preparation_date</td>
<td>Date</td>
</tr>
<tr>
<td>Microwavable</td>
<td>Microwavable</td>
<td>Checkbox</td>
</tr>
</tbody>
</table>

**Fig. 2: Add custom fields to the entity**
Note: If you want to present data of a linked item. Read the template field section on the custom fields page.

28.3.3 Permissions

The permissions for studio modules work are straightforward and work in a standard way.

28.3.4 Model options

Since the database objects are saved as entities, one can enter code generation options for the model:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Name Field</td>
<td>Generates the search name for a certain entity</td>
</tr>
<tr>
<td>Search description fields</td>
<td>Generate search descriptions based on fields.</td>
</tr>
<tr>
<td>Auto expand field</td>
<td>Configure the name of the column to be automatically expanded in the grid</td>
</tr>
</tbody>
</table>
Fig. 4: Configure the model class
28.3.5 Frontend options

This step enables you to generate the actual screens and is arguably the most interesting step for the end user.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display detail panel</td>
<td>Toggles the right panel within the grid</td>
</tr>
<tr>
<td>Display search bar</td>
<td>Toggle search bar on top of the grid.</td>
</tr>
<tr>
<td>Display totals</td>
<td>Toggles the sum of total items in the grid</td>
</tr>
<tr>
<td>Enable CSV</td>
<td>Toggles the CSV import and export functions</td>
</tr>
<tr>
<td>Toggle ID</td>
<td>Toggles the display of the ID in the grid</td>
</tr>
<tr>
<td>Show creator and modifier</td>
<td>Toggles the display of creation and modification information in the grid</td>
</tr>
<tr>
<td>Display links</td>
<td>Enables or disables :ref:&lt;links&gt; on top of the detail panel</td>
</tr>
<tr>
<td>Display comments</td>
<td>Enables or disables comments in the detail panel</td>
</tr>
<tr>
<td>Display file uploads</td>
<td>Enables or disables file uploads from the detail panel</td>
</tr>
<tr>
<td>Display modification info</td>
<td>Toggles creation and modification date in the detail panel</td>
</tr>
</tbody>
</table>

Note: Frontend options are saved in a separate Javascript file: www/go/modules/PACKAGE/MODULE/views/extjs3/ModuleConfig.js If you want to toggle certain frontend options, you can directly edit this file, thereby...
preventing any overwrites of custom code.

28.3.6 Confirmation

![Configure module lunches, package intermesh](image)

Fig. 6: “Now I am become Shiva, destroyer of worlds” (R. Oppenheimer)

**Warning:** This is the point that makes you want to evaluate your choices. You will now overwrite any generated code. If you have any self-written code in your module, you will lose it unless you have a backup ready.

The ‘Enable module’ will make the module available for your end users. You may want to disable it if you wish to do some custom editing. In that case, you can re-enable it from the System Settings.

You can disable Studio for your module by ticking the ‘Lock code generation’ checkbox. This is particularly useful for manual changes to the code. Upon saving, a studio module can not be regenerated.

In order to finish the workflow, you **must** check the ‘Overwrite existing code’ checkbox. This will re-generate all code and throw away any and all manual additions to the code. By clicking finish, code generation will be initialized.

**Note:** Removing a studio record will **not** delete the module. It will merely disable the regeneration of code in a very
28.4 The Generated Screens

28.4.1 Grid

![Fig. 7: A newly generated grid with one item](image)

The overview of objects of a certain type in a table is called a grid. By default, a generated grid looks like below:

28.4.2 Detail panel

Single items are displayed right of the grid in a so-called detail panel. Please note that you can disable the detail panel. In that case, you can only display items in a grid.

28.4.3 Dialog

The dialog for editing items looks like the screen shot above.

28.5 Deleting a Generated Module

Everybody likes experimenting. The Studio module is particularly inviting. We know!

If you need to entirely clean up your generated code, there’s two steps to be taken.

1. Remove the module from Group Office:
   1. Log in as administrator;
   2. Right click the module and select the delete option;
   3. Click the save button on the bottom right;
2. Physically delete the subdirectory with the generated code.

**Warning:** You **will** encounter error messages if you perform the second step without performing the first step. In certain cases, you will not be able to log into GroupOffice anymore.
Fig. 8: The optional detail panel

Dish data
- Dish: Bread with peanut butter
- Preparation date: 04-09-2020
- Is vegan: Yes

Comments

Files
- Drop files here

Info
- Created: 04-09-2020 12:05
- Modified: 04-09-2020 12:05
Fig. 9: The dialog for editing items.
Group-Office comes in many languages thanks to our community!

At the moment we have the following:

- Arabic
- (Bangladesh)
- Català
- Chinese Simplified
- Chinese Traditional
- Čeština
- Dansk
- Deutsch
- English/American
- English/British
- Español
- Estonian
- Ελληνικά
- Francais
- Hrvatski
- Italiano
- Magyar
- Nederlands
- Norsk bokmål
29.1 Contribute

We’ve made it easy for you to contribute to translations. You can download all language as a spreadsheet. Make the translations and send it to support@intermesh.nl. Then we can import it into the project and include it in the next release.

You can find the download spreadsheet button next to the language selection in System Settings -> General.

**Note:** If you want to import the language CSV file your self you can run:

```bash
cli.php community/dev/Language/import --path=lang.csv
```

29.2 Customize

It’s possible to customize language by overriding the default language string. For example if you want to rename “Address book” to “Contact” you create this file in the Admin personal file area:

`language/legacy/addressbook/en.php`

So this is `language/<MODULE PACKAGE>/<MODULE NAME>/<LANGCODE>.php`

Enter this PHP code in the file:

```php
<?php
return [
  "Address book" => "Contacts"
];
```

After placing the file run `/install/upgrade.php` to rebuild the server cache. Then check if the address book tab text changed into “Contacts”.


This part is for developers that want to build Group-Office modules or integrate other systems.

30.1 Building a server module

In this tutorial we’ll show you how to build a Group Office module. As an example we’re going to build a Music module.

Group Office is a JMAP-based server API and a webclient. We’ll start with implementing the JMAP server API.
30.1.1 Development environment

If you haven’t got your development environment set up, then please do this first.
You can install Group-Office like described in this manual and get started or use our Docker compose project that installs our images for development:
https://github.com/Intermesh/docker-groupoffice-development

30.1.2 Required software

To follow this tutorial you need the following software installed:

1. 
git. For version management.
2. An editor to edit PHP and Javascript files.
3. A HTTP client like Postman for testing the backend API without the User Interface.

30.1.3 Code standards

When writing code we following standards:

1. Use tabs to indent code
2. Use braces with all structures
3. Don’t use ?> close tag at the end of class files
4. One class per file
5. YAGNI

30.1.4 Naming conventions

<table>
<thead>
<tr>
<th>Properties</th>
<th>lowerCamelCase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>lowerCamelCase</td>
</tr>
<tr>
<td>Constants</td>
<td>UPPER_UNDERSCORED</td>
</tr>
<tr>
<td>Database tables</td>
<td>lower_underscored (For windows compatibility) and singular eg. “contact” and not “contacts”</td>
</tr>
<tr>
<td>Database fields</td>
<td>lowerCamelCase</td>
</tr>
<tr>
<td>Namespaces</td>
<td>lower_underscored</td>
</tr>
</tbody>
</table>

Note: We’re currently refactoring the whole code base. So you will encounter the namespaces “go” and “GO”. The “GO” namespace with capitals is old and you should not use it in new code. Legacy modules are found in “modules” folder and new modules are in “go/modules/<package>/<name>”.

30.1.5 Server module

The code for the module can be found at https://github.com/Intermesh/groupoffice-tutorial
The server modules are created in the following path:
The package is a group of modules that belong to each other. It is used to group modules per type or per customer. So our music module will be created in:

"go/modules/tutorial/music"

**Database**

Start with creating the database tables. The tables would be prefixed with the module name. For example “music_artist”.

Create the tables by importing this SQL into your database:

```sql
CREATE TABLE `music_album` (  `id` int(11) NOT NULL,  `artistId` int(11) NOT NULL,  `name` varchar(190) COLLATE utf8mb4_unicode_ci NOT NULL,  `releaseDate` date NOT NULL,  `genreId` int(11) NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `music_artist` (  `id` int(11) NOT NULL,  `name` varchar(190) COLLATE utf8mb4_unicode_ci NOT NULL,  `photo` binary(40) DEFAULT NULL,  `createdAt` datetime NOT NULL,  `modifiedAt` datetime NOT NULL,  `createdBy` int(11) NOT NULL,  `modifiedBy` int(11) NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `music_genre` (  `id` int(11) NOT NULL,  `name` varchar(190) COLLATE utf8mb4_unicode_ci NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

INSERT INTO `music_genre` (`id`, `name`) VALUES  (1, 'Pop'),  (2, 'Rock'),  (3, 'Blues'),  (4, 'Jazz');

ALTER TABLE `music_album`  ADD PRIMARY KEY (`id`),  ADD KEY `artistId` (`artistId`),  ADD KEY `genreId` (`genreId`);

ALTER TABLE `music_artist`  ADD PRIMARY KEY (`id`),  ADD KEY `photo` (`photo`);

ALTER TABLE `music_genre`
(continues on next page)
ADD PRIMARY KEY (`id`);

ALTER TABLE `music_album`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=14;

ALTER TABLE `music_artist`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

ALTER TABLE `music_genre`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

ALTER TABLE `music_album`
ADD CONSTRAINT `music_album_ibfk_1` FOREIGN KEY (`artistId`) REFERENCES `music_artist` (`id`) ON DELETE CASCADE,
ADD CONSTRAINT `music_album_ibfk_2` FOREIGN KEY (`genreId`) REFERENCES `music_genre` (`id`);

ALTER TABLE `music_artist`
ADD CONSTRAINT `music_artist_ibfk_1` FOREIGN KEY (`photo`) REFERENCES `core_blob` (`id`);

Database rules

1. Also see naming conventions.
2. Date’s use column type “DATE”.
3. Date and time columns use type “DATETIME”.
4. Foreign key’s must be defined for relationships. Think about cascading delete set to null or restrict. In general. Properties should always be cascaded and entities should be restricted. They should be cascaded by overriding the internalSave() function so all the application logic will be executed like cleaning up links, logging etc.
5. We often choose a varchar to be 190 characters so it can be indexed on all database versions.
6. Columns modifiedBy (int), createdBy (int), createdAt (DATETIME), modifiedAt (DATETIME) are automatically set by Group Office.

30.1.6 Code generator

We’ve written a command line tool to make it easy to start with a new module. When you’ve created your database tables then you can run it to generate the models and controllers. You need to install the “Development tools” module that is part of the “Community” package. Login as administrator and go to System Settings -> Modules.

Then you can run it at any time from within the project directory to add new model properties, models or controllers:

```
php {PATH_TO_GROUPOFFICE}/cli.php community/dev/Module/init --package=tutorial --name=music
```

Note: When using docker-compose use: command with:

```
docker-compose exec groupoffice-master php /usr/local/share/groupoffice/cli.php
  --community/dev/Module/init --package=tutorial --name=music
```
the command should output:

```
Generating model/Album.php
Updating go\modules\community\music\model\Album with new properties
Generating model/Artist.php
Updating go\modules\community\music\model\Artist with new properties
Generating model/Genre.php
Updating go\modules\community\music\model\Genre with new properties
Done
```

This will generate:

1. Module.php, required for every module. Contains Author info and controls the installation.
2. views/extjs3, The webclient code. We’ll get to that later.
4. install/install.sql, uninstall.sql and updates.php, these files handle installation and upgrading.
5. model, this folder contains all models.

**Note:** Docker runs as root and will write these files as root.

So you need to change the ownership to your own user by running:

```
sudo chown -R $USER:$USER src/master/www/go/modules/tutorial
```

### 30.1.7 Property and Entity models

You can read more about entities and properties [here](#). By default, the tool generates only “Property” models. It doesn’t know which models should be “Entities”. An entity can be modified by the API directly and a property is only modifiable through an entity. For example an email address of a contact is a property of the entity contact.

So the first step is to change some properties into JMAP entities. In this example Artist and Genre are entities.

So in `model/Artist.php` change:

```php
use go\core\orm\Property;

class Artist extends Property {
```

Into:

```php
use go\core\jmap\Entity;

class Artist extends Entity {
```

Do the same for Genre.

Now run the code generator tool again and it will generate controllers for these entities. It should output:

```
Generating controller/Artist.php
Generating controller/Genre.php
Done
```
Relations

Now we must define relations in the models. Add the “albums” relation to the artist by creating a new public property:

```php
/**
 * The albums created by the artist
 * @var array
 */
public $albums;
```

And then change the mapping:

```php
protected static function defineMapping() {
    return parent::defineMapping()
        ->addTable("music_artist", "artist")
        ->addArray('albums', Album::class, ['id' => 'artistId']);
}
```

**Note:** When making changes to the database, model properties or mappings, you must run `http://localhost/install/upgrade.php` in your browser to rebuild the cache.

Custom properties

For the sake of learning, we will add a property to our model, that does not directly stem from its own table. In this example, we want to display the number of albums for a certain artist in the artist grid.

The first step is to declare the property in the model and to implement a getter:

```php
/** @var int */
protected $albumCount;
// (...)
public function getAlbumCount() :int
{
    return $this->albumCount;
}
```

The next step is to update the `defineMapping` method, to actually count the albums:

```php
protected static function defineMapping() {
    return parent::defineMapping()
        // The artistId property is simply retrieved by counting the albums that are related to the current artist.
        ->addTable("music_artist", "artist")
        ->addMap('albums', Album::class, ['id' => 'artistId'])
        ->setQuery((new Query())->select('COUNT(alb.id) AS albumCount')
            ->join('music_album', 'alb', 'artist.id=alb.artistId')->groupBy(['alb.artistId']));
}
```

The `albumCount` property is simply retrieved by counting the albums that are related to the current artist.
Note: Please note that in this example, we do not define a setter method. It should not be possible to manually set an album counter, since it is a counter of related properties.

There is actually a better way of getting an album count. In order to keep things simple, one can remove the extra query and redefine the \texttt{getAlbumCount()} function as follows:

```php
public function getAlbumCount() : int
{
    return count($this->albums);
}
```

30.1.8 Translations

To define a nice name for the module create \texttt{language/en.php}:

```php
<?php
return [
    'name' => 'Music',
    'description' => 'Simple module for the Group-Office tutorial'
];
```

If you’re interested you can read more about translating \textit{here}.

30.1.9 Install the module

Now we’ve got our basic server API in place. Now it’s time to install the module at: System Settings -> Modules. There should be a Tutorial package with the “music” module.

Check the box to install it.

Note: If it doesn’t show up it might be due to cache. Run \texttt{/install/upgrade.php} to clear it.

30.1.10 Connecting to the API with POSTMan

Using the API with Postman is not strictly necessary but it’s nice to get a feel on how the backend API works.

Install Postman or another tool to make API requests. Download it from here:
https://www.getpostman.com

Authenticate

Send a POST request to:

\texttt{http://localhost/api/auth.php}

use content type:

\texttt{application/json}
And the following request body:

```json
{
    "username": "admin",
    "password": "adminadmin"
}
```

When successfully logged on you should get a response with status:

```
201 Authentication is complete, access token created
```

Find the “accessToken” property and save it. From now on you can do API requests to:

```
http://localhost/api/jmap.php
```

You must set the access token as a header on each request:

```
Authorization: Bearer 5b7576e5c50ac30f0e53373f0fa614cedbde49df7637
Content-Type: application/json
```

**Create an artist**

To create an artist, POST this JSON body:

```json
[
    [
        "Artist/set",
        {
            "create": {
                "clientId-1": {
                    "name": "The Doors",
                    "albums": [
                        {
                            "name": "The Doors",
                            "artistId": 1,
                            "releaseDate": "1967-01-04",
                            "genreId": 2
                        },
                        {
                            "name": "Strange Days",
                            "artistId": 1,
                            "releaseDate": "1967-09-25",
                            "genreId": 2
                        }
                    ]
                }
            }
        },
        "call1"
    ],
    [
        "community/dev/Debugger/get",
        {},
        "call4"
    ]
]
```

**Query artists**

The Artist/query method is used to retrieve an ordered / filtered list of id's for displaying a list of artists. We'll do a direct followup call to “Artist/get” to retrieve the full artist data as well. We can use the special “#” parameter to use a previous query result as parameter. Read more on this at the JMAP website.

POST the following to make the request:

```json
[
    [
        "Artist/query",
        {},
        "call1"
    ],
    [
        "Artist/get",
        {
            "identity": "%id%
        }
    ]
]
```
30.1. Building a server module
POST http://localhost:8080/jmap.php

```json
[{
  "Artist/set": {
    "create": {
      "clientId": {
        "name": "The Doors",
        "albums": [
          {
            "name": "The Doors", "artistId": 1, "releaseDate": "1967-01-01", "genreId": 2,
            "name": "Strange Days", "artistId": 1, "releaseDate": "1967-09-25", "genreId": 2
          }
        ]
      }
    }
  }
},
  "call1"],

["community/dev/Debugger/get", {}, "call4"]
]}
```
30.1.11 Query filters

When doing “Artist/query” requests, it’s possible to filter the results. You can pass for example:

```json
{"filter": {"text": "Foo"}}
```

We generally use the “text” filter for a quick search query. We also want to filter artists by their album genres. We can implement this in our “Artist” entity in by overriding the “filter” method:

```php
/**
 * This function returns the columns to search when using the "text" filter.
 */
public static function textFilterColumns() {
    return ['name'];
}

/**
 * Defines JMAP filters
 *
 * Adds the 'genres' filter which can be an array of genre id's.
 * @link https://jmap.io/spec-core.html#/query
 * @return Filters
 */
protected static function defineFilters() {
    return parent::defineFilters()->add('genres', function ($criteria, $value, $query, $filter) {
        if (!empty($value)) {
            $query->join('music_album', 'album', 'album.artistId = artist.id')
            ->groupBy(['artist.id']) // group the results by id to filter out duplicates because of the join
            ->where(['album.genreId' => $value]);
        }
    });
}
```

After defining this you can filter on genre by posting:

```json
[
    "Artist/query", {
        "filter": {
            "genres": [1, 2, 3]
        },
        "call1"
    }
]
```
Chapter 30. Developer
JMAP API protocol

These are some basic request examples. Read more on https://jmap.io about the protocol.

30.1.12 Module installation

When you’re done with the module, you should export your finished database into:

```go
modules/tutorial/music/install/install.sql
```

Put all ‘DROP TABLE x’ commands in:

```go
modules/tutorial/music/install/uninstall.sql
```

30.1.13 Module upgrades

When the database changes later on you can put upgrade queries and php functions in:

```go
modules/tutorial/music/install/updates.php
```

For example:

```php
$updates['201808161606'][] = "ALTER TABLE ...";
$updates['201808161606'][] = function() {
    //some migration code here
};
```

The timestamp is important. Use YYYYMMDDHHII. All the module upgrades will be mixed together and put into chronological order so dependant modules won’t break.

30.1.14 Custom Fields

Custom development is a fact of life, even for your simple tutorial module. In this example, we will add a few custom fields to different entities.

First, we need to update the database. For an entity to be customized, we need to add a table that follows the convention below:

```{go}
<Module>_<Entity>_custom_fields
```

We will have to write a database migration. Open `go/modules/tutorial/music/updates.php` and add the following code:

```php
$updates['202002041445'][] = <<<'EOT'
CREATE TABLE IF NOT EXISTS `music_artist_custom_fields`  
( id INT(11) NOT NULL PRIMARY KEY,  
CONSTRAINT `music_artist_custom_fields_ibfk_1` FOREIGN KEY(id) REFERENCES music_artist (id)  
ON DELETE CASCADE ON UPDATE RESTRICT ) ENGINE = INNODB;
EOT;
```

The next step is to add the CustomFieldsTrait trait to the Artist model.
<?php
namespace go\modules\tutorial\music\model;

use go\core\jmap\Entity;
use go\core\orm\CustomFieldsTrait;

/**
 * Artist model
 * @copyright (c) 2020, Intermesh BV http://www.intermesh.nl
 * @author Merijn Schering <mschering@intermesh.nl>
 * @license http://www.gnu.org/licenses/agpl-3.0.html AGPLv3
 */
class Artist extends Entity {
    use CustomFieldsTrait;

    // Et cetera...
}

After rerunning the install script, the Custom fields screen in System settings should look somewhat like this:

For now, we are done. In the next chapter, we will see how custom fields are made available in the web client.

### 30.1.15 ACL Entities

In certain cases, you want to either have some private entities or entities that are shared with certain groups of other users. This section deals with developing entities that have some form of access control.

In our tutorial module, we will add a ACL entity named ‘review’, which will allow you to add reviews to albums and control with whom you want to share your guilty pleasures.

The first step is to create the `music_review` table. Open `install/updates.php` and add the code below:

```sql
$updates['202002071045'][] = '<?EOT
CREATE TABLE IF NOT EXISTS `music_review`
(`id` INT(11) NOT NULL PRIMARY KEY,
`albumId` INT(11) NOT NULL,
`aclId` INT(11) NOT NULL,
`createdBy` int(11) NOT NULL,
`modifiedBy` int(11) NOT NULL,
`rating` SMALLINT(5) UNSIGNED NOT NULL,
`title` VARCHAR(190) COLLATE utf8mb4_unicode_ci NOT NULL,
(continues on next page)
```
The next step is to create a model, which we extend from the AclOwnerEntity class:

```php
<?php
namespace go\modules\tutorial\music\model;
use Exception;
use go\core\acl\model\AclOwnerEntity;
use go\core\orm\Query;

class Review extends AclOwnerEntity
{
    /** @var int */
    public $id;
    /** @var int */
    public $aclId;
    /** @var int */
    public $createdBy;
    /** @var int */
    public $albumId;
    /** @var int */
    public $modifiedBy;
    /** @var int */
    public $rating;
    /** @var string */
    public $title;
    /** @var string */
    public $body;
    /** @var string */
    public $albumTitle;

    protected static function defineMapping()
    {
        return parent::defineMapping()
            ->addTable('music_review')
            ->setQuery((new Query())->select('a.name AS albumTitle')
                ->join('music_album', 'a', 'a.id=music_review.albumId'));
    }
}
```

Run the database install script again and you should see the newly created table in the database.

30.1. Building a server module
protected function internalSave()
{
    if($this->isNew()) {
        $this->albumtitle = go()->getDbConnection()
            ->selectSingleValue('name')
            ->from('music_album')
            ->where(['id' => $this->albumId])
            ->single();
    }
    $this->changeArtist([$this->albumId]);
    return parent::internalSave();
}

protected static function internalDelete(Query $query)
{
    //Create clone to avoid changes to the original delete query object
    $deleteQuery = clone $query;
    //Select albums of artists affected by this delete
    $deleteQuery->selectSingleValue('albumId');
    static::changeArtist($deleteQuery->all());
    return parent::internalDelete($query);
}

/**
 * Our review has effect on Artist entities because they implement getAlbumCount().
 * @param array $albumIds
 * @throws Exception
 */
private static function changeArtist(array $albumIds) {
    Artist::entityType()->changes(
        go()->getDbConnection()
            ->select(['art.id', null, 0])
            ->from('music_artist', 'art')
            ->join('music_album', 'alb', 'alb.artistId = art.id')
            ->where('alb.id', 'IN', $albumIds)
    );
}

protected static function defineFilters()
{
    return parent::defineFilters();
    ->add('albumId', function (\go\core\db\Criteria $criteria, $value,
        \go\core\orm\Query $query, array $filter) {
        if (!empty($value)) {
            $query->where(['music_review.albumId' => $value]);
        }
    });
}
Interestingly, we need to make sure that the Artist entity is updated upon adding or deleting a review. We force this by overriding the `internalSave` and `internalDelete` methods.

### 30.1.16 The end

Now you’re done with the server code of the module. It is time to move on and build the web client!

### 30.2 Building a webclient module

When you’ve finished the server module it’s time to build a web client. Please note that the reviews are being added in the second part of this tutorial and that they fall outside the scope of this document. We’re going to create a 3 column responsive layout with a Genre filter, artist list and artist detail view.

Our webclient framework is based on ExtJS 3.4 so you can find examples and API documentation here:

https://docs.sencha.com/extjs/3.4.0/

We’ve enhanced ExtJS with our own components and created a theme for Group Office.

The webclient code is located in the `go/modules/tutorial/music/views/extjs3` folder. The code generator already created these files:
1. Module.js: Required for each module. It registers the module, entities, system and user setting panels.
2. MainPanel.js: The main panel of the module shown in the Group Office UI
3. scripts.txt: All js files must be listed in the correct order here.
4. themes/default/style.css. Module specific style can be placed here. You can use our _base.scss file to use functions and variables from the main style.

When opening Group-Office you should see “Music” in the start menu. When opening it shows “Hello world”.

### 30.2.1 Entities

First add all entities to the module in Module.js:

```javascript
go.Modules.register("tutorial", "music", {  
    mainPanel: "go.modules.tutorial.music.MainPanel",
    //The title is shown in the menu and tab bar
    title: t("Music"),
    //All module entities must be defined here. Stores will be created for them.
    entities: ["Genre", "Artist"],
    //Put code to initialize the module here after the user is authenticated
    //and has access to the module.
    initModule: function () {}  
});
```

This will create a go.data.EntityStore for each entity. This store will sync all entity data. This store is kept up to date automatically. When for example a form dialog makes a Foo/set request, the store will receive the dispatched action and fire an “updated” event. All view stores connected to grids and detail views for example can observe this store and render the view on this event.

Read more about entities [here](#).

### 30.2.2 Genre filter

Create a new file Genrefilter.js:

```javascript
go.modules.tutorial.music.GenreFilter = Ext.extend(go.grid.GridPanel, {  
    viewConfig: {  
        forceFit: true, 
        autoFill: true 
    },
    //This component is going to be the side navigation
    cls: 'go-sidenav',
    initComponent: function () {  
        // Row actions is a special grid column with an actions menu in it.
        var actions = this.initRowActions();
        // A selection model with checkboxes in this filter.
        var selModel = new Ext.grid.CheckboxSelectionModel();
    }
});
```

(continues on next page)
// A toolbar that consists out of two rows.
var tbar = {
  xtype: "container",
  items: [
    {
      items: this.tbar || [],
      xtype: 'toolbar'
    },
    new Ext.Toolbar({
      items:[{xtype: "selectallcheckbox"}]
    })
  ],
};

Ext.apply(this, {
  tbar: tbar,
  // We use a "go.data.Store" that connects with an Entity
  store: new go.data.Store({
    fields: ['id', 'name', 'aclId', "permissionLevel"],
    entityStore: "Genre"
  }),
  selModel: selModel,
  plugins: [actions],
  columns: [
    // The checkbox selection model must be added as a
    selModel,
    {
      id: 'name',
      header: t('Name'),
      sortable: false,
      dataIndex: 'name',
      hideable: false,
      draggable: false,
      menuDisabled: true
    },
    // The actions column showing a menu with delete and
    actions
  ],
  // Change to true to remember the state of the panel
  stateful: false,
  stateId: 'music-genre-filter'
});


call(this);

initRowActions: function () {

  var actions = new Ext.ux.grid.RowActions({
    menuDisabled: true,
  }

(continues on next page)
hideable: false,
 draggable: false,
 fixed: true,
 header: '',
 hideMode: 'display',
 keepSelection: true,

actions: [{
   iconCls: 'ic-more-vert'
 }]
});

actions.on({
   action: function (grid, record, action, row, col, e, target) {
      this.showMoreMenu(record, e);
   },
   scope: this
});

return actions;
,

showMoreMenu : function(record, e) {
   if(!this.moreMenu) {
      this.moreMenu = new Ext.menu.Menu({
         items: [
            {
               itemId: "edit",
               iconCls: 'ic-edit',
               text: t("Edit"),
               handler: function() {
                  var dlg = new go.modules.
                     tutorial.music.GenreForm();
                  dlg.load(this.moreMenu.record.
                     id).show();
               },
               scope: this
            },
            {
               itemId: "delete",
               iconCls: 'ic-delete',
               text: t("Delete"),
               handler: function() {
                  Ext.MessageBox.confirm(t{
                     "Confirm delete"), t("Are you sure you want to delete this item?")}, function (btn) {
                     if (btn != "yes") {
                        return;
                     }
                     go.Stores.get("Genre 
                    ").set({destroy: [this.moreMenu.record.id]});
               }, this);
            }
         ]
      });
   }
   return this.moreMenu;
};
Every Javascript file must be added to the `scripts.txt` file so add `GenreFilter.js` to the bottom of this file. Study the component and take a look at all the comments. This component is a grid with check boxes showing all Genres.

Now add this component to the main panel by changing `MainPanel.js` with the following code:

```javascript
// Will make a single item fit in this panel. We'll change this later.
layout : "fit",

initComponent : function() {

    //create the genre filter component
    this.genreFilter = new go.modules.tutorial.music.GenreFilter({
        tbar : [{
            xtype : "tbtitle",
            text : t("Genres")
        }]
    });

    //add it to the main panel's items.
    this.items = [this.genreFilter];


    this.on("afterrender", function() {
        //when this panel renders, load the filter.
        this.genreFilter.store.load();
    },this);

};
```

Reload Group Office and the Music panel should now look like this:
30.2.3 Relations

To present data from related entities. For example. The user who created an Artist you can use relations. For each entity you can define relations to other entities. Change the string “Artist” in the entities property in Module.js to the following:

```json
{
    name: "Artist",
    relations: {
        creator: {store: "User", fk: "createdBy"},
        modifier: {store: "User", fk: "createdBy"},
        // 'albums' is a property of artist and has a nested relation.
        albums: {
            genre: {store: "Genre", fk: "genreId"}
        }
    }
}
```

We’ve defined two “has one” relations for the creator and modifier and a “has many” relation for the albums.

The complete Module.js looks like this now:

```javascript
go.Modules.register("tutorial", "music", {
    mainPanel: "go.modules.tutorial.music.MainPanel",
    //The title is shown in the menu and tab bar
    title: t("Music"),
});
```

(continues on next page)
//All module entities must be defined here. Stores will be created for them.
entities: [
  "Genre",
  {  
    name: "Artist",
    relations: {  
      creator: {store: "User", fk: "createdBy"},
      modifier: {store: "User", fk: "createdBy"},
      // 'albums' is a property of artist and has a nested relation.
      albums: {  
        genre: {store: "Genre", fk: "genreId"}
      }
    }
  }
],

//Put code to initialize the module here after the user is authenticated
//and has access to the module.
initModule: function () {}

We can use these relations in the artist grid in the next chapter.

### 30.2.4 Artist grid

Now that we’ve got our Genre filter in place it’s time to create the artist grid.

Create the file `ArtistGrid.js`:

```javascript
go.modules.tutorial.music.ArtistGrid = Ext.extend(go.grid.GridPanel, {
  initComponent: function () {
    this.store = new go.data.Store({
      fields: [
        'id',
        'name',
        'photo', // This is a blob id. A download URL can be retrieved with go.Jmap.downloadUrl(record.data.photo)
        {name: 'createdAt', type: 'date'},
        {name: 'modifiedAt', type: 'date'},
      ]
    })
    // You can use "relation" as a store data type. This will automatically
    // fetch the related entity by the definition in Module.js.
    {name: 'creator', type: "relation"},
    {name: 'modifier', type: "relation"},
    // Every entity has permission levels. go.
    permissionLevels.read, write,
    // writeAndDelete and manage 'permissionLevel'
  }
});
```

(continues on next page)
// The connected entity store. When Artists are changed the store will update automatically
entityStore: "Artist"
});
Ext.apply(this, {
  columns: [
    {
      id: 'id',
      hidden: true,
      header: 'ID',
      width: dp(40),
      sortable: true,
      dataIndex: 'id'
    },
    {
      id: 'name',
      header: t('Name'),
      width: dp(75),
      sortable: true,
      dataIndex: 'name',
      renderer: function (value, metaData, record, rowIndex, colIndex, store) {
        return '<div class="user">
          <div class="avatar" style="background-image: url(' + go.Jmap.downloadUrl(record.data.photo) + ')" />
          <div class="wrap single">' + record.get('name') + '</div>
        </div>'
      }
    },
    {
      id: 'albumcount',
      sortable: false,
      header: t('album_count','music','tutorial'),
      dataIndex: 'albumcount',
      width: dp(80)
    },
    {
      xtype: "datecolumn",
      id: 'createdAt',
      header: t('Created at'),
      width: dp(160),
      sortable: true,
      dataIndex: 'createdAt',
      hidden: true
    },
    {
      xtype: 'colors',
      id: 'color',
      header: t('Color'),
      width: dp(80),
      sortable: true,
      dataIndex: 'color',
      hidden: true
    },
  ]
});
//Render an avatar for the artist.
var style = record.data.photo ? 'background-image: url(' + go.Jmap.downloadUrl(record.data.photo) + ')"' : '"
return '<div class="user">
  <div class="avatar" style="background-image: url(' + go.Jmap.downloadUrl(record.data.photo) + ')" />
  <div class="wrap single">' + record.get('name') + '</div>
</div>'
</div>
</div>
</div>
</div>
And add the file `ArtistGrid.js` to the bottom of `scripts.txt`. Study the code and comments of this file.

Now change `MainPanel.js` to use the grid:

```javascript
go.modules.tutorial.music.MainPanel = Ext.extend(go.modules.ModulePanel, {
    // Use a responsive layout
    layout : "responsive",
    initComponent : function() {
        // (continues on next page)
    }
});
```

30.2. Building a webclient module
//create the genre filter component
this.genreFilter = new go.modules.tutorial.music.GenreFilter({
    region: "west",
    width: dp(300),

    //render a split bar for resizing
    split: true,
    tbar: [{
        xtype: "tbtitle",
        text: t("Genres")
    }
]},

//Create the artist grid
this.artistGrid = new go.modules.tutorial.music.ArtistGrid({
    region: "center",

    //toolbar with just a search component for now
    tbar: [`,
        {
            xtype: 'tbsearch'
        }
    ]
},

//add the components to the main panel's items.
this.items = [this.genreFilter, this.artistGrid];

// Call the parent class' initComponent

---call(this);

//Attach listener to changes of the filter selection.
//add buffer because it clears selection first and this would cause
//it to fire twice
this.genreFilter.getSelectionModel().on('selectionchange', this.
    onGenreFilterChange, this, {buffer: 1});

// Attach listener for running the module
this.on("afterrender", this.runModule, this);

// Fired when the Genre filter selection changes
onGenreFilterChange : function (sm) {

    var selectedRecords = sm.getSelections(),
        ids = selectedRecords.column('id'); //column
    //is a special GO method that get's all the id's from the records in an array.

    this.artistGrid.store.setFilter('genres', {genres: ids});
    this.artistGrid.store.load();
}

// Fired when the module panel is rendered.
runModule : function() {
  // when this panel renders, load the genres and artists.
  this.genreFilter.store.load();
  this.artistGrid.store.load();
}

Study this component code and comments again. The changes that are made are:

1. The layout to a responsive layout so the components can be next to each other. A responsive layout is based on Ext.layout.BorderLayout but changes into a Ext.layout.CardLayout when the device width is smaller than a specified trigger point.

2. Added the Artist grid component.

3. Added a listener to the Genre filter to apply the filter to the artist grid’s store parameters.

When you reload Group Office now it should look like this:

![Group Office interface](image)

**Note:** Feel free to add some more artists with Postman so your filter results are more interesting :) You might also notice that when you change things with postman the web interface updates automatically.

### 30.2.5 Genre combo box

Before we can create an Artist dialog we’ll need a Genre combo box for selecting the album genre. Create the file GenreCombo.js:
go.modules.tutorial.music.GenreCombo = Ext.extend(go.form.ComboBox, {
    fieldLabel: t("Genre"),
    hiddenName: 'genreId',
    anchor: '100%',
    emptyText: t("Please select..."),
    pageSize: 50,
    valueField: 'id',
    displayField: 'name',
    triggerAction: 'all',
    editable: true,
    selectOnFocus: true,
    forceSelection: true,
    allowBlank: false,
    store: {
        xtype: "gostore",
        fields: ['id', 'name'],
        entityStore: "Genre"
    }
});

// Register an xtype so we can use the component easily.
Ext.reg("genrecombo", go.modules.tutorial.music.GenreCombo);

Study the component and add it to the scripts.txt file.

### 30.2.6 Artist dialog

Now we need an Artist dialog for creating and editing Artists.

Create a file called `ArtistDialog.js`:

```javascript
go.modules.tutorial.music.ArtistDialog = Ext.extend(go.form.Dialog, {
    // Change to true to remember state
    stateful: false,
    stateId: 'music-artist-dialog',
    title: t('Artist'),

    //The dialog set's entities in an go.data.EntityStore. This store notifies all
    //connected go.data.Store view stores to update.
    entityStore: "Artist",
    autoHeight: true,

    // return an array of form items here.
    initFormItems: function () {
        return []
            // it's recommended to wrap all fields in field sets
            // for consistent style.
            xtype: 'fieldset',
            title: t("Artist information"),
            items: [
                this.avatarComp = new go.form.ImageField({
                    name: 'photo'
                }),
                {
                    xtype: 'textfield',
```
name: 'name',
fieldLabel: t("Name"),
anchor: '100%',
allowBlank: false
]
},
{
xtype: "fieldset",
title: t("Albums"),
items: [
  }
  ]
  xtype: "formgroup",
  name: "albums",
  hideLabel: true,
  mapKey: 'id',
  // this will add dp(16) padding
  pad: true,
  // the itemCfg is used to create a
  itemCfg: {
    layout: "form",
    defaults: {
      anchor: "100%"
    },
    items: [{
      xtype: "textfield",
      fieldLabel: t("Name"),
      name: "name"
    },
    {
      xtype: "datefield",
      fieldLabel: t("Release date"),
      name: "releaseDate"
    },
    {
      xtype: "genrecombo"
    }]
  }
}
}
Add this file to the `scripts.txt` file again.

Then update `MainPanel.js`:

```javascript
go.modules.tutorial.music.MainPanel = Ext.extend(go.modules.ModulePanel, {
    // Use a responsive layout
    layout: "responsive",

    initComponent: function() {
        // create the genre filter component
        this.genreFilter = new go.modules.tutorial.music.GenreFilter({
            region: "west",
            width: dp(300),
            split: true,
            tbar: [{
                xtype: "tbtitle",
                text: t("Genres")
            }]
        });

        // Create the artist grid
        this.artistGrid = new go.modules.tutorial.music.ArtistGrid({
            region: "center",
            tbar: [{
                xtype: 'tbsearch'
            }]
        });

        // toolbar with just a search component for now
        tbar: [{
            xtype: "tbsearch",
            text: t("Genres")
        }]
    };

    // add button for creating new artists
    this.addButton = new Ext.Button({
        iconCls: 'ic-add',
        tooltip: t('Add'),
        handler: function (btn) {
            var dlg = new go.modules.tutorial.music.ArtistDialog({
                formValues: {
                    // you can pass form values like this
                };
            dlg.show();
        }
    };
```
scope: this
},

listeners: {
  rowdblclick: this.onGridDblClick,
  keypress: this.onGridKeyPress,
  scope: this
};

//add the components to the main panel's items.
this.items = [this.genreFilter, this.artistGrid];

// Call the parent class' initComponent
→call(this);

//Attach listener to changes of the filter selection.
→add buffer because it clears selection first and this would cause
→it to fire twice
this.genreFilter.getSelectionModel().on('selectionchange', this.
→onGenreFilterChange, this, {buffer: 1});

  // Attach listener for running the module
  this.on("afterrender", this.runModule, this);
},

// Fired when the Genre filter selection changes
onGenreFilterChange : function (sm) {
  var selectedRecords = sm.getSelections(),
      ids = selectedRecords.column('id'); //column
  →is a special GO method that get's all the id's from the records in an array.
  this.artistGrid.store.setFilter('genres', {genres: ids});
  this.artistGrid.store.load();
},

// Fired when the module panel is rendered.
runModule : function() {
  // when this panel renders, load the genres and artists.
  this.genreFilter.store.load();
  this.artistGrid.store.load();
},

// Fires when an artist is double clicked in the grid.
onGridDblClick : function (grid, rowIndex, e) {
  // check permissions
  var record = grid.getStore().getAt(rowIndex);
  if (record.get('permissionLevel') < GO.permissionLevels.write) {
    return;
  }
  // Show dialog
(continues on next page)
```
var dlg = new go.modules.tutorial.music.ArtistDialog();
    dlg.load(record.id).show();
    }

    // Fires when enter is pressed and a grid row is focussed
    onGridKeyPress : function(e) {
        if(e.keyCode != e.ENTER) {
            return;
        }
        var record = this.artistGrid.getSelectionModel().getSelected();
        if(!record) {
            return;
        }
        if (record.get('permissionLevel') < GO.permissionLevels.write) {
            return;
        }
        var dlg = new go.modules.tutorial.music.ArtistDialog();
        dlg.load(record.id).show();
    }
});
```

Study the changes in the component:

1. Added an Add button in the grid’s toolbar.
2. Added a double click listener to the grid to edit an Artist.

When you reload Group Office now it should look like this:

### 30.2.7 Delete button

You can add a delete button to the grid’s toolbar in `MainPanel.js` to delete selected artists. Note that this code will add a “More options” menu button with 3 dots that has the delete button in the menu:

```
{
    iconCls: 'ic-more-vert',
    tooltip: t("More options"),
    menu: [
        {
            itemId: "delete",
            iconCls: 'ic-delete',
            text: t("Delete"),
            handler: function () {
                this.artistGrid.deleteSelected();
            },
            scope: this
        }
    ]
}
```

**Note:** You can also delete items by pressing the “delete” key.
30.2.8 Detail view

Finally we’re going to add a detail panel for artists.

Create the file `ArtistDetail.js`:

```javascript
go.modules.tutorial.music.ArtistDetail = Ext.extend(go.detail.Panel, {

    // The entity store is connected. The detail view is automatically updated.
    entityStore: "Artist",

    // set to true to enable state saving
    stateful: false,
    stateId: 'music-contact-detail',

    // Fetch these relations for this view
    relations: ["albums.genre"],

    initComponent: function () {
        this.tbar = this.initToolbar();

        Ext.apply(this, {
            // all items are updated automatically if they have a "tpl" property or an "onLoad" function. The panel is passed as argument.
            items: [
                // Artist name component
                { cls: 'content', xtype: 'box',


```
tpl: '<h3>{name}</h3>'
}

//Render the avatar
{
    xtype: "box",
    cls: "content",
    tpl: new Ext.XTemplate('<div class="go-detail-view-avatar">
        <div class="avatar" style="width:120px;height:120px;{[this.getStyle(values.photo)]}"></div>
    </div>',
    getCls: function(isOrganization) {
        return isOrganization ? "organization" : "";
    },
    getStyle: function(photoBlobId) {
        return photoBlobId ? 'background-image: url(' + go.Jmap.downloadUrl(photoBlobId) + ')" : "";
    }
})

// Albums component
{
    collapsible: true,
    title: t("Albums"),
    xtype: "panel",
    tpl: '<div class="icons">
        <tpl for="albums">
            <p class="s6"><tpl if="xindex==1"><i class="icon label">album</i></tpl><span>{name}</span><label>{[go.util.Format.date(values.releaseDate)]} - <tpl for="genre">{name}</tpl></label>
        </p>
        </tpl>
    </div>
    <tpl for="albums">
        <p class="s6"><tpl if="xindex==1"><i class="icon label">album</i></tpl><span>{name}</span><label>{[go.util.Format.date(values.releaseDate)]} - <tpl for="genre">{name}</tpl></label>
        </p>
        </tpl>
    }
}

// Enable edit button according to permission level.
_onLoad: function () {
    this.getTopToolbar().getComponent("edit").setDisabled(this.data.permissionLevel < go.permissionLevels.write);
    this.deleteItem.setDisabled(this.data.permissionLevel < go.permissionLevels.writeAndDelete);
}
go.modules.tutorial.music.ArtistDetail.superclass.onLoad.call(this);

initToolbar: function () {
    var items = this.tbar || [];

    items = items.concat([...
      {itemId: "edit",
        iconCls: 'ic-edit',
        tooltip: t("Edit"),
        handler: function (btn, e) {
          var dlg = new go.modules.tutorial.music.ArtistDialog();
          dlg.show();
          dlg.load(this.data.id);
        },
        scope: this
      },
      {iconCls: 'ic-more-vert',
        menu: [
          {
            iconCls: "ic-print",
            text: t("Print"),
            handler: function () {
              this.body.print({ title: this.data.name });
            },
            scope: this
          },
        ],
        scope: this
      },
      this.deleteItem = new Ext.menu.Item({
        itemId: "delete",
        iconCls: 'ic-delete',
        text: t("Delete"),
        handler: function () {
          Ext.MessageBox.confirm(t("Confirm delete"), t("Are you sure you want to delete this item?")), function (btn) {
            if (btn != "yes") {
              return;
            }
            this.entityStore.set({ destroy: [this.currentId] });
          }, this);
        },
        scope: this
      }
    ]);

    var tbarCfg = {
      ...}

    (continues on next page)
disabled: true,
items: items
};

return new Ext.Toolbar(tbarCfg);
}
}

Study the code and add it to scripts.txt. Now we’re going to update the MainPanel.js file:

go.modules.tutorial.music.MainPanel = Ext.extend(go.modules.ModulePanel, {

    // Use a responsive layout
    layout: "responsive",

    // change responsive mode on 1000 pixels
    layoutConfig: {
        triggerWidth: 1000
    },

    initComponent: function () {

        //create the genre filter component
        this.genreFilter = new go.modules.tutorial.music.GenreFilter({
            region: "west",
            width: dp(300),

            //render a split bar for resizing
            split: true,

            //add a hamburger button for smaller screens
            tbar: [{
                xtype: "tbtitle",
                text: t("Genres")
            },
            '->',

            //add back button for smaller screens
            { //this class will hide it on larger screens
                cls: 'go-narrow',
                iconCls: "ic-arrow-forward",
                tooltip: t("Artists"),
                handler: function () {
                    this.artistGrid.show();
                },
                scope: this
            }
        ]
    }

    //Create the artist grid
    this.artistGrid = new go.modules.tutorial.music.ArtistGrid({
        region: "center",

        tbar: [
            //add a hamburger button for smaller screens
            {
                //this class will hide it on larger screens
                cls: 'go-narrow',
                iconCls: "ic-arrow-forward",
                toolti
// this class will hide the button on large screens

cls: 'go-narrow',
iconCls: "ic-menu",
handler: function () {
    this.genreFilter.show();
},
scope: this

// add button for creating new artists
this.addButton = new Ext.Button({
    iconCls: 'ic-add',
tooltip: t('Add'),
handler: function (btn) {
    var dlg = new go.modules.tutorial.
    music.ArtistDialog({
        formValues: {
            // you can pass form values like this
        }
        }
    dlg.show();
},
scope: this
}),
{
    iconCls: 'ic-more-vert',
    menu: [
        {
            itemId: "delete",
            iconCls: 'ic-delete',
text: t("Delete"),
handler: function () {
    this.artistGrid.
    deleteSelected();
},
scope: this
        }
    ]
},
layers: {
    rowdblclick: this.onGridDoubleClick,
    scope: this
}
});

// Every entity automatically configures a route. Route to the entity when selecting it in the grid.
this.artistGrid.on('navigate', function (grid, rowIndex, record) {
    go.Router.goto('artist/' + record.id);
});

(continues on next page)
[GROUP-OFFICE manual](#), this);

// Create artist detail component
this.artistDetail = new go.modules.tutorial.music.ArtistDetail({
    region: "center",
    tbar: [
        // add a back button for small screens
        {
            // this class will hide the button on large
            // screens
            cls: 'go-narrow',
            iconCls: "ic-arrow-back",
            handler: function () {
                this.westPanel.show();
            },
            scope: this
        }
    ];
});

// Wrap the grids into another panel with responsive layout for the 3
// column responsive layout to work.
this.westPanel = new Ext.Panel({
    region: "west",
    layout: "responsive",
    stateId: "go-music-west",
    split: true,
    width: dp(800),
    narrowWidth: dp(500), // this will only work for panels inside
    // another panel with layout=responsive. Not ideal but at the moment the only way I
    // could make it work
    items: [
        this.artistGrid, // first item is shown as default in
        // narrow mode.
        this.genreFilter
    ];
});

// Add the components to the main panel's items.
this.items = [
    this.westPanel, // first is default in narrow mode
    this.artistDetail
];

// Call the parent class' initComponent
// call(this);

// Attach listener to changes of the filter selection.
// add buffer because it clears selection first and this would cause
// it to fire twice
this.genreFilter.getSelectionModel().on('selectionchange', this.
// onGenreFilterChange, this, {buffer: 1});

// Attach listener for running the module
this.on("afterrender", this.runModule, this);

(continues on next page)
// Fired when the Genre filter selection changes
onGenreFilterChange: function (sm) {
    var selectedRecords = sm.getSelections(),
        ids = selectedRecords.column('id'); // column is a special GO method that get's all the id's from the records in an array.
    this.artistGrid.store.setFilter('genres', {genres: ids});
    this.artistGrid.store.load();
},

// Fired when the module panel is rendered.
runModule: function () {
    // when this panel renders, load the genres and artists.
    this.genreFilter.store.load();
    this.artistGrid.store.load();
},

// Fires when an artist is double clicked in the grid.
onGridDbClick: function (grid, rowIndex, e) {
    //check permissions
    var record = grid.getStore().getAt(rowIndex);
    if (record.get('permissionLevel') < GO.permissionLevels.write) {
        return;
    }
    // Show dialog
    var dlg = new go.modules.tutorial.music.ArtistDialog();
    dlg.load(record.id).show();
});

The changes:
1. Added a layout trigger width
2. Added the detail view component and wrapped the grids in a new panel. So that we have two responsive panels one reacting for tables and the other one switching for phones.
3. Added buttons for navigating on smaller screens. See the new buttons with the “go-narrow” class on them.
4. We’ve added a row select listener to navigate to the artist page using the router.

When you reload Group Office it should look like this:

And on tablets:
And on phones:

**30.2.9 Override CSS**

It is pretty easy to add your own CSS classes. When initially generating your module boilerplate, an empty style.css file is automatically generated. You can either edit it directly or use the SASS docker image from the development environment to generate your own CSS overrides.
30.2. Building a webclient module
SASS

In your module directory, create a new SASS file:

```
views/extjs3/themes/default/src/style.scss
```

Please note that the file name should be `style.scss` to make sure that it is automatically compiled.

**Note:** You can override a certain theme file, by substituting `default` by the theme name if necessary.

By default, an avatar is displayed at 35 by 35 pixels and they are aligned to the left. Let’s make the avatar image somewhat larger, e.g. 120 px and let’s center it. We will have to define a CSS class that centers its content and make sure that the nested `avatar` CSS class renders at 120px by 120px. Update your `style.scss` file to look like this:

```
.go-detail-view-avatar {  
text-align:center;
  
  > .avatar {  
    width: 120px;
    height: 120px;
  }  
}
```

Assuming that you use the Intermesh development docker environment, you have a running SASS container that monitors any scss files for changes. You can check the output of sass by opening up a console and entering the following command:

```
docker-compose logs --follow sass
```

In case that your module SCSS code is not automatically compiled, you can restart the sass docker container:

```
docker-compose restart sass
```

..and run the database install script again. This will make sure that your custom CSS is included.

### 30.2.10 Custom fields

Remember that you made the Artist model customizable? Now, we are going to add them to the webclient module! Let’s add a custom checkbox as per this guide, as well as a biography field. For the sake of simplicity, this field will be a text field.

The custom fields will work out of the box in the editing modal, provided that you have the proper rights:

The next step is making the custom information available in the artist details. Add the following line at the end of the `initComponent()` function in `ArtistDetail.js`:
30.2. Building a webclient module
this.addCustomFields();

Reload your screen and you will see your custom fields in the artist detail pane:

The custom fields will work out of the box in the editing modal, provided that you have the proper rights:

![Artist with custom fields](image)

**Add a custom Filter**

We're not done yet. After all, we would like to be able to search by our custom fields. Furthermore, we want to group our filters in the side panel. In order to be able to search by our custom field (e.g. the 'Active' checkbox above), we have to add a custom filter panel and define a filter based on the custom field.

Once again, we open the `MainPanel.js` file. Enter the following code:
go.modules.tutorial.music.MainPanel = Ext.extend(go.modules.ModulePanel, {
    // Use a responsive layout
    layout: "responsive",

    // change responsive mode on 1000 pixels
    layoutConfig: {
        triggerWidth: 1000
    },

    initComponent: function () {
        this.createArtistGrid();

        // Every entity automatically configures a route. Route to the entity when selecting it in the grid.
        this.artistGrid.on('navigate', function (grid, rowIndex, record) {
            go.Router.goto("artist/" + record.id);
        }, this);

        this.sidePanel = new Ext.Panel({
            layout: 'anchor',
            defaults: {
                anchor: '100%
            },
            width: dp(300),
            cls: 'go-sidenav',
            region: "west",
            split: true,
            autoScroll: true,
            items: [
                this.createGenreFilter(),
                this.createFilterPanel()
            ]
        });

        // Create artist detail component
        this.artistDetail = new go.modules.tutorial.music.ArtistDetail({
            region: "center",
            tbar: [
                //add a back button for small screens
                {
                    // this class will hide the button on large screens
                    cls: 'go-narrow',
                    iconCls: "ic-arrow-back",
                    handler: function () {
                        this.westPanel.show();
                    },
                    scope: this
                }
            ]
        });

        //Wrap the grids into another panel with responsive layout for the 3 column responsive layout to work.
        this.westPanel = new Ext.Panel({
            region: "west",
            layout: "responsive",
            stateId: "go-music-west",
            split: true,
        })
    });
});

30.2. Building a webclient module
width: dp(800),
narrowWidth: dp(500), //this will only work for panels inside another panel with layout=responsive. Not ideal but at the moment the only way I could make it work
items: [
    this.artistGrid,
    this.sidePanel
];

//add the components to the main panel's items.
this.items = [
    this.westPanel, //first is default in narrow mode
    this.artistDetail
];

// Call the parent class' initComponent
go.modules.tutorial.music.MainPanel.superclass.initComponent.call(this);

//Attach lister to changes of the filter selection.
//add buffer because it clears selection first and this would cause it to fire twice
this.genreFilter.getSelectionModel().on('selectionchange', this.onGenreFilterChange, this, {buffer: 1});

// Attach listener for running the module
this.on("afterrender", this.runModule, this);

createArtistGrid: function() {
    this.artistGrid = new go.modules.tutorial.music.ArtistGrid({
        region: "center",
        tbar: [
            //this class will hide the button on large screens
            cls: 'go-narrow',
            iconCls: "ic-menu",
            handler: function () {
                this.genreFilter.show();
            },
            scope: this
        ],
        xtype: 'tbsearch'
    },

    // add button for creating new artists
    this.addButton = new Ext.Button({
        iconCls: 'ic-add',
        tooltip: t('Add'),
        handler: function (btn) {
            var dlg = new go.modules.tutorial.music.ArtistDialog({
                formValues: {
                    // you can pass form values like this
return this.artistGrid;
},

// Fired when the Genre filter selection changes
onGenreFilterChange: function (sm) {
    var selectedRecords = sm.getSelections(),
        ids = selectedRecords.column('id'); // column is a special GO method that get's all the id's from the records in an array.
    this.artistGrid.store.setFilter('genres', {genres: ids});
    this.artistGrid.store.load();
},

createGenreFilter: function () {
    this.genreFilter = new go.modules.tutorial.music.GenreFilter({
        region: "west",
        width: dp(300),
        autoHeight: true,

        // render a split bar for resizing
        split: true,

        tbar: [{
            xtype: "tbtitle",
            text: t("Genres")
        },
        '->',
        }]
    this.genreFilter.initDisplay();
    this.genreFilter.rightsided();
    this.artistGrid.contentPane.add(this.genreFilter);
    this.genreFilter.onSelectChange = function () {
        var count = this.genreFilter.count(),
            selected = this.genreFilter.getSelections();
        this.artistGrid.store.setFilter('genres', {genres: selected});
        this.artistGrid.store.load();
    }; // this function was added in place of the original one.
},

// Fired when the Artist Grid's row dbl click
onGridDblClick: function (rec) {
    var id = rec.data.id;
    this.browserWin = new go.Modules.Tutorial.BrowserWindow({
        title: t("Select the Artist"),
        width:(dp(500)),
        height: dp(500),
        region:"center",
        closed: function () {
            this.browserWin = null;
        },
        items: []
    });

    var dlg = this.browserWin.items[0];
    dlg.findIndex(id);
    dlg.show();
},

scope: this
});

iconCls: 'ic-more-vert',
tooltip: t("More options"),
menu: [
    {
        itemId: "delete",
        iconCls: 'ic-delete',
text: t("Delete"),
handler: function () {
    this.artistGrid.deleteSelected();
},
scope: this
}
],

listeners: {
    rowdblclick: this.onGridDblClick,
    scope: this
}]);

return this.artistGrid;
//add back button for smaller screens
{
    //this class will hide it on larger screens
    cls: 'go-narrow',
    iconCls: "ic-arrow-forward",
    tooltip: t("Artists"),
    handler: function () {
        this.artistGrid.show();
    },
    scope: this
}
];
return this.genreFilter;
,
createFilterPanel: function() {
    return new Ext.Panel({
        autoHeight: true,
        tbar: [
            {
                xtype: 'tbtitle',
                text: t("Filters")
            },
            '->',
            {
                xtype: "button",
                iconCls: "ic-add",
                handler: function() {
                    var dlg = new go.filter.FilterDialog({
                        entity: "Artist"
                    });
                    dlg.show();
                },
                scope: this
            }
        ],
        items: [
            this.filterGrid = new go.filter.FilterGrid({
                filterStore: this.artistGrid.store,
                entity: "Artist"
            })
        ]
    });
},

// Fired when the module panel is rendered.
runModule: function () {
    // when this panel renders, load the genres and artists.
    this.genreFilter.store.load();
    this.artistGrid.store.load();
},

// Fires when an artist is double clicked in the grid.
onGridDblClick: function (grid, rowIndex, e) {
    //check permissions
Again: study the code. The following changes were made:

1. The artist grid has been put in a wrapper function
2. The Genre Filter has been put in another wrapper function
3. A new function `createFilterPanel()` has been added, which renders a filter panel
4. The Genre Filter and Filter Panel were merged into the left panel

**Note:** When defining a new `Ext.Panel` element, make sure to set `autoHeight` to true. Otherwise, your panels may look squashed.

Save and reload your module. The main screen should look somewhat like this:

Now we have to define the custom filter:

- Click the ‘Add’ button, next to the ‘Filters’ header in the left panel
- In the new modal, enter the following parameters:

** Enter a name that makes sense, e.g. ‘Active’ ** Make sure that the option ‘Match ALL conditions’ is selected ** Select the field ‘Active’ ** Select the value ‘Yes’ * In the ‘Permissions’ tab, you can play around with permissions if you wish to * Click ‘Save’
You will notice that your custom filter has been added:

Toggling the custom filter will filter your artist list by the ‘active’ status.

**30.2.11 The end**

You have finished part one! Now head to part two, in which we learn to add ACL to the client side part of the module.

**30.3 Building a webclient module, part 2**

In which we build access control into the web client.

In the server module tutorial we built one `AclOwnerEntity` model, being Review. The web client was created in the webclient tutorial, without the review part. This part of the webclient tutorial covers the client-side part of working with access control lists.

For the sake of this tutorial, we made it possible for reviews to be shared with your fellow users, or being hidden since nobody needs to know about your guilty pleasures. :-)

We will be creating a modal in which to add or edit reviews. Furthermore, there will be a window, in which shared reviews are displayed. In order to be a bit flashy, we borrow some code from the comments module and display the reviews in a somewhat playful manner.

**30.3.1 Add the Review model to the store**

The first step is to make sure that reviews can be retrieved and managed through the store. In `Module.js` we add a new entity named `Review`:
entities: {
    // Rest of the entities
    {
        name: "Review",
        relations: {
            creator: {store: "User", fk: "createdBy"},
            modifier: {store: "User", fk: "modifiedBy"}
        }
    }
},

Please note that a useful relation would be the album relation. After all, a review is connected to one album. Since we defined album as a property of artist, you will not be able to retrieve this relation from the store.

### 30.3.2 Artist Detail Panel

Both modals should be called from the `ArtistDetail.js` file (which renders the artist detail panel) and there should be a way to distinguish between albums with or without reviews.

In order to achieve this, we update the `InitComponent` like this:

```
initComponent: function () {
    this.tbar = this.initToolbar();
    // Render a 'new review' modal
    this.addReviewModal = function (v) {
        var dlg = new go.modules.tutorial.music.ReviewDialog();
        dlg.setValues({albumId: v.id});
        dlg.show();
    };
    // Render all reviews for the current album in a window (which is offered as a modal)
    this.showReviewsModal = function (v) {
        var dlg = new go.modules.tutorial.music.ReviewsModal();
        dlg.store.setFilter('albumId', {albumId: v.id});
        dlg.store.load();
        dlg.show();
    };
    Ext.apply(this, {
        // all items are updated automatically if they have a "tpl" (Ext.XTemplate) property or an "onLoad" function. The panel is passed as argument.
        items: [
            //Artist name component
            {
                cls: 'content',
                xtype: 'box',
                tpl: '<h3>{name}</h3>'
            },
            //Render the avatar
            {
                xtype: "box",
                cls: "content",
                tpl: new Ext.XTemplate('<div class="go-detail-view-avatar">
                                    <div class="avatar" style="{[this.getStyle(values.photo)]}"></div></div>',
                                    <div class="avatar" style="{[this.getStyle(values.photo)]}"</div></div>',
            }
        ]
    });
```

(continues on next page)
getStyle: function (photoBlobId) {
  return photoBlobId ?
    'background-image: url(' + go.Jmap.downloadUrl(photoBlobId) + ')" : "";
}

// Albums component, render number of reviews
{
  collapsible: true,
  title: t("Albums"),
  xtype: "panel",
  listeners: {
    scope: this,
    afterrender: function (box) {
      box.getEl().on('click', function (e) {
        // don't execute when user selects text
        if (window.getSelection().toString().length > 0) {
          return;
        }
        var container = box.getEl().dom.childNodes[1],
        item = e.getTarget("a", box.getEl()),
        i = Array.prototype.indexOf.call(container.getElementsByTagName("a"), item);
        if (i >=0) {
          var album = go.util.Object.convertMapToArray(this.data.albums, 'id')[i];
          if (album.reviews.length > 0) {
            this.showReviewsModal(album);
          } else {
            this.addReviewModal(album);
          }
        }
      }, this);
    }
  }
}

tpl: new Ext.XTemplate('<div class="icons">
  <tpl for="go.util.Object.values(values.albums)"
    displayNumReviews="function(v) {
      v = v || null;
      if (v === null) {
        return "";
      }
      var displayNum = Math.ceil(v / 10) * 10;
      return displayNum;
    }"
    heading="Albums"
    releaseDate="[this.displayNumReviews(values.releaseDate)]""
    }
    />
</div>','

(continues on next page)
Study the code. The following things were added:

1. A link was added to the album overview items. Dependent on the number of known reviews, the link text is altered

2. If there are no reviews, a Review modal will be opened in which you can add a new review

3. Otherwise, a window is opened which displays current reviews.

30.3.3 Review modal

The next step is adding a modal, in which to enter a review. Please note that an albumId is supplied from the addReviewModal method:

dlg.setValues({albumId:v.id});

30.3. Building a webclient module, part 2
Create a new javascript file, name it `ReviewDialog.js` and enter the following code:

```javascript
const ReviewDialog = Ext.extend(go.form.Dialog, {
    stateId: 'album-review',
    title: t("Review"),
    entityStore: "Review",
    width: dp(800),
    height: dp(600),
    maximizable: false,
    collapsible: false,
    modal: true,

    initFormItems: function () {
        this.addPanel(new go.permissions.SharePanel());

        var items = [{
            xtype: 'fieldset',
            anchor: "100% 100%",
            items: [{
                xtype: 'textfield',
                name: 'title',
                fieldLabel: t("Title"),
                anchor: '100%',
                allowBlank: false
            },
            {
                xtype: 'radiogroup',
                fieldLabel: t("Rating"),
                name: "rating",
                value: null,
                items: [
                    {boxLabel: t("It stinks"), inputValue: 1},
                    {boxLabel: t("Meh"), inputValue: 2},
                    {boxLabel: t("It's OK"), inputValue: 3},
                    {boxLabel: t("It's pretty good"), inputValue: 4},
                    {boxLabel: t("A stroke of genius"), inputValue: 5}
                ]
            },
            {
                xtype: 'xhtmleditor',
                name: 'body',
                fieldLabel: "",
                hideLabel: true,
                anchor: '0 -90',
                allowBlank: false,
                listeners: {
                    scope: this,
                    ctrlenter: function () {
                        this.submit();
                    }
                }
            }
        }]
    }];

    return items;
});
```

(continues on next page)
onLoad : function(entityValues) {
    this.supr().onLoad.call(this, entityValues);
};

... and add the line:

ReviewDialog.js

to the bottom of your scripts.txt file.

The code is pretty straightforward, but please note a few things:

• The albumId field does not need to be defined, since the albumId value is already passed from the artistDetail panel;

• Permission management is added by the following line:

  this.addPanel(new go.permissions.SharePanel());

That’s it. We can now add our own review to the selected album:

### 30.3.4 Reviews screen

In the reviews screen, a number of things need to be checked:

1. A user may enter only one review for a certain album.

2. It must be possible to read, edit or delete other reviews, depending on the ACL settings entered by the creator.

3. Upon adding or deleting a review, the Artist store is to be reloaded. Please note that we have already implemented in the Review model.

A new javascript file is to be created and added to the scripts.txt file. We name it ReviewsModal.js.

go.modules.tutorial.music.ReviewsModal = Ext.extend(go.Window, {
    stateId: 'album-reviews',
    title: t("Reviews"),
    width: dp(1000),
    height: dp(800),
    maximizable: true,
    collapsible: false,
    modal: true,
    stateful: true,
    layout: 'fit',
    initComponent: function () {
        this.tools = [{
            id: "add",
            handler: function () {
                var dlg = new go.modules.tutorial.music.ReviewDialog();
                dlg.setValues({albumId: this.albumid});
                dlg.show();
            }
        }];
        this.store = new go.data.Store({

(continues on next page)
Do you see any Teletubbies in here? Do you see a slender plastic tag clipped to my shirt with my name printed on it? Do you see a little Asian child with a blank expression on his face sitting outside on a mechanical helicopter that shakes when you put quarters in it? No? Well, that's what you see at a toy store. And you must think you're in a toy store, because you're here shopping for an infant named Jeb.
### Review

#### General

<table>
<thead>
<tr>
<th>Name</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admins</td>
<td>Manage</td>
</tr>
<tr>
<td>admin</td>
<td></td>
</tr>
<tr>
<td>demo</td>
<td></td>
</tr>
<tr>
<td>elmer</td>
<td>Write and delete</td>
</tr>
<tr>
<td>Everyone</td>
<td></td>
</tr>
<tr>
<td>Internal</td>
<td></td>
</tr>
<tr>
<td>linda</td>
<td></td>
</tr>
</tbody>
</table>

**Admins**
System Administrator

**admin**
System Administrator

**demo**
Demo User

**elmer**
Elmer Fudd

**Everyone**
System Administrator, Elmer Fudd, Demo User, Linda Smith

**Internal**
Elmer Fudd, Demo User, Linda Smith

**linda**
Linda Smith
fields: [
  'id',
  'title',
  'body',
  'rating',
  'albumTitle',
  'createdBy',
  {name: 'creator', type: "relation"},
  'albumId', 'aclId', "permissionLevel"
],
entityStore: "Review"
});

// Use a Group Office store that is connected with an go.data.EntityStore for
// automatic updates.
this.store.on('load', function (store, records, options) {
  this.updateView();
  this.updateTitle();
  this.toggleAddBtn();
}, this);

this.store.on('remove', function () {
  this.updateView();
  this.toggleAddBtn();
}, this);

this.on('destroy', function () {
  this.store.destroy();
}, this);

this.on("expand", function () {
  this.updateView();
}, this);

// Add a simple context menu. Make sure that the correct permissions are set
this.contextMenu = new Ext.menu.Menu({
  items: [{
    iconCls: 'ic-delete',
    text: t("Delete"),
    handler: function () {
      Ext.MessageBox.confirm(t("Confirm delete"), t("Are you sure you want to delete this item?")), function (btn) {
        if (btn !== "yes") {
          return;
        } go.Db.store("Review").set({destroy: [this.contextMenu.record.id]});
      }, this);
    },
    scope: this
  },
  { iconCls: 'ic-edit',
    text: t("Edit"),
    handler: function () {
      var dlg = new go.modules.tutorial.music.ReviewDialog();
    }
  });
dlg.load(this.contextMenu.record.id).show();
,
scope: this
})
});

var cntrClass = Ext.extend(Ext.Container, {
  initComponent: function () {
    Ext.Container.superclass.initComponent.call(this);
    Ext.applyIf(this, go.panels.ScrollLoader);
    this.initScrollLoader();
    },
    store: this.store,
    scrollUp: true
  });

this.items = [
  this.commentsContainer = new cntrClass({
    region: 'center',
    autoScroll: true
  })
];

go.modules.tutorial.music.ReviewsModal.superclass.initComponent.call(this);
,
updateView: function () {
  this.commentsContainer.removeAll();
  this.store.each(function (r) {
    var mineCls = r.get("createdBy") == go.User.id ? 'mine' : '';
    var readMore = new go.detail.ReadMore({
      cls: mineCls
    });
    var creator = r.get("creator");
    if (!creator) {
      creator = {
        display: t("Unknown user")
      };
    }
    var avatar = {
      xtype: 'box',
      autoEl: {
        tag: 'span', 'ext:qtip': t('{author} wrote: ') .replace('{author}', creator.display)
      ,
      cls: 'photo ' + mineCls
    };
    if (creator.avatarId) {
      avatar.style = 'background-image: url(' + go.Jmap.thumbUrl(creator.avatarId, {
        w: 40,
        h: 40,
        zc: 1
      }) + ');background-color: transparent;';
    } else {
      avatar.html = go.util initials(creator.display);
      avatar.style = 'background-image: none';
    }
  });
};
readMore.setText(this.getReviewText(r));

this.commentsContainer.add(
    {
        xtype: "container",
        cls: 'go-messages',
        items: [{
            xtype: 'container',
            label: t("Creator"),
            items: [avatar, readMore]
        }]
    });

// Add a context menu, make permissions dependent on ACL
readMore.on('render', function (me) {
    me.getEl().on("contextmenu", function (e, target, obj) {
        e.stopEvent();
        if (r.data.permissionLevel >= go.permissionLevels.write) {
            this.contextMenu.record = r;
            this.contextMenu.showAt(e.xy);
        }
    }, this);
}, this);

this.doLayout();
var height = 7; // padding on composer
this.commentsContainer.items.each(function (item, i) {
    height += item.getOuterSize().height;
});

// Update window title by adding the album title
updateTitle: function () {
    var r = this.store.getAt(0), title = this.title;
    if (typeof (r) !== "undefined") {
        this.setTitle(t("Reviews") + " for " + Ext.util.Format.htmlEncode(r.get('albumTitle')));
    } else {
        this.setTitle(t("Reviews"));
    }
},

// Check whether current user had added a review. If they have, hide the add button.
toggleAddBtn: function () {
    if (this.store.query('createdBy', go.User.id).getCount() > 0) {
        this.tools.add.hide();
    } else {
        var r = this.store.getAt(0);
        if (typeof (r) !== "undefined") {
            this.tools.add.albumid = r.get("albumId");
        }
    }
},

(continues on next page)
In this file, a number of things happen:

1. The modal is based on the (relatively empty) `go.Window` class. A number of sane default settings is preconfigured.

2. In the `initComponent` function, a new store is defined. Please note that the `aclId` and `permissionLevel` fields are being retrieved. We will need these later.

3. A context menu is added.

4. Each review is being rendered in a container class. We borrowed the layout from the 'comments' module to make it look nice.

5. For each review, the permission level is matched with the user’s permissions. If applicable, the user can use the context menu.

### 30.3.5 The End

This concludes our weblncient tutorial. There is one thing left to say: have fun coding your own Group-Office modules!

### 30.4 Server API

#### 30.4.1 Command Line Interface

Group Office also has a Command Line Interface. To implement this do the following:

1. Create a controller that extends the abstract controller in folder 'go/modules/tutorial/music/cli/controller':

```php
<?php
namespace go\modules\tutorial\music\cli\controller;
use go\core\Controller;

class CliDemo extends Controller {
    public function hello($name = "World") {
        echo "Hello $name!\n";
    }
}
```

2. Run (Notice the path is case sensitive!):
Reviews for Bloodflowers

Back to the disintegration era

🌟🌟🌟🌟

This album is seen as a third golden age, a return to the "disintegration" era.

Great title

🌟🌟🌟🌟🌟

It has some happy-sad avant-la-lettre songs, as well as some very melodramatic songs. The title is what gets me though.
Group-Office manual

sudo -u www-data php /usr/share/groupoffice/cli.php tutorial/music/CliDemo/hello ˓→-name=Merijn

or with docker-compose:
docker-compose exec --user www-data groupoffice-master php /usr/local/share/
˓→groupoffice/cli.php tutorial/music/CliDemo/hello --name=Merijn

3. This should output:
Hello Merijn!

Note: Command line methods always run as the administrator user so you should probably need to sudo into the web
user (www-data)

Specify config file
If you have a multi instance environment you can specify the config file location with the ‘c’ argument:
php /usr/share/groupoffice/cli.php tutorial/music/CliDemo/hello --name=Merijn -c=/etc/
˓→groupoffice/multi_instance/domain.com/config.php

30.4.2 Database Abstraction Layer
Group Office has a powerful database abstraction layer to make database queries. It’s designed to be as close to regular
SQL as possible but to minimize the risk of security problems such as SQL injections.
For testing these queries I would suggest to use a Command Line Interface. controller. You can simply extend the
CliDemo controller from the Music demo module.
Select query
First create a helper function in the controller used to print results:
private function printQuery(Query $query) {
//Query objects are traversable, but you can also use $query->single() to
//fetch a single record or $query->all() to fetch all in an array.
foreach($query as $record) {
foreach($record as $key => $value) {
echo $key . ": " . var_export($value, true) . "\n";
}
echo "-----\n";
}
}

Then create this select function. It demonstrates basic usage. It also shows you can easily print SQL queries for
debugging.
public function select() {
//Build new select query object
(continues on next page)

30.4. Server API

289


$query = (new Query())
    ->select('*')
    ->from('music_artist')
    ->orderBy(["name" => 'ASC']);

//Query objects can be stringified for debugging
echo $query . "\n";

$this->printQuery($query);

//Quick way to fetch single result
$record = (new Query)
    ->select('id, name')
    ->from('music_artist')
    ->single();

var_dump($record);
}

Run these queries by executing:

    --php community/music/cliDemo/select

Joins

The next example shows how to make a join with a GROUP BY.

public function join() {
    // Join example with aliases for the table names
    $query = (new Query)
        ->select('art.id, art.name, count(alb.id) as albumCount')
        ->from('music_artist', 'art')
        ->join('music_album', 'alb', 'art.id = alb.artistId')
        ->groupBy(['art.id']);

    $this->printQuery($query);
}

Where conditions

Here's the simplest example:

$query = (new Query)
    ->select('id, name')
    ->from('music_artist')
    ->where('name', '=' , 'The Doors');

The same query can be written like this:

$query = (new Query)
    ->select('id, name')
Or with a string and bind parameters:

```php
$query = (new Query)
    ->select('id, name')
    ->from('music_artist')
    ->where('name = :name')
    ->bind(['name' => 'The Doors']);
```

Array values are automatically processed as IN conditions:

```php
$query = (new Query)
    ->select('id, name')
    ->from('music_artist')
    ->where(['name' => ['The Doors', 'The war on drugs']]);
```

You can use a gocoredbCriteria object for sub groups:

```php
$query = (new Query)
    ->select('id, name')
    ->from('music_artist')
    // Select only artists that were created in the past 3 hours.
    ->where('createdAt', '>=', new DateTime("-3 hours"))
    ->andWhere(
        // This will become a grouped where condition between parenthesis
        (new Criteria())
        ->where(['name' => 'The Doors'])
        ->orWhere(['name' => 'The war on drugs'])
    );
```

### Sub queries

Here’s an example of a WHERE EXISTS subquery:

```php
$query = (new Query)
    ->select('id, name')
    ->from('music_artist', 'art')
    ->whereExists(
        (new Query)
        ->select('*')
        ->from('music_album', 'alb')
        ->where('alb.artistId = art.id')
    );
```

And here’s how to do an IN subquery:

```php
$query = (new Query)
    ->select('id, name')
    ->from('music_album', 'alb')
    ->where('artistId', 'IN',
        (new Query)
        ->select('id')
        ->from('music_artist', 'art')
    );
```
Expressions

Sometimes it’s useful to pass raw expressions to the Query object. You can use a gocoredbExpression object to do this. Here’s an example:

```php
$query = (new Query)
    ->select('art.id, art.name, count(alb.id) as albumCount')
    ->from('music_artist', 'art')
    ->join('music_album', 'alb', 'art.id = alb.artistId')
    ->groupBy(['art.id'])
    // Normally group by expects a [column => ASC] array but you can use
    // functions with an Expression object
    ->orderBy([new Expression("count(alb.id) DESC")]);
```

30.4.3 Object Relational Mapping

Group Office features an ORM package in go\core\orm.

There are two main types of models:

1. Entity: A model that can be changed directly by the user. Such as a Contact.
2. Property: A model that belongs to an Entity. Such as a Contact’s E-mail address.

Mapping

Each model must implement the defineMapping method. This mapping maps database tables to the object and defines other models as properties.

For example the go\modules\community\music\model\Artist model defines:

```php
protected static function defineMapping() {
    return parent::defineMapping()
        ->addTable("music_artist", "a")
        ->addArray('albums', Album::class, ['id' => 'artistId']);
}
```

You can see that it maps a table music_artist and adds a property ‘albums’ as an array. If you’d like to retrieve relations between entity models, use the addScalar method instead of the addArray method.

Furthermore, one can define custom properties from database queries by using the setQuery method. For example, one could define an albumcount property for an artist like this:

```php
protected static function defineMapping() {
    return parent::defineMapping()
        ->addTable("music_artist", "art")
        ->setQuery((new Query())->select('COUNT(alb.id) AS albumcount')
            ->join('music_album', 'alb', 'art.id = alb.artistId')
            ->groupBy(['alb.artistId']) );
}
```
A property must be defined for all database columns the model should use. There should also be a public $albums property.

Note: Mappings are cached for performance. When making changes you need to run /install/upgrade.php to rebuild the cache. You can also disable cache in the config.php file:

```php
$config['core'] = [
    'general' => [
        'cache' => 'go\core\cache\None'
    ]
];
```

**addTable() method**

With the `addTable()` method you map table database columns to object properties. All protected and public properties of the object that match a database column name will be loaded from and saved to that table. You can add multiple tables to one entity. The primary keys must match or a key mapping can be passed to the method. See the code documentation for more details.

**addArray() method**

If you wish to retrieve the full set of um.. properties of a property for an entity, you can use the `addArray()` method. For instance, if you want to retrieve entire albums for an artist as per the tutorial, your `defineMapping()` method for the Artist model should look like this:

```php
protected static function defineMapping() {
    return parent::defineMapping()
        ->addTable('music_artist', 'artist')
        ->addArray('albums', Album::class, ['id' => ...
        ->'artistId']);
}
```

The `addArray()` method yields a flat array of properties, i.e. an array that does not use a key-value pair.

**addMap() method**

If you prefer to retrieve relationships as key-value pairs, for instance of relation management, use the `addMap()` method:

```php
protected static function defineMapping() {
    return parent::defineMapping()
        ->addTable('music_artist', 'artist')
        ->addMap('albums', Album::class, ['id' => 'artistId']);
}
```

This method returns an object, in which the `id` of a relation entity serves as a key. Its values are nested inside the object.
**Group-Office manual**

**addScalar() method**

The `addScalar()` method retrieves an array of id fields for related items. If there are no known relations, an empty array is returned.

**Note:** This is a way to relate entities to properties. However, it is more common practice to define such relatable items as entities.

**addHasOne() method**

A special relation is the `addHasOne()` method. It is commonly used with a `UserSettings` model, which gives a default entity of a certain type to each user.

For instance, the address book module has the following lines in its module.php file:

```php
public static function onMap(Mapping $mapping) {
    $mapping->addHasOne('addressBookSettings', UserSettings::class, ['id' => 'userId'], true);
}
```

This will create a new default address book for each new user and will assign it as default address book.

**addRelation() method**

With `addRelation()` you can map “Property” models with a has one or has many relation. These properties will be loaded and saved automatically.

**Note:** You can’t add relations to other entities. Only “Property” models can be mapped. Fetch other entities in the client by key. If you would implement a `getOtherEntity()` method, it would be very hard to synchronize the entities to clients. Because each entity keeps its own sync state. If the “OtherEntity” changes it would mean that this entity would change too.

If you need to create a method to retrieve another entity on the server side only then it’s recommended to name it “findOtherEntity()” so it won’t become a public API property.

**Getters and Setters**

All models can implement get and set methods to create API properties.

For example if you have a property “foo” in the database but this property needs some processing when you get or set it. You can make this property “protected”.

**Note:** You should never make database properties private because then the parent class can’t access it for saving and loading.

In this example the property “foo” is JSON encoded in the database but turned into an array in the API:
protected $foo;

public function setFoo($value) {
    $this->foo = json_encode($value);
}

public function getFoo() {
    return json_decode($value, true);
}

### Working with entities

You can find entities with the find() and findById() method.

**Note:** The method find() returns a Query object. You can read more on that in the *Database Abstraction Layer* chapter.

Here's how to find the first Artist entity.

```php
$artist = \go\modules\community\music\model\Artist::find()->single();
echo json_encode($artist);
```

This will output the artist in JSON format:

```json
{
    "permissionLevel": 50,
    "name": "De Scherings",
    "createdAt": "2018-08-17T14:42:17+00:00",
    "modifiedAt": "2018-08-24T12:42:20+00:00",
    "createdBy": 1,
    "modifiedBy": 1,
    "albums": [
        {
            "artistId": 3,
            "name": "Good times",
            "releaseDate": "2018-08-24T00:00:00+00:00",
            "genreId": 2
        }
    ],
    "photo": "a1a82b74532fcd822f0923cd84ab23533eb92d5f",
    "id": "3"
}
```

Here's how to create a new one with an album:

```php
$artist = new Artist();
$artist->name = "The Doors";
$artist->albums[] = (new Album())->setValues(["name" => 'The Doors', 'releaseDate' => new DateTime('1968-01-04'), 'genreId' => 2]);

if(!$artist->save()) {
    echo "Save went wrong: ". var_export($artist->getValidationErrors(), true) . "\n";
} else {
```

(continues on next page)
Or you can use “setValues” this is what the JMAP API uses when it POSTS values in JSON:

```php
$artist = (new Artist)
    ->setValues(
        [
            'name' => 'The War On Drugs',
            'albums' => [
                ['name' => 'Album 1', 'releaseDate' => new DateTime('2018-01-04'), 'genreId' => 2],
                ['name' => 'Album 2', 'releaseDate' => new DateTime('2018-01-04'), 'genreId' => 2]
            ]
        ]
    );
if(!$artist->save()) {
    echo "Save went wrong: ". var_export($artist->getValidationErrors(), true) . "\n";
} else {
    echo "Artist saved!\n";
}
```

**Cascading delete**

It’s recommended to take advantage of the database foreign keys to cascade delete relations. This is much faster than deleting relations in code. It does however cause a problem in the JMAP sync protocol. Because these deletes are not automatically registered as a change. You can use Entity::getType()->change() and Entity::getType()->changes() for an example. See the address books’s Group entity for an example.

### 30.4.4 Events

Because Group Office is a modular system it has events. Modules can register listeners to Objects implementing the “go\core\event\EventEmitterTrait” trait.

All entities do this already. They fire an event on:

1. save
2. delete
3. mapping, so you can add new relational properties.

**Adding listeners**

The community/googleauthenticator uses events for example, to add a “googleauthenticator” property to the User entity.

Listeners are always defined in the Module.php file of the module. A method “defineListeners” can be created for it. Look at go/modules/community/googleauthenticator/Module.php for the complete file.

```php
public function defineListeners() {
    User::on(Property::EVENT_MAPPING, static::class, 'onMap');
}
```
public static function onMap(Mapping $mapping) {
    $mapping->addRelation("googleauthenticator", model\Googleauthenticator::class,
        ['id' => 'userId'], false);
    return true;
}

Firing events

To fire an event your object must use the “go\core\event\EventEmitterTrait” trait.
Then define an EVENT_-* constant and document your event:

```php
/**
 * Something very special happens here.
 */
const EVENT_MYEVENT;
```

Then you can call:

```php
$this->fireEvent(self::EVENT_MYEVENT, $this, "arg2", "arg3");
```

30.4.5 Filters

Entities can be filtered according to the JMAP filter specification.
You can implement filters by overriding the “defineFilters” method in your Entity. Here’s an advanced example of the Contact filters:

```php
/**
 * Defines JMAP filters
 *
 * Adds the 'genres' filter which can be an array of genre id's.
 *
 * @link https://jmap.io/spec-core.html#/query
 *
 * @return Filters
 */
protected static function defineFilters() {
    return parent::defineFilters()
        ->add("addressBookId", function(Criteria $criteria, $value) {
            $criteria->andWhere('addressBookId', '=', $value);
        })
        ->add("groupId", function(Criteria $criteria, $value, Query $query) {
            $query->join('addressbook_contact_group', 'g', 'g.contactId = c.id');
            $criteria->andWhere('g.groupId', '=', $value);
        });
}
```
->add('isOrganization', function(Criteria $criteria, $value) {
    $criteria->andWhere('isOrganization', '=', $value);
})
->add('hasEmailAddresses', function(Criteria $criteria, $value, Query $query) {
    $query->join('addressbook_email_address', 'e', 'e.contactId = c.id', "LEFT")
    ->groupBy(['c.id'])
    ->having('count(e.id) '.'$value ? '>: ']='.'0');
})
->addText('email', function(Criteria $criteria, $comparator, $value, Query $query) {
    $query->join('addressbook_email_address', 'e', 'e.contactId = c.id', "INNER");
    $criteria->where('e.email', $comparator, $value);
})
->addText('name', function(Criteria $criteria, $comparator, $value) {
    $criteria->where('name', $comparator, $value);
})
->addText('country', function(Criteria $criteria, $comparator, $value, Query $query) {
    if(!
        $query->isJoined('addressbook_address')) {
        $query->join('addressbook_address', 'adr', 'adr.contactId = c.id', "INNER");
    }
    $criteria->where('adr.country', $comparator, $value);
})
->addText('city', function(Criteria $criteria, $comparator, $value, Query $query) {
    if(!
        $query->isJoined('addressbook_address')) {
        $query->join('addressbook_address', 'adr', 'adr.contactId = c.id', "INNER");
    }
    $criteria->where('adr.city', $comparator, $value);
})
->addNumber('age', function(Criteria $criteria, $comparator, $value, Query $query) {
    if(!
        $query->isJoined('addressbook_date')) {
        $query->join('addressbook_date', 'date', 'date.contactId = c.id', "INNER");
    }
    $criteria->where('adr.city', $comparator, $value);
})
When this has been implemented you can do use it in a Foo/query call like this:

```json
{ "filter": { "addressBookId": [1, 2] } }
```

### Quick search

By default a “q” filter is already defined. You can use it by specifying the text fields to search in the method “search-Columns”:

```php
protected static function searchColumns() {
    return ['name'];
}
```

You can use it like this:

```json
{ "filter": { "q": "test" } }
```
Group-Office manual

ACL Permission level

If the entity is derived from an AclEntity then the filter “permissionLevel” is also available:

```json
{"filter": {"permissionLevel" : GO.permissionLevels.write}}
```

By default entities are filtered on read permissions.

Extend filters with modules

You can also extend filters using the Entity::EVENT_FILTER event.

For example:

```php
public function defineListeners() {
    User::on(User::EVENT_FILTER, static::class, 'onUserFilter');
}

/**
 * Extends the User filters with "isIntermediair". So we can show only
 * users that are being an intermediair.
 *
 * @param Filters $filters
 */
public static function onUserFilter(Filters $filters) {
    $filters->add('isIntermediair', function(Criteria $criteria, $value, Query $query) {
        if($value) {
            $query ->join('applications_application', 'a', 'a.createdBy = u.id')
                    ->groupBy(['u.id']);
            //We don't want to use the Users acl but the applications acl.
            Acl::applyToQuery($query, 'a.aclId');
        }
    });
}
```

30.4.6 Binary Large Objects

Group Office has a BLOB system to store files. When uploading a file a unique hash is calculated for the file to identify it. So when the same file is stored more than once in Group Office it will only be saved to disk once. You don’t have to worry about uploading or downloading the data. Because this has already been implemented for you.

Database

In the database you can store the BLOB id in a BINARY (40) type column.

**Warning:** It’s very important that a foreign key constraint is defined for the BLOB id when it’s used in a table. Because the garbage collection mechanism uses these keys to determine if a BLOB is stale and to be cleaned up. In other words if you don’t do this your BLOB data will be removed automatically.
**Download / Upload URL's**

You can get the download and upload URL's by doing a GET request to the auth.php endpoint. 
Also read: https://jmap.io/spec-core.html#binary-data

**Using**

In the webclient you can use for uploading:


And for downloading:


**30.4.7 Custom fields**

TODO...

**30.4.8 Router**

Group-Office has 3 routers:

1. The JMAP router for routing JMAP requests. See https://jmap.io/spec-core.html
2. The CLI router for routing Command Line Interface commands
3. A download hash for routing download id’s to a method.

**JMAP Router**

The standard JMAP calls are already built in. Every Entity is registered in the “core_entity” table. The router uses this table to find the corresponding module. For example method call “Contact/get” is mapped to PHP class:

```php
/go/modules/community/addressbook/controller/Contact
```

and public method:

```php
get($params);
```

All parameters are passed in the $params argument.

**CLI Router**

There’s also a command line interface router. Read more about that here
**Download router**

All downloads go through “download.php?blob=<HASH>”. The <HASH> is typically a hash pointing to a BLOB on the system. But it can also contain a route to a special download method defined in the Module.php file of the module. Methods prefixed with “download” can be accessed. For example method:

```
module\community\addressbook\Module::downloadVcard($contactId);
```

can be accessed with:

```
download.php?blob=community/addressbook/vcard/1
```

### 30.4.9 Account settings

Often it is necessary to create user account settings for modules. You can do that by extending the User entity with properties.

Create a new property model. For example:

```php
<?php
namespace go\modules\community\addressbook\model;
use go\core\orm\Property;

class UserSettings extends Property {
    /**
     * Primary key to User id
     *
     * @var int
     */
    public $userId;

    /**
     * Default address book ID
     *
     * @var int
     */
    public $defaultAddressBookId;

    protected static function defineMapping() {
        return parent::defineMapping()->addTable("addressbook_user_settings", 
        "abs");
    }
}
```

Also create a database table for it:

```sql
CREATE TABLE IF NOT EXISTS `addressbook_user_settings` (
    `userId` int(11) NOT NULL,
    `defaultAddressBookId` int(11) DEFAULT NULL,
    PRIMARY KEY (`userId`),
    KEY `defaultAddressBookId` (`defaultAddressBookId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT;
```
ALTER TABLE `addressbook_user_settings`  
ADD CONSTRAINT `addressbook_user_settings_ibfk_1` FOREIGN KEY (`userId`)  
→ REFERENCES `core_user` (`id`) ON DELETE CASCADE,  
ADD CONSTRAINT `addressbook_user_settings_ibfk_2` FOREIGN KEY  
→(`defaultAddressBookId`) REFERENCES `addressbook_addressbook` (`id`) ON DELETE SET  
→ NULL;

In the Module.php we can use the “Property::EVENT_MAPPING” event to dynamically extend the User entity with this property:

```php
<?php
namespace go\modules\community\addressbook;

use go\core\module\Base;
use go\modules\community\addressbook\model\Contact;
use go\modules\core\links\model\Link;
use go\modules\core\users\model\User;
use go\modules\community\addressbook\model\UserSettings;

class Module extends Base {
    public static function defineListeners() {
        // When the User mapping is created listen call static::onMap
        User::on(Property::EVENT_MAPPING, static::class, 'onMap');
    }

    public static function onMap(Mapping $mapping) {
        // Add the relation to the User mapping
        $mapping->addRelation('addressBookSettings', UserSettings::class, ['id'  
→' => 'userId'], false);
    }
}
```

After these changes you must run install/upgrade.php to rebuild the cache.

Now the User entity has this new property:

```json
{
    id: 1,
    username: "admin",
    addressBookSettings: {
        defaultAddressBookId: 1
    }
}
```

You can now implement a settings pane in the webclient.

### 30.4.10 System settings

The system settings can be extended by modules. All system settings are stored in the core_settings table.

To create new settings you must implement a settings module:
<?php
namespace go\modules\core\users\model;

use go\core;
use go\core\db\Query;

class Settings extends core\Settings {
    /**
     * Default time zone for users
     *
     * @var string
     */
    public $defaultTimezone = "Europe/Amsterdam";
}

This would create a new module with a single setting called “defaultTimezone”.
You also need a controller to handle API calls.

30.5 Webclient API

30.5.1 Best practice

Structure

We are in transition from an old to a new API. Old code use captial letters “GO.*” namespaces. New code uses small letters “go.*”.

The new API can be found in the following folders:

- go/modules/<package>/<module>/views/extjs3/*
- go/core/views/extjs3/*

Extending

When extending components with Ext.extend() always override initComponent and not the constructor:

```javascript
  go.modules.pkg.mymodule.MyComp = Ext.extend(Ext.Panel, {
      initComponent: function() {
          go.modules.pkg.mymodule.MyComp.superclass.initComponent.call(this);

          //Do your initialization here
      }
  });
```

Core overrides are put in views/Extjs3/javascript/overrides.js

30.5.2 Icons

We use Material design icons. Look them up at https://material.io/tools/icons/?style=baseline.
When using icons inline:
When using ExtJS iconCls properties use the “ic-” prefix:

```json
{
  iconCls: "ic-edit"
}
```

### 30.5.3 Entities

Each module can register entities. For example the notes module registers a “Note” entity. This entity will be synchronized with the server.

```javascript
go.Modules.register("community", 'notes', {
    mainPanel: "go.modules.community.notes.MainPanel",
    title: t("Notes"),
    entities: [{
        name: "Note",
        hasFiles: true,
        links: [{
          /**
           * Opens a dialog to create a new linked item
           *
           * @param {string} entity eg. "Note"
           * @param {string|int} entityId
           * @returns {go.form.Dialog}
           */
           linkWindow: function(entity, entityId) {
            return new go.modules.community.notes.NoteForm();
          },
          /**
           * Return component for the detail view
           *
           * @returns {go.detail.Panel}
           */
           linkDetail: function() {
            return new go.modules.community.notes.NoteDetail();
          }
        }]
    }, "NoteBook"],
    systemSettingsPanels: ["go.modules.community.notes.SystemSettingsPanel"]
});
```

An entity can be configured as string for simple entities or as object to provide more options for linking and files.

### 30.5.4 Data stores

We assume you’re already familiar with the data stores ExtJS already has built in.

We’ve extended the Ext.data.JsonStore with our own go.data.Store. We also introduced a new type of store. The go.data.EntityStore.
**go.data.EntityStore**

The EntityStore is a single source of truth for all remote data. You can pass an “entityStore” property to any component or go.data.Store. The component can then implement and onChanged function to handle data changes to the entity.

A simple contact name component example:

```javascript
// A component showing a contact name
var nameComponent = new Ext.BoxComponent({
    // The current contact
    contact: null,

    // Create <h1>John Doe</h1>
    autoEl: "h1",

    // Attach to Contact entity store
    entityStore: "Contact",

    // The onChanged handler
    onChanged: function(entityStore, added, changed, destroyed) {
        // If current contact changed then update the view
        if (this.contact && changed[this.contact.id]) {
            this.setContact(changed);
        }
    },

    // Set contact
    setContact: function(contact) {
        this.contact = contact;

        if (this.rendered) {
            this.update(contact.name);
        }
    }
});
```

**go.data.Store**

This store is used in views and works with an entityStore.

For example:

```javascript
var store = new go.data.Store({
    fields: [
        'id',
        'name',
        'content',
        'excerpt',
        {name: 'createdAt', type: 'date'},
        {name: 'modifiedAt', type: 'date'},
        {name: 'creator', type: 'relation'},
        {name: 'modifier', type: 'relation'},
        {name: 'hasManyRelation', type: 'relation', limit: 5},
        'permissionLevel'
    ],
});
```
Relations

We’ve created a new data field type. Notice the type = ‘relation’. This field will fetch the relation defined in the entity, so you can use this object in your grid renderer for example.

For example:

```
{name: "Artist",
 relations: {
   creator: {store: "User", fk: "createdBy"},
   modifier: {store: "User", fk: "createdBy"},
   albums: {
      genre: {store: "Genre", fk: "genreId"}
   }
}}
```

30.5.5 Custom fields

You don’t have to do anything to make custom fields work in the web client.

The only thing you might want to do is customize the filter of the field sets. With the filter property you can define when a fieldset displays in a dialog. For example when set to:

```
{ "isOrganization": true
}
```

the field set will only show if that property matches the form entity.

You can override the field set dialog by defining the field set dialog class in the entity in Module.js:

```
entities: [{
   name: "Contact",
   customFields: {
      fieldSetDialog: "go.modules.community.addressbook. CustomFieldSetDialog"
   }
}]
```
30.5.6 Account settings

Often it is necessary to create user account settings for modules. You can do that by extending the User entity with properties.

Create the user property on the server before implementing the client.

1. Create a new file SettingsPanel.js:

```javascript
/* global go, Ext */

go.modules.community.addressbook.SettingsPanel = Ext.extend(Ext.Panel, {
    title: t("Address book"),
    iconCls: 'ic-contacts',
    labelWidth: 125,
    layout: "form",
    initComponent: function () {

        // The account dialog is an go.form Dialog that loads the current User as entity.
        this.items = [{
            xtype: "fieldset",
            items: [{
                xtype: "addressbookcombo",
                hiddenName: "addressBookSettings.defaultAddressBookId",
                fieldLabel: t("Default address book")
            }]
        }];

        go.modules.community.addressbook.SettingsPanel.superclass.initComponent.call(this);
    }
});

2. Add this file to scripts.txt

3. Edit Module.js and add the new "userSettingsPanel" to the array:

```javascript
/* global go */
go.Modules.register("community", "addressbook", {
    mainPanel: "go.modules.community.addressbook.MainPanel",
    title: t("Address book"),
    userSettingsPanels: ["go.modules.community.addressbook.SettingsPanel"],
    etc...
});
```

30.5.7 System settings

TODO
30.6 JMAP API

This part explains how you can use the JMAP API from 3rd party software.

30.6.1 API key

First make sure the “API keys” module from the ‘Community’ package is installed at System settings -> modules. Then visit System settings -> API Keys and add a token.

Obtain the token via the more menu and choose “View token”.

30.6.2 Example script

You can find some example PHP scripts to use the JMAP API here:
https://github.com/Intermesh/groupoffice/tree/master/www/go/modules/community/apikeys/examples
You can of course use any programming language with the JMAP API.