1 Įvadas

2 Pradžia
   2.1 Installation on Debian-based systems ......................................................... 3
      2.1.1 Apache HTTP Server ................................................................. 4
      2.1.2 NGINX HTTP Server ................................................................. 6
      2.1.3 Xvfb ................................................................. 11
   2.2 Installation on Windows .................................................................................. 12
   2.3 Serve a project ............................................................................................... 14
   2.4 Configure your project .................................................................................. 15
      2.4.1 WMS capabilities .............................................................................. 17
      2.4.2 WMTS capabilities ........................................................................... 18
      2.4.3 WFS/OAPIF capabilities .................................................................... 20
      2.4.4 WCS capabilities ............................................................................. 21
      2.4.5 Fine tuning your OWS ...................................................................... 22
   2.5 Integration with third parties .......................................................................... 22
      2.5.1 Integration with QGIS Desktop .......................................................... 22
      2.5.2 Integration with MapProxy .................................................................. 22
      2.5.3 Integration with QWC2 ...................................................................... 22

3 Services
   3.1 Basics ............................................................................................................ 23
      3.1.1 SERVICE ....................................................................................... 24
      3.1.2 REQUEST ....................................................................................... 24
      3.1.3 MAP ............................................................................................... 24
      3.1.4 FILE_NAME ................................................................................... 25
      3.1.5 Short name ......................................................................................... 25
   3.2 Žiniatinklio žemėlapio paslauga (WMS) ........................................................... 25
      3.2.1 GetCapabilities ............................................................................... 26
      3.2.2 GetMap ........................................................................................... 26
      3.2.3 GetFeatureInfo ................................................................................. 35
      3.2.4 GetLegendGraphic .......................................................................... 38
      3.2.5 GetStyle(s) ..................................................................................... 51
      3.2.6 DescribeLayer ................................................................................. 51
      3.2.7 GetPrint ........................................................................................... 52
      3.2.8 GetProjectSettings .......................................................................... 54
      3.2.9 GetSchemaExtension ....................................................................... 55
      3.2.10 External WMS layers ....................................................................... 55
      3.2.11 Redlining ......................................................................................... 56
   3.3 Tinklo Geoobjektų Paslauga - Web Feature Service (WFS) ............................. 58
      3.3.1 GetCapabilities ............................................................................... 58
QGIS Server is an open source WMS, WFS, OGC API for Features 1.0 (WFS3) and WCS implementation that, in addition, implements advanced cartographic features for thematic mapping. QGIS Server is a FastCGI/CGI (Common Gateway Interface) application written in C++ that works together with a web server (e.g., Apache, Nginx). It has Python plugin support allowing for fast and efficient development and deployment of new features.

QGIS Server uses QGIS as back end for the GIS logic and for map rendering. Furthermore, the Qt library is used for graphics and for platform-independent C++ programming. In contrast to other WMS software, the QGIS Server uses cartographic rules as a configuration language, both for the server configuration and for the user-defined cartographic rules.

As QGIS desktop and QGIS Server use the same visualization libraries, the maps that are published on the web look the same as in desktop GIS.

In the following sections, we will provide a sample configuration to set up a QGIS Server on Linux (Debian, Ubuntu and derivatives) and on Windows. For more information about server plugin development, please read server_plugins.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section gnu_fdl.
2.1 Installation on Debian-based systems

We will give a short and simple installation how-to for a minimal working configuration on Debian based systems (including Ubuntu and derivatives). However, many other distributions and OSs provide packages for QGIS Server.

Pastaba: In Ubuntu you can use your regular user, prepending `sudo` to commands requiring admin permissions. In Debian you can work as admin (`root`), without using `sudo`.

Requirements and steps to add official QGIS repositories to install QGIS Server on a Debian based system are provided in [QGIS installers page](#). You may want to install at least the latest Long Term Release.

Once the target version repository is configured and QGIS Server installed, you can test the installation with:

```
/usr/lib/cgi-bin/qgis_mapserv.fcgi
```

If you get the following output, the server is correctly installed.

Pastaba: Depending on the version of QGIS, you might see slightly different output reported when you run `qgis_mapserv.fcgi`.

```xml
<ServiceExceptionReport version="1.3.0" xmlns="https://www.opengis.net/ogc">
  <ServiceException code="Service configuration error">Service unknown or unsupported</ServiceException>
</ServiceExceptionReport>
```
Pastaba: As seen below, QGIS reports a Status 400 code, which correctly identifies the request has failed because there is no active http session. This is not a bug and indicates the server is functioning properly.

```
Application path not initialized
Application path not initialized
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver ECW to unload from GDAL_SKIP environment variable.
Warning 1: Unable to find driver JP2ECW to unload from GDAL_SKIP environment variable.
"Loading native module /usr/lib/qgis/server/libdummy.so"
"Loading native module /usr/lib/qgis/server/liblandingpage.so"
"Loading native module /usr/lib/qgis/server/libwcs.so"
"Loading native module /usr/lib/qgis/server/libwfs.so"
"Loading native module /usr/lib/qgis/server/libwfs3.so"
"Loading native module /usr/lib/qgis/server/libwms.so"
"Loading native module /usr/lib/qgis/server/libwmts.so"
QFSFileEngine::open: No file name specified
Content-Length: 102
Content-Type: application/json
Server: QGIS FCGI server - QGIS version 3.22.6-Białowieża
Status: 400
["code":"Bad request error","description":"Requested URI does not match any _registered API handler"]
```

Let's add a sample project. You can use your own, or one from Training demo data:

```
mkdir /home/qgis/projects/
cd /home/qgis/projects/
wget https://github.com/qgis/QGIS-Training-Data/archive/release_3.22.zip
unzip release_3.22.zip
mv QGIS-Training-Data-release_3.22/exercise_data/qgis-server-tutorial-data/world.qgs
mv QGIS-Training-Data-release_3.22/exercise_data/qgis-server-tutorial-data/naturalearth.sqlite.
```

Of course, you can use your favorite GIS software to open this file and take a look at the configuration and available layers.

To properly deploy QGIS server you need a HTTP server. Recommended choices are Apache or NGINX.

### 2.1.1 Apache HTTP Server

Pastaba: In the following, please replace qgis.demo with the name or IP address of your server.

1. Install Apache and `mod_fcgid`:

   ```bash
   apt install apache2 libapache2-mod-fcgid
   ```

2. You can run QGIS Server on your default website, but let's configure a `virtualhost` specifically for this, as follows.
   1. In the `/etc/apache2/sites-available` directory, create a file called `qgis.demo.conf`, with this content:

   ```html
   <VirtualHost *:80>
   ServerAdmin webmaster@localhost
   ServerName qgis.demo
   ```
   (continues on next page)
Pastaba: Some of the above configuration options are explained in the Server environment variables and pg_service file sections.

2. Let’s now create the directories that will store the QGIS Server logs and the authentication database:

```
mkdir -p /var/log/qgis/
chown www-data:www-data /var/log/qgis
mkdir -p /home/qgis/qgisserverdb
chown www-data:www-data /home/qgis/qgisserverdb
```
Pastaba: *www-data* is the Apache user on Debian based systems and we need Apache to have access to those locations or files. The `chown www-data...` commands change the owner of the respective directories and files to *www-data*.

3. We can now enable the virtual host and the `fcgid` mod if it's not already done:

   ```
   a2enmod fcgid
   a2ensite qgis.demo
   ```

4. Now restart Apache for the new configuration to be taken into account:

   ```
   systemctl restart apache2
   ```

5. Now that Apache knows that he should answer requests to `http://qgis.demo` we also need to setup the client system so that it knows who `qgis.demo` is. We do that by adding `127.0.0.1 qgis.demo` in the `hosts` file.

   ```
   # Replace 127.0.0.1 with the IP of your server.
   sh -c "echo '127.0.0.1 qgis.demo' >> /etc/hosts"
   ```

Svarbu: Remember that both the `qgis.demo.conf` and `/etc/hosts` files should be configured for your setup to work. You can also test the access to your QGIS Server from other clients on the network (e.g. Windows or macos machines) by going to their `/etc/hosts` file and point the `qgis.demo` name to whatever IP the server machine has on the network (not `127.0.0.1` as it is the local IP, only accessible from the local machine). On *nix machines the `hosts` file is located in `/etc`, while on Windows it's under the C:\Windows\System32\drivers\etc directory. Under Windows you need to start your text editor with administrator privileges before opening the hosts file.

QGIS Server is now available at `http://qgis.demo`. To check, type in a browser, as in the simple case:

```
http://qgis.demo/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
```

### 2.1.2 NGINX HTTP Server

Pastaba: In the following, please replace `qgis.demo` with the name or IP address of your server.

You can also use QGIS Server with **NGINX**. Unlike Apache, NGINX does not automatically spawn FastCGI processes. The FastCGI processes are to be started by something else.

Install NGINX:

```
apt install nginx
```

- As a first option, you can use `spawn-fcgi` or `fcgiwrap` to start and manage the QGIS Server processes. Official Debian packages exist for both. When you have no X server running and you need, for example, printing, you can use `xvfb`.

- Another option is to rely on **Systemd**, the init system for GNU/Linux that most Linux distributions use today. One of the advantages of this method is that it requires no other components or processes. It’s meant to be simple, yet robust and efficient for production deployments.
NGINX Configuration

The include fastcgi_params; used in the previous configuration is important, as it adds the parameters from /etc/nginx/fastcgi_params:

```plaintext
fastcgi_param QUERY_STRING $query_string;
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;
fastcgi_param SCRIPT_NAME $fastcgi_script_name;
fastcgi_param REQUEST_URI $request_uri;
fastcgi_param DOCUMENT_URI $document_uri;
fastcgi_param SERVER_PROTOCOL $server_protocol;
fastcgi_param REQUEST_SCHEME $scheme;
fastcgi_param GATEWAY_INTERFACE CGI/1.1;
fastcgi_param SERVER_SOFTWARE nginx/$nginx_version;
fastcgi_param REMOTE_ADDR $remote_addr;
fastcgi_param REMOTE_PORT $remote_port;
fastcgi_param SERVER_ADDR $server_addr;
fastcgi_param SERVER_PORT $server_port;
fastcgi_param SERVER_NAME $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param REDIRECT_STATUS 200;
```

Moreover, you can use some Environment variables to configure QGIS Server. In the NGINX configuration file, /etc/nginx/nginx.conf, you have to use fastcgi_param instruction to define these variables as shown below:

```plaintext
location /qgisserver {
    gzip off;
    include fastcgi_params;
    fastcgi_param QGIS_SERVER_LOG_STDERR 1;
    fastcgi_param QGIS_SERVER_LOG_LEVEL 0;
    fastcgi_pass unix:/var/run/qgisserver.socket;
}
```

FastCGI wrappers

Išpėjimas: fcgiwrap is easier to set up than spawn-fcgi, because it’s already wrapped in a Systemd service. But it also leads to a solution that is much slower than using spawn-fcgi. With fcgiwrap, a new QGIS Server process is created on each request, meaning that the QGIS Server initialization process, which includes reading and parsing the QGIS project file, is done on each request. With spawn-fcgi, the QGIS Server process remains alive between requests, resulting in much better performance. For that reason, spawn-fcgi is recommended for production use.

spawn-fcgi

If you want to use spawn-fcgi:

1. The first step is to install the package:

   ```
   apt install spawn-fcgi
   ```

2.1. Installation on Debian-based systems
2. Then, introduce the following block in your NGINX server configuration:

```
location /qgisserver {
    gzip off;
    include fastcgi_params;
    fastcgi_pass unix:/var/run/qgisserver.socket;
}
```

3. And restart NGINX to take into account the new configuration:

```bash
systemctl restart nginx
```

4. Finally, considering that there is no default service file for spawn-fcgi, you have to manually start QGIS Server in your terminal:

```bash
spawn-fcgi -s /var/run/qgisserver.socket \        -U www-data -G www-data -n \          /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

QGIS Server is now available at http://qgis.demo/qgisserver.

**Pastaba:** When using spawn-fcgi, you may directly define environment variables before running the server. For example: `export QGIS_SERVER_LOG_STDERR=1`

Of course, you can add an init script to start QGIS Server at boot time or whenever you want. For example with `systemd`:

1. Edit the file `/etc/systemd/system/qgis-server.service` with this content:

```
[Unit]
Description=QGIS server
After=network.target

[Service]
;; set env var as needed
;Environment="LANG=en_EN.UTF-8"
;Environment="QGIS_SERVER_PARALLEL_RENDERING=1"
;Environment="QGIS_SERVER_MAX_THREADS=12"
;Environment="QGIS_SERVER_LOG_LEVEL=0"
;Environment="QGIS_SERVER_LOG_STDERR=1"
;; or use a file:
;EnvironmentFile=/etc/qgis-server/env


[Install]
WantedBy=multi-user.target
```

2. Then enable and start the service:

```bash
systemctl enable --now qgis-server
```

**Įspėjimas:** With the above commands spawn-fcgi spawns only one QGIS Server process.
fcgiwrap

Using fcgiwrap is much easier to setup than spawn-fcgi but it’s much slower.

1. You first have to install the corresponding package:

```
apt install fcgiwrap
```

2. Then, introduce the following block in your NGINX server configuration:

```
location /qgisserver {
    gzip off;
    include fastcgi_params;
    fastcgi_pass unix:/var/run/fcgiwrap.socket;
    fastcgi_param SCRIPT_FILENAME /usr/lib/cgi-bin/qgis_mapserv.fcgi;
}
```

3. Finally, restart NGINX and fcgiwrap to take into account the new configuration:

```
systemctl restart nginx
systemctl restart fcgiwrap
```

QGIS Server is now available at http://qgis.demo/qgisserver.

Systemd

QGIS Server needs a running X Server to be fully usable, in particular for printing. In the case you already have a running X Server, you can use systemd services.

This method, to deploy QGIS Server, relies on two Systemd units to configure: a Socket unit and a Service unit.

1. The QGIS Server Socket unit defines and creates a file system socket, used by NGINX to start and communicate with QGIS Server. The Socket unit has to be configured with `Accept=false`, meaning that the calls to the `accept()` system call are delegated to the process created by the Service unit. It is located in `/etc/systemd/system/qgis-server@.socket`, which is actually a template:

```
[Unit]
Description=QGIS Server Listen Socket (instance %i)

[Socket]
Accept=false
ListenStream=/var/run/qgis-server-%i.sock
SocketUser=www-data
SocketGroup=www-data
SocketMode=0600

[Install]
WantedBy=sockets.target
```

2. Now enable and start sockets:

```
for i in 1 2 3 4; do systemctl enable --now qgis-server@$i.socket; done
```

3. The QGIS Server Service unit defines and starts the QGIS Server process. The important part is that the Service process’ standard input is connected to the socket defined by the Socket unit. This has to be configured using `StandardInput=socket` in the Service unit configuration located in `/etc/systemd/system/qgis-server@.service`:

```
[Unit]
Description=QGIS Server Service (instance %i)
```

(continues on next page)
Pastaba: The QGIS Server environment variables are defined in a separate file, /etc/qgis-server/env. It could look like this:

```
QGIS_PROJECT_FILE=/etc/qgis/myproject.qgs
QGIS_SERVER_LOG_STDERR=1
QGIS_SERVER_LOG_LEVEL=3
```

4. Now start socket service:

```
for i in 1 2 3 4; do systemctl enable --now qgis-server@$i.service; done
```

5. Finally, for the NGINX HTTP server, let's introduce the configuration for this setup:

```
upstream qgis-server_backend {
    server unix:/var/run/qgis-server-1.sock;
    server unix:/var/run/qgis-server-2.sock;
    server unix:/var/run/qgis-server-3.sock;
    server unix:/var/run/qgis-server-4.sock;
}

server {
    
    location /qgis-server {
        gzip off;
        include fastcgi_params;
        fastcgi_pass qgis-server_backend;
    }
}
```

6. Now restart NGINX for the new configuration to be taken into account:

```
systemctl restart nginx
```

Thanks to Oslandia for sharing their tutorial.
2.1.3 Xvfb

QGIS Server needs a running X Server to be fully usable, in particular for printing. On servers it is usually recommended not to install it, so you may use xvfb to have a virtual X environment.

If you’re running the Server in graphic/X11 environment then there is no need to install xvfb. More info at https://www.itopen.it/qgis-server-setup-notes/.

1. To install the package:

   ```
   apt install xvfb
   ```

2. Create the service file, `/etc/systemd/system/xvfb.service`, with this content:

   ```
   [Unit]
   Description=X Virtual Frame Buffer Service
   After=network.target
   [Service]
   ExecStart=/usr/bin/Xvfb :99 -screen 0 1024x768x24 -ac +extension GLX +render - -noreset
   [Install]
   WantedBy=multi-user.target
   ```

3. Enable, start and check the status of the xvfb.service:

   ```
   systemctl enable --now xvfb.service
   systemctl status xvfb.service
   ```

4. Then, according to your HTTP server, you should configure the DISPLAY parameter or directly use xvfb-run.

   - Using Apache:
     1. Add to your Fcgid configuration (see Apache HTTP Server):

        ```
        FcgidInitialEnv DISPLAY ":99"
        ```

     2. Restart Apache for the new configuration to be taken into account:

        ```
        systemctl restart apache2
        ```

   - Using NGINX
     - With spawn-fcgi using xvfb-run:

        ```
        xvfb-run /usr/bin/spawn-fcgi -f /usr/lib/cgi-bin/qgis_mapserv.fcgi \
        -s /tmp/qgissserver.socket \
        -G www-data -U www-data -n
        ```

     - With the DISPLAY environment variable in the HTTP server configuration.

        ```
        fastcgi_param DISPLAY ":99";
        ```
2.2 Installation on Windows

QGIS Server can also be installed on Windows systems using the 64 bit version of the OSGeo4W network installer (https://qgis.org/en/site/forusers/download.html).

A simple procedure is the following:

1. Download and run the OSGeo4W installer
2. Follow the „Advanced Install“ and install the QGIS Desktop, QGIS Server apache and mod_fcgid packages.

3. Apache is not directly installed as a service on Windows. You need to:
   1. Right-click the OSGeo4W.bat file at the root of the C:\OSGeo4W64\ folder (if the default installation paths have been used) and select Run as administrator
   2. In the console, run apache-install.bat, which will output

   ```
   > apache-install.bat
   Installing the 'Apache OSGeo4W Web Server' service
   The 'Apache OSGeo4W Web Server' service is successfully installed.
   Testing httpd.conf....
   Errors reported here must be corrected before the service can be started.
   ...
   ```

   The service is started as you can notice in the report. But the server may fail to run due to missing custom configuration.

4. Edit the C:\OSGeo4w64\apps\apache\conf\httpd.conf file with the following changes (various other combinations are possible):
## Purpose

(Optional) Customize the address to listen to using an IP and/or port. You can and add as many entries as you wish.

Indicate where to find the script files.

Provide permissions on the script folder.

Enable file extensions to use for script files. Uncomment and complete.

Add more OSGeo4W custom configuration variables.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Existing config</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Optional) Customize the address to listen to using an IP and/or port. You can and add as many entries as you wish.</td>
<td>Listen ${SRVPORT}</td>
<td>Listen localhost:8080</td>
</tr>
<tr>
<td>Indicate where to find the script files</td>
<td>ScriptAlias /cgi-bin/ &quot;$\rightarrow$ {SRVROOT}/cgi-bin/&quot;</td>
<td>ScriptAlias /cgi-bin/ &quot;C:/OSGeo4W64/apps/qgis/bin/&quot;</td>
</tr>
<tr>
<td>Provide permissions on the script folder</td>
<td>&lt;Directory &quot;${SRVROOT}/cgi-bin&quot;&gt; AllowOverride None Options None Require all granted &lt;/Directory&gt;</td>
<td>&lt;Directory &quot;C:/OSGeo4W64/apps/qgis/bin&quot;&gt; SetHandler cgi-script AllowOverride None Options ExecCGI Require all granted &lt;/Directory&gt;</td>
</tr>
<tr>
<td>Enable file extensions to use for script files. Uncomment and complete</td>
<td>#AddHandler cgi-script .cgi</td>
<td>AddHandler cgi-script .cgi .exe</td>
</tr>
<tr>
<td>Add more OSGeo4W custom configuration variables</td>
<td># parse OSGeo4W apache conf files IncludeOptional &quot;C:/OSGeo4W64/httpd.d/ httpd_* .conf&quot;</td>
<td># parse OSGeo4W apache conf files IncludeOptional &quot;C:/OSGeo4W64/httpd.d/ httpd_* .conf&quot; SetEnv GDAL_DATA &quot;C:/OSGeo4W64/share/gdal&quot; SetEnv QGIS_AUTH_DB_DIR_ PATH &quot;C:/OSGeo4W64/apps/qgis/resources&quot;</td>
</tr>
</tbody>
</table>

5. Restart the Apache web server

```bash
> apache-restart.bat
```

6. Open browser window to testing a GetCapabilities request to QGIS Server. Replace localhost:8080 with the IP and port you set to listen.

```bash
http://localhost:8080/cgi-bin/qgis_mapserv.fcgi.exe?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
```

A XML file with the capabilities should be returned. Your server is ready to use.

### 2.2. Installation on Windows
2.3 Serve a project

Now that QGIS Server is installed and running, we just have to use it.

Obviously, we need a QGIS project to work on. Of course, you can fully customize your project by defining contact information, precise some restrictions on CRS or even exclude some layers. Everything you need to know about that is described later in Configure your project.

But for now, we are going to use a simple project already configured and previously downloaded in /home/qgis/projects/world.qgs, as described above.

By opening the project and taking a quick look on layers, we know that 4 layers are currently available:

- airports
- places
- countries
- countries_shapeburst

You don’t have to understand the full request for now but you may retrieve a map with some of the previous layers thanks to QGIS Server by doing something like this in your web browser to retrieve the countries layer:

- If you followed the above instructions to install an Apache HTTP Server:

```plaintext
http://qgis.demo/cgi-bin/qgis_mapserv.fcgi?
  MAP=/home/qgis/projects/world.qgs&
  LAYERS=_countries&
  SERVICE=WMS&
  VERSION=1.3.0&
  REQUEST=GetMap&
  CRS=EPSG:4326&
  WIDTH=400&
  HEIGHT=200&
  BBOX=-90,-180,90,180
```

- If you followed the above instructions to install an NGINX HTTP Server:

```plaintext
http://qgis.demo/qgisserver?
  MAP=/home/qgis/projects/world.qgs&
  LAYERS=_countries&
  SERVICE=WMS&
  VERSION=1.3.0&
  REQUEST=GetMap&
  CRS=EPSG:4326&
  WIDTH=400&
  HEIGHT=200&
  BBOX=-90,-180,90,180
```

If you obtain the next image, then QGIS Server is running correctly:
Note that you may define `QGIS_PROJECT_FILE` environment variable to use a project by default instead of giving a `MAP` parameter (see Environment variables).

For example with spawn-fcgi:

```
export QGIS_PROJECT_FILE=/home/qgis/projects/world.qgs
spawn-fcgi -f /usr/lib/bin/cgi-bin/qgis_mapserv.fcgi \
    -s /var/run/qgisserver.socket \
    -U www-data -G www-data -n
```

### 2.4 Configure your project

To provide a new QGIS Server WMS, WFS, OAPIF or WCS, you have to create a QGIS project file with some data or use one of your current project. Define the colors and styles of the layers in QGIS and the project CRS, if not already defined. Then, go to the QGIS Server menu of the Project ➤ Properties… dialog and provide some information about the OWS in the Service Capabilities tab.
2.2 Fig.: Definitions for a QGIS Server WMS/WFS/WCS project

You have to Enable Service Capabilities first, if it is deactivated. This will appear in the GetCapabilities response of the WMS, WFS or WCS. If you don’t check Enable Service capabilities, QGIS Server will use the information given in the wms_metadata.xml file located in the cgi-bin folder.
2.4.1 WMS capabilities

In the **WMS capabilities** tab, you can define the extent advertised in the WMS GetCapabilities response by entering the minimum and maximum X and Y values in the fields under *Advertised extent*.

![WMS capabilities tab](image)

2.3 Fig.: Definitions in the WMS tab

Clicking *Use Current Canvas Extent* sets these values to the extent currently displayed in the QGIS map canvas. By checking *CRS restrictions*, you can restrict in which coordinate reference systems (CRS) QGIS Server will offer to render maps. It is recommended that you restrict the offered CRS as this reduces the size of the WMS GetCapabilities...
response. Use the button below to select those CRSs from the Coordinate Reference System Selector, or click Used to add the CRSs used in the QGIS project to the list.

If you have print layouts defined in your project, they will be listed in the GetProjectSettings response, and they can be used by the GetPrint request to create prints, using one of the print layouts as a template. This is a QGIS-specific extension to the WMS 1.3.0 specification. If you want to exclude any print layout from being published by the WMS, check Exclude layouts and click the button below. Then, select a print layout from the Select print layout dialog in order to add it to the excluded layouts list.

If you want to exclude any layer or layer group from being published by the WMS, check Exclude Layers and click the button below. This opens the Select restricted layers and groups dialog, which allows you to choose the layers and groups that you don’t want to be published. Use the Shift or Ctrl key if you want to select multiple entries. It is recommended that you exclude from publishing the layers that you don’t need as this reduces the size of the WMS GetCapabilities response which leads to faster loading times on the client side.

If you check Use layer ids as name, layer ids will be used to reference layers in the GetCapabilities response or GetMap LAYERS parameter. If not, layer name or short name if defined (see vectorservermenu) is used.

You can receive requested GetFeatureInfo as plain text, XML and GML. The default is XML.

If you wish, you can check Add geometry to feature response. This will include the bounding box for each feature in the GetFeatureInfo response. See also the WITH_GEOMETRY parameter.

As many web clients can’t display circular arcs in geometries you have the option to segmentize the geometry before sending it to the client in a GetFeatureInfo response. This allows such clients to still display a feature’s geometry (e.g. for highlighting the feature). You need to check the Segmentize feature info geometry to activate the option.

You can also use the GetFeatureInfo geometry precision option to set the precision of the GetFeatureInfo geometry. This enables you to save bandwidth when you don’t need the full precision.

If you want QGIS Server to advertise specific request URLs in the WMS GetCapabilities response, enter the corresponding URL in the Advertised URL field.

Furthermore, you can restrict the maximum size of the maps returned by the GetMap request by entering the maximum width and height into the respective fields under Maximums for GetMap request.

You can change the Quality for JPEG images factor. The quality factor must be in the range 0 to 100. Specify 0 for maximum compression, 100 for no compression.

You can change the limit for atlas features to be printed in one request by setting the Maximum features for Atlas print requests field.

When QGIS Server is used in tiled mode (see TILED parameter), you can set the Tile buffer in pixels. The recommended value is the size of the largest symbol or line width in your QGIS project.

If one of your layers uses the Map Tip display (i.e. to show text using expressions) this will be listed inside the GetFeatureInfo output. If the layer uses a Value Map for one of its attributes, this information will also be shown in the GetFeatureInfo output.

### 2.4.2 WMTS capabilities

In the WMTS capabilities tab you can select the layers you want to publish as WMTS and specify if you want to publish as PNG or JPEG.
If you enter a URL in the **Advertised URL** field of the **WMTS capabilities** section, QGIS Server will advertise this specific URL in the WMTS GetCapabilities response.
2.4.3 WFS/OAPIF capabilities

In the WFS/OAPIF capabilities tab, you can select the layers you want to publish as WFS or OAPIF, and specify if they will allow update, insert and delete operations.

2.5 Fig.: Definitions in the WFS/OAPIF tab

If you enter a URL in the Advertised URL field of the WFS capabilities section, QGIS Server will advertise this specific URL in the WFS GetCapabilities response.
2.4.4 WCS capabilities

In the WCS capabilities tab, you can select the layers that you want to publish as WCS.

2.6 Fig.: Definitions in the WCS tab

If you enter a URL in the Advertised URL field of the WCS capabilities section, QGIS Server will advertise this specific URL in the WCS GetCapabilities response.
2.4.5 Fine tuning your OWS

For vector layers, the *Fields* menu of the *Layer ▶ Layer Properties* dialog allows you to define for each attribute if it will be published or not. By default, all the attributes are published by your WMS and WFS. If you don’t want a specific attribute to be published, uncheck the corresponding checkbox in the *WMS* or *WFS* column.

You can overlay watermarks over the maps produced by your WMS by adding text annotations or SVG annotations to the project file. See the sec.Annotations section for instructions on creating annotations. For annotations to be displayed as watermarks on the WMS output, the *Fixed map position* checkbox in the *Annotation text* dialog must be unchecked. This can be accessed by double clicking the annotation while one of the annotation tools is active. For SVG annotations, you will need either to set the project to save absolute paths (in the *General* menu of the *Project ▶ Properties* dialog) or to manually modify the path to the SVG image so that it represents a valid relative path.

2.5 Integration with third parties

QGIS Server provides standard OGC web services like WMS, WFS, etc. thus it can be used by a wide variety of end user tools.

2.5.1 Integration with QGIS Desktop

QGIS Desktop is the map designer where QGIS Server is the map server. The maps or QGIS projects will be served by the QGIS Server to provide OGC standards. These QGIS projects can either be files or entries in a database (by using *Project ▶ Save to ▶ PostgreSQL* in QGIS Desktop).

Furthermore, dedicated update workflow must be established to refresh a project used by a QGIS Server (ie. copy project files into server location and restart QGIS Server). For now, automated processes (as server reloading over message queue service) are not implemented yet.

2.5.2 Integration with MapProxy

MapProxy is a tile cache server and as it can read and serve any WMS/WMTS map server, it can be directly connected to QGIS server web services and improve end user experience.

2.5.3 Integration with QWC2

QWC2 is a responsive web application dedicated to QGIS Server. It helps you to build a highly customized map viewer with layer selection, feature info, etc.. Also many plugins are available like authentication or print service, the full list is available in this repository.
QGIS Server is able to serve data according to standard protocols as described by the Open Geospatial Consortium (OGC):

- WMS 1.1.1 and 1.3.0
- WFS 1.0.0 and 1.1.0
- OGC API - Features (WFS3)
- WCS 1.0.0 and 1.1.1
- WMTS 1.0.0

Extra vendor parameters and requests are supported in addition to the original standard that greatly enhance the possibilities of customizing its behavior thanks to the QGIS rendering engine.

### 3.1 Basics

This section describes concepts and parameters mutually shared by services. Some of these are standard and defined in OGC specifications while others are very specific to QGIS Server.

**Standard concepts:**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Name of the service</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Name of the request</td>
</tr>
</tbody>
</table>

**Vendor concepts:**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>File name of the downloaded file</td>
</tr>
<tr>
<td>Short name</td>
<td>Short name definition</td>
</tr>
</tbody>
</table>
3.1.1 SERVICE

This standard parameter allows to specify the name of the service to use for a specific request and has to be formed like `SERVICE=NAME`.

URL example for the WMS service:

```
http://localhost/qgisserver?
SERVICE=WMS
&...
```

Pastaba: Not available for REST based services like WFS3 (OGC API Features).

3.1.2 REQUEST

This standard parameter allows to specify the name of the request to execute for a specific service and has to be formed like `REQUEST=RequestName`.

URL example for the GetCapabilities request:

```
http://localhost/qgisserver?
REQUEST=GetCapabilities
&...
```

Pastaba: Not available for REST based services like WFS3 (OGC API Features).

3.1.3 MAP

This vendor parameter allows to define the QGIS project file to use. It may be an absolute path or a path relative to the location of the server executable `qgis_mapser fcgi`. MAP is mandatory by default because a request needs a QGIS project to actually work. However, the `QGIS_PROJECT_FILE` environment variable may be used to define a default QGIS project. In this specific case, MAP is no longer a required parameter. For further information you may refer to the Advanced configuration chapter.

URL example:

```
http://localhost/qgisserver?
MAP=/tmp/QGIS-Training-Data/exercise_data/qgis-server-tutorial-data/world.qgs
&...
```

3.1.4 FILE_NAME

If this vendor parameter is set, the server response will be sent to the client as a file attachment with the specified file name.

URL example to save an XML GetCapabilities document:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetCapabilities
&FILE_NAME=wms_capabilities.xml
&...
```
3.1.5 Short name

A number of elements have both a **short name** and a **title**. The short name is a text string used for machine-to-machine communication while the title is for the benefit of humans. For example, a dataset might have the descriptive title "Maximum Atmospheric Temperature" and be requested using the abbreviated short name "ATMAX". You can set title, short name and abstract for:

- **Layers**: right-click on a layer and choose *Properties… ➤ QGIS Server ➤ Description*.
- **Groups**: right-click on a group and select *Set Group WMS data*
- **Project**: go to *Project ➤ Properties… ➤ QGIS Server ➤ Service Capabilities*.

Thus, the short name may be used to identify these items when interacting with QGIS Server. For example with the standard **LAYERS** parameter:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=shortname1,shortname2
&...
```

3.2 Žiniatinklio žemėlapio paslauga (WMS)

QGIS Serveris palaiko **1.1.1** ir **1.3.0** WMS standartus, kurie teikia HTTP sąsają užklausoms žemėlapiai ar legendos piešiniams, sukurtiems pagal QGIS projektą. Tipinė WMS užklausa nurodo, kokį reikia naudoti QGIS projektą, kokius braižyti sluoksnius bei kokiu formatu pateikti piešinį. Taipogi yra bazinis **Stilizuoto sluoksnio aprašymo (SLD)** palaikymas.

Specifikacijos

- WMS 1.1.1
- WMS 1.3.0
- SLD 1.1.0 WMS profile

Standartinės QGIS Serverio teikiamos užklausos:
### Request

<table>
<thead>
<tr>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>GetMap</td>
<td>Grąžina žemėlapį</td>
</tr>
<tr>
<td>GetFeatureInfo</td>
<td>Grąžina duomenis (geometriją ir reikšmes) pikelio vietai</td>
</tr>
<tr>
<td>GetLegendGraphic</td>
<td>Grąžina legendos simbolius</td>
</tr>
<tr>
<td>GetStyle(s)</td>
<td>Grąžina XML dokumentą su SLD stiliais aprašymu</td>
</tr>
<tr>
<td>DescribeLayer</td>
<td>Grąžina informaciją apie WFS ir WCS prieinamumą atitinkamai vektoriniam ir rastro sluoksniamų</td>
</tr>
</tbody>
</table>

Tiekėjo užklausos, kurias teikia QGIS Serveris:

<table>
<thead>
<tr>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPrint</td>
<td>Grąžina QGIS išdėstymą</td>
</tr>
<tr>
<td>GetProjectSettings</td>
<td>Grąžina specifinę informaciją apie QGIS Serverį</td>
</tr>
<tr>
<td>GetSchemaExtension</td>
<td>Grąžina XML metaduomenis apie neprivalomas išplėstines galimybes</td>
</tr>
</tbody>
</table>

#### 3.2.1 GetCapabilities

Standartiniai GetCapabilities užklausos parametrai pagal OGC WMS 1.1.1 ir 1.3.0 specifikacijas:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugos pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetCapabilities)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
</tbody>
</table>

The GetCapabilities request supports as well the following vendor parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetCapabilities
```

#### 3.2.2 GetMap

Standard parameters for the GetMap request according to the OGC WMS 1.1.1 and 1.3.0 specifications:
In addition to the standard ones, QGIS Server supports redlining, external WMS layers as well as the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>BGCOLOR</td>
<td>No</td>
<td>Specify the background color</td>
</tr>
<tr>
<td>DPI</td>
<td>No</td>
<td>Specify the output resolution</td>
</tr>
<tr>
<td>IMAGE_QUALITY</td>
<td>No</td>
<td>JPEG compression</td>
</tr>
<tr>
<td>OPACITIES</td>
<td>No</td>
<td>Opacity for layer or group</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>Subset of features</td>
</tr>
<tr>
<td>SELECTION</td>
<td>No</td>
<td>Highlight features</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>No</td>
<td>File name of the downloaded file</td>
</tr>
<tr>
<td>FORMAT_OPTIONS</td>
<td>No</td>
<td>Options of the specified file format</td>
</tr>
<tr>
<td>TILED</td>
<td>No</td>
<td>Working in tiled mode</td>
</tr>
</tbody>
</table>

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&VERSION=1.3.0
&REQUEST=GetMap
&MAP=/home/qgis/projects/world.qgs
&LAYERS=mylayer1,mylayer2,mylayer3
&STYLES=style1,default,style3
&OPACITIES=125,200,125
&CRS=EPSG:4326
&WIDTH=400
&HEIGHT=400
&FORMAT=image/png
&TRANSPARENT=TRUE
&DPI=300
&TILED=TRUE
```
VERSION

This parameter allows to specify the version of the service to use. Available values for the VERSION parameter are:

- 1.1.1
- 1.3.0

According to the version number, slight differences have to be expected as explained later for the next parameters:

- CRS / SRS
- BBOX

LAYERS

This parameter allows to specify the layers to display on the map. Names have to be separated by a comma.

In addition, QGIS Server introduced some options to select layers by:

- a short name
- the layer id

The project option allowing to select layers by their id is in QGIS Server ► WMS tab of the Project ► Properties… dialog. Check the Use layer ids as names checkbox to activate this option.

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayerid1,mylayerid2
&...
```

STYLES

This parameter can be used to specify a layer's style for the rendering step. Styles have to be separated by a comma. The name of the default style is default.

SRS / CRS

This parameter allows to indicate the map output Spatial Reference System in WMS 1.1.1 and has to be formed like EPSG:XXXX. Note that CRS is also supported if current version is 1.1.1.

For WMS 1.3.0, CRS parameter is preferable but SRS is also supported.

Note that if both CRS and SRS parameters are indicated in the request, then it’s the current version indicated in VERSION parameter which is decisive.

In the next case, the SRS parameter is kept whatever the VERSION parameter because CRS is not indicated:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&SRS=EPSG:2854
&...
```

In the next case, the SRS parameter is kept instead of CRS because of the VERSION parameter:
In the next case, the CRS parameter is kept instead of SRS because of the VERSION parameter:

```http
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=EPSG:4326
&SRS=EPSG:2854
&...
```

**BBOX**

This parameter allows to specify the map extent with units according to the current CRS. Coordinates have to be separated by a comma.

The BBOX parameter is formed like `min_a,min_b,max_a,max_b` but a and b axis definition is different according to the current VERSION parameter:

- in WMS 1.1.1, the axis ordering is always east/north
- in WMS 1.3.0, the axis ordering depends on the CRS authority

For example in case of EPSG:4326 and WMS 1.1.1, a is the longitude (east) and b the latitude (north), leading to a request like:

```http
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.1.1
&SRS=epsg:4326
&BBOX=-180,-90,180,90
&...
```

But in case of WMS 1.3.0, the axis ordering defined in the EPSG database is north/east so a is the latitude and b the longitude:

```http
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&CRS=epsg:4326
&BBOX=-90,-180,90,180
&...
```
**WIDTH**

This parameter allows to specify the width in pixels of the output image.

**HEIGHT**

This parameter allows to specify the height in pixels of the output image.

**FORMAT**

This parameter may be used to specify the format of map image. Available values are:

- `jpg`
- `jpeg`
- `image/jpeg`
- `image/png`
- `image/png; mode=1bit`
- `image/png; mode=8bit`
- `image/png; mode=16bit`
- `image/webp`
- `application/dxf`: only layers that have read access in the WFS service are exported in the DXF format

**URL example:**

```
http://localhost/qgisserver?
SERVICE=WMS&VERSION=1.3.0
&REQUEST=GetMap
&FORMAT=application/dxf
&LAYERS=Haltungen,Normschacht,Spezialbauwerke
&CRS=EPSG%3A21781
&BBOX=696136.28844801,245797.12108743,696318.91114315,245939.25832905
&WIDTH=1042
&HEIGHT=811
&FORMAT_OPTIONS=MODE:SYMBOLSYMBOLSYMBOLY;SCALE:250
&FILE_NAME=plan.dxf
```

**TRANSPARENT**

This boolean parameter can be used to specify the background transparency. Available values are (not case sensitive):

- `TRUE`
- `FALSE`

However, this parameter is ignored if the format of the image indicated with `FORMAT` is different from PNG.
**BGCOLOR**

This parameter allows to indicate a background color for the map image. However it cannot be combined with TRANSPARENT parameter in case of PNG images (transparency takes priority). The colour may be literal or in hexadecimal notation.

**URL example with the literal notation:**

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&BGCOLOR=green
&...
```

**URL example with the hexadecimal notation:**

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&BGCOLOR=0x00FF00
&...
```

**DPI**

This parameter can be used to specify the requested output resolution.

**IMAGE_QUALITY**

This parameter is only used for JPEG images. By default, the JPEG compression is -1.

You can change the default per QGIS project in the OWS Server ➤ WMS capabilities menu of the Project ➤ Properties… dialog. If you want to override it in a GetMap request you can do it using the IMAGE_QUALITY parameter.

**OPACITIES**

Comma separated list of opacity values. Opacity can be set on layer or group level. Allowed values range from 0 (fully transparent) to 255 (fully opaque).

**URL example:**

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&WIDTH=400
&HEIGHT=200
&CRS=EPSG:4326
&LAYERS=countries,places
&BBOX=42,-6,52,15
&OPACITIES=255,0
```
FILTER

A subset of layers can be selected with the FILTER parameter. The syntax is basically the same as for the QGIS subset string. However, there are some restrictions to avoid SQL injections into databases via QGIS Server. If a dangerous string is found in the parameter, QGIS Server will return the next error:

```
<ServiceExceptionReport>
  <ServiceException code="Security">The filter string XXXXXXXXX has been rejected because of security reasons.
  Note: Text strings have to be enclosed in single or double quotes. A space between each word / special character is mandatory.
  Allowed Keywords and special characters are IS,NOT,NULL,AND,OR,IN,=,<,<>,>=,!=, '-', ',', (), DMETAPHONE, SOUNDEX.
  Not allowed are semicolons in the filter expression.</ServiceException>
</ServiceExceptionReport>
```

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&VERSION=1.3.0
&WIDTH=400
&HEIGHT=300
&CRS=EPSG:4326
&BBOX=41,-6,52,10
&LAYERS=countries_shapeburst,countries,places
&FILTER=countries_shapeburst,countries:"name" = 'France';places: "name" = 'Paris'
```
In this example, the same filter "name" = 'France' is applied to layers countries and countries_shapeburst, while the filter "name" = 'Paris' is only applied to places.

Pastaba: It is possible to make attribute searches via GetFeatureInfo and omit the X/Y parameter if a FILTER is there. QGIS Server then returns info about the matching features and generates a combined bounding box in the XML output.

**SELECTION**

The **SELECTION** parameter can highlight features from one or more layers. Vector features can be selected by passing comma separated lists with feature ids.

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=mylayer1,mylayer2
&SELECTION=mylayer1:3,6,9;mylayer2:1,5,6
&...
```

The following image presents the response from a GetMap request using the **SELECTION** option e.g. http://myserver.com/...&SELECTION=countries:171,65.

As those features id's correspond in the source dataset to France and Romania they’re highlighted in yellow.
FORMAT_OPTIONS

This parameter can be used to specify options for the selected format. Only for FORMAT=application/dxf. A list of key:value pairs separated by semicolon:

- SCALE: to be used for symbology rules, filters and styles (not actual scaling of the data - data remains in the original scale).
- MODE: corresponds to the export options offered in the QGIS Desktop DXF export dialog. Possible values are NOSYMOLOGY, FEATURESYMOLOGY and SYMBOLLAYERSYMBOLOGY.
- LAYERSATTRIBUTES: specify a field that contains values for DXF layer names - if not specified, the original QGIS layer names are used.
- USE_TITLE_AS_LAYERNAME: if enabled, the title of the layer will be used as layer name.
- CODEC: specify a codec to be used for encoding. Default is ISO-8859-1 check the QGIS desktop DXF export dialog for valid values.
- NO_MTEXT: Use TEXT instead of MTEXT for labels.
- FORCE_2D: Force 2D output. This is required for polyline width.
For performance reasons, QGIS Server can be used in tiled mode. In this mode, the client requests several small fixed size tiles, and assembles them to form the whole map. Doing this, symbols at or near the boundary between two tiles may appeared cut, because they are only present in one of the tile.

Set the `TILED` parameter to `TRUE` to tell QGIS Server to work in tiled mode, and to apply the `Tile buffer` configured in the QGIS project (see Configure your project).

When `TILED` is `TRUE` and when a non-zero Tile buffer is configured in the QGIS project, features outside the tile extent are drawn to avoid cut symbols at tile boundaries.

`TILED` defaults to `FALSE`.

### 3.2.3 GetFeatureInfo

Standard parameters for the `GetFeatureInfo` request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugos pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetFeatureInfo)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>QUERY_L</td>
<td>Yes</td>
<td>Layers to query</td>
</tr>
<tr>
<td>LAYERS</td>
<td>Yes</td>
<td>Layers to display (identical to QUERY_LAYERS)</td>
</tr>
<tr>
<td>STYLES</td>
<td>No</td>
<td>Layers’ style</td>
</tr>
<tr>
<td>SRS / CRS</td>
<td>Yes</td>
<td>Coordinate reference system</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>Map extent</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes</td>
<td>Width of the image in pixels</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Yes</td>
<td>Height of the image in pixels</td>
</tr>
<tr>
<td>TRANSPAI</td>
<td>No</td>
<td>Transparent background</td>
</tr>
<tr>
<td>INFO_FORMAT</td>
<td>No</td>
<td>Output format</td>
</tr>
<tr>
<td>FEATURE_COUNT</td>
<td>No</td>
<td>Maximum number of features to return</td>
</tr>
<tr>
<td>I</td>
<td>No</td>
<td>Pixel column of the point to query</td>
</tr>
<tr>
<td>X</td>
<td>No</td>
<td>Same as <code>I</code> parameter, but in WMS 1.1.1</td>
</tr>
<tr>
<td>J</td>
<td>No</td>
<td>Pixel row of the point to query</td>
</tr>
<tr>
<td>Y</td>
<td>No</td>
<td>Same as <code>J</code> parameter, but in WMS 1.1.1</td>
</tr>
<tr>
<td>WMS_PR1</td>
<td>No</td>
<td>The precision (number of digits) to be used when returning geometry (see how to add geometry to feature response). The default value is –1 meaning that the precision defined in the project is used.</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>Subset of features</td>
</tr>
<tr>
<td>FI_POINT_TOLERANCE</td>
<td>No</td>
<td>Tolerance in pixels for point layers</td>
</tr>
<tr>
<td>FI_LINE_TOLERANCE</td>
<td>No</td>
<td>Tolerance in pixels for line layers</td>
</tr>
<tr>
<td>FI_POLYGON_TOLERANCE</td>
<td>No</td>
<td>Tolerance in pixels for polygon layers</td>
</tr>
<tr>
<td>FILTER_GEOM</td>
<td>No</td>
<td>Geometry filtering</td>
</tr>
<tr>
<td>WITH_MAPTIP</td>
<td>No</td>
<td>Add map tips to the output</td>
</tr>
<tr>
<td>WITH_GEOMETRY</td>
<td>No</td>
<td>Add geometry to the output</td>
</tr>
</tbody>
</table>

URL example:
INFO_FORMAT

This parameter may be used to specify the format of the result. Available values are:

- text/xml
- text/html
- text/plain
- application/vnd.ogc.gml
- application/json

QUERY_LAYERS

This parameter specifies the layers to display on the map. Names are separated by a comma.

In addition, QGIS Server introduces options to select layers by:

- short name
- layer id

See the LAYERS parameter defined in GetMap for more information.

FEATURE_COUNT

This parameter specifies the maximum number of features per layer to return. For example if QUERY_LAYERS is set to layer1,layer2 and FEATURE_COUNT is set to 3 then a maximum of 3 features from layer1 will be returned. Likewise a maximum of 3 features from layer2 will be returned.

By default, only 1 feature per layer is returned.
I
This parameter, defined in WMS 1.3.0, allows you to specify the pixel column of the query point.

X
Same parameter as I, but defined in WMS 1.1.1.

J
This parameter, defined in WMS 1.3.0, allows you to specify the pixel row of the query point.

Y
Same parameter as J, but defined in WMS 1.1.1.

FI_POINT_TOLERANCE
This parameter specifies the tolerance in pixels for point layers.

FI_LINE_TOLERANCE
This parameter specifies the tolerance in pixels for line layers.

FI_POLYGON_TOLERANCE
This parameter specifies the tolerance in pixels for polygon layers.

FILTER_GEOM
This parameter specifies a WKT geometry with which features have to intersect.

WITH_MAPTIP
This parameter specifies whether to add map tips to the output.
Available values are (not case sensitive):
  • TRUE
  • FALSE

WITH_GEOMETRY
This parameter specifies whether to add geometries to the output. To use this feature you must first enable the Add geometry to feature response option in the QGIS project. See Configure your project.
Available values are (not case sensitive):
  • TRUE
  • FALSE
3.2.4 GetLegendGraphic

Standard parameters for the GetLegendGraphic request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugos pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetLegendGraphic)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>LAYERS</td>
<td>Yes</td>
<td>Layers to display</td>
</tr>
<tr>
<td>STYLES</td>
<td>No</td>
<td>Layers' style</td>
</tr>
<tr>
<td>SRS / CRS</td>
<td>No</td>
<td>Coordinate reference system</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>Map extent</td>
</tr>
<tr>
<td>WIDTH</td>
<td>No</td>
<td>Width of the image in pixels</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>No</td>
<td>Height of the image in pixels</td>
</tr>
<tr>
<td>FORMAT</td>
<td>No</td>
<td>Legend format</td>
</tr>
<tr>
<td>TRANSPARENT</td>
<td>No</td>
<td>Transparent background</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports extra parameters to change the size of the legend elements or the font properties for layer titles and item labels:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>SRCWIDTH</td>
<td>No</td>
<td>Map width</td>
</tr>
<tr>
<td>SRCHEIGHT</td>
<td>No</td>
<td>Map height</td>
</tr>
<tr>
<td>SHOWFEATURECOUNT</td>
<td>No</td>
<td>Add feature count of features</td>
</tr>
<tr>
<td>RULE</td>
<td>No</td>
<td>Rule symbol to render</td>
</tr>
<tr>
<td>RULELABEL</td>
<td>No</td>
<td>Item labels rendering</td>
</tr>
<tr>
<td>BOXSPACE</td>
<td>No</td>
<td>Space between legend frame and content (mm)</td>
</tr>
<tr>
<td>LAYERSPACE</td>
<td>No</td>
<td>Vertical space between layers (mm)</td>
</tr>
<tr>
<td>LAYERTITLESPACE</td>
<td>No</td>
<td>Vertical space between layer title and items (mm)</td>
</tr>
<tr>
<td>SYMBOLSPACE</td>
<td>No</td>
<td>Vertical space between symbol and items (mm)</td>
</tr>
<tr>
<td>ICONLABELSPACE</td>
<td>No</td>
<td>Horizontal space between symbol and label (mm)</td>
</tr>
<tr>
<td>SYMBOLWIDTH</td>
<td>No</td>
<td>Width of the symbol preview (mm)</td>
</tr>
<tr>
<td>SYMBOLHEIGHT</td>
<td>No</td>
<td>Height of the symbol preview (mm)</td>
</tr>
<tr>
<td>LAYERTITLE</td>
<td>No</td>
<td>Layer title rendering</td>
</tr>
<tr>
<td>LAYERFONTFAMILY</td>
<td>No</td>
<td>Layer font family</td>
</tr>
<tr>
<td>LAYERFONTBOLD</td>
<td>No</td>
<td>Layer title bold rendering</td>
</tr>
<tr>
<td>LAYERFONTSIZE</td>
<td>No</td>
<td>Layer title font size (pt)</td>
</tr>
<tr>
<td>LAYERFONTITALIC</td>
<td>No</td>
<td>Layer title italic rendering</td>
</tr>
<tr>
<td>LAYERFONTCOLOR</td>
<td>No</td>
<td>Layer title color</td>
</tr>
<tr>
<td>ITEMFONTFAMILY</td>
<td>No</td>
<td>Item font family</td>
</tr>
<tr>
<td>ITEMFONTBOLD</td>
<td>No</td>
<td>Item label bold rendering</td>
</tr>
<tr>
<td>ITEMFONTSIZE</td>
<td>No</td>
<td>Item label font size (pt)</td>
</tr>
<tr>
<td>ITEMFONTITALIC</td>
<td>No</td>
<td>Item label italic rendering</td>
</tr>
<tr>
<td>ITEMFONTCOLOR</td>
<td>No</td>
<td>Item label color</td>
</tr>
</tbody>
</table>
BBOX

This parameter can be used to specify the geographical area for which the legend should be built (its format is described here) but cannot be combined with the RULE parameter. The SRS/CRS parameter becomes mandatory when using the BBOX parameter.

URL example:

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYER=countries,airports
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326

Pastaba: When the BBOX parameter is defined, the legend is referred to as a content based legend.

WIDTH

This parameter is not used by default but becomes mandatory when the RULE parameter is set. In this case it allows to specify the width in pixels of the output image.

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYER=testlayer%20%C3%A8%C3%A9
&RULE=rule1
&WIDTH=30
&HEIGHT=30

HEIGHT

This parameter is not used by default but becomes mandatory when the RULE parameter is set. In this case it allows to specify the height in pixels of the output image.

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYER=testlayer%20%C3%A8%C3%A9
&RULE=rule1
&WIDTH=30
&HEIGHT=30

FORMAT

This parameter may be used to specify the format of legend image. Available values are:

- image/jpeg
- image/png
- application/json

For JSON, symbols are encoded with Base64 and most other options related to layout or fonts are not taken into account because the legend must be built on the client side. The RULE parameter cannot be combined with this format.

URL example with the corresponding JSON output:
http://localhost/qgisserver?
SERVICE=WMS&
REQUEST=GetLegendGraphic&
LAYERS=airports&
FORMAT=application/json

And the corresponding JSON output:

```
{
    "nodes": [
        {
            "icon": "<base64 icon>",
            "title": "airports",
            "type": "layer"
        }
    ],
    "title": ""
}
```

**SRCWIDTH**

This parameter may be defined when the `RULE` parameter is set. In this case, the `SRCWIDTH` value is forwarded to the underlying `GetMap` request as the `WIDTH` parameter while the `WIDTH` parameter of `GetLegendGraphic` is used for the image legend size.

**SRCHEIGHT**

This parameter may be defined when the `RULE` parameter is set. In this case, the `SRCHEIGHT` value is forwarded to the underlying `GetMap` request as the `HEIGHT` parameter while the `HEIGHT` parameter of `GetLegendGraphic` is used for the image legend size.

**SHOWFEATURECOUNT**

This parameter can be used to activate feature count in the legend. Available values are (not case sensitive):

- TRUE
- FALSE

For example:

```
Tram lines [10]
- 1 [2]
- 2 [2]
- 3 [4]
- 4 [2]
```
RULE

This parameter is available on layers with Rule-based rendering and allows to build a legend with only the named rule symbol. It cannot be combined with BBOX parameter nor the JSON format.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=mylayer,
&RULE=myrulename
```

RULELABEL

This parameter allows to control the item label rendering. Available values are (not case sensitive):

- TRUE: display item label
- FALSE: hide item label
- AUTO: hide item label for layers with Single symbol rendering

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,airports
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&RULELABEL=AUTO
```

3.5 Fig.: Legend rendering without label for single symbol layers

3.2. Žiniatinklio žemėlapio paslauga (WMS)
**BOXSPACE**

This parameter allows to specify the space between legend frame and content in millimeters. By default, the space value is 2 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&BOXSPACE=0
```

3.6 Fig.: To the left BOXSPACE=0 and to the right BOXSPACE=15

**LAYERSPACE**

This parameter allows to specify the vertical space between layers in millimeters. By default, the space value is 3 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERSPACE=0
```

3.7 Fig.: To the left LAYERSPACE=0 and to the right LAYERSPACE=10
LAYERTITLESPACE

This parameter allows to specify the vertical space between layer title and items following in millimeters. By default the space value is 3 mm.

URL example:

```xml
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERTITLESPACE=0
```

3.8 Fig.: To the left LAYERTITLESPACE=0 and to the right LAYERTITLESPACE=10

SYMBOLSPACE

This parameter allows to specify the vertical space between symbol and item following in millimeters. By default the space value is 2 mm.

URL example:

```xml
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&SMBOLSPACE=0
```
ICONLABELSPACE

This parameter allows to specify the horizontal space between symbol and label text in millimeters. By default the space value is 2 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=getlegendgraphic
&LAYERS=countries,
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ICONLABELSPACE=0
```

3.9 Fig.: To the left SYMBOLSPACE=0 and to the right SYMBOLSPACE=5

3.10 Fig.: To the left ICONLABELSPACE=0 and to the right ICONLABELSPACE=10
SYMBOLWIDTH

This parameter allows to specify the width of the symbol preview in millimeters. By default the width value is 7 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&SYMBOLWIDTH=2
```

3.11 Fig.: To the left SYMBOLWIDTH=2 and to the right SYMBOLWIDTH=20

SYMBOLHEIGHT

This parameter allows to specify the height of the symbol preview in millimeters. By default the height value is 4 mm.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries,
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&SYMBOLHEIGHT=2
```
3.12 Fig.: To the left $\text{SYMBOLHEIGHT}=2$ and to the right $\text{SYMBOLHEIGHT}=6$

**LAYERTITLE**

This parameter specifies whether to render layer title.

Available values are (not case sensitive):

- **TRUE** (default value)
- **FALSE**

**LAYERFONTFAMILY**

This parameter specifies the font family to use for rendering layer title.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=countries
&LAYERFONTFAMILY=monospace
```

**LAYERFONTBOLD**

This parameter specifies whether the layer title is rendered in bold. Available values are (not case sensitive):

- **TRUE**
- **FALSE**

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTBOLD=TRUE
```
3.13 Fig.: Legend with \texttt{LAYERFONTBOLD=TRUE}

**LAYERFONTSIZE**

This parameter specifies the font size for rendering layer title in point.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTSIZE=20
```

3.14 Fig.: Legend with \texttt{LAYERFONTSIZE=20}

**LAYERFONTITALIC**

This parameter specifies whether the layer title is rendered in italic. Available values are (not case sensitive):

- \texttt{TRUE}
- \texttt{FALSE}

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTITALIC=TRUE
```
### LAYERFONTCOLOR

This parameter specifies the layer title color. The color may be literal (red, green, ...) or in hexadecimal notation (0xFF0000, 0x00FF00, ...).

URL example:

```plaintext
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYER=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&LAYERFONTCOLOR=0x5f9930
```

![Legend with LAYERFONTCOLOR=0x5f9930](image)

### ITEMFONTFAMILY

This parameter specifies the font family to use for rendering item label.

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYER=countries
&ITEMFONTFAMILY=monospace
```
ITEMFONTBOLD

This parameter specifies whether the item label is rendered in bold. Available values are (not case sensitive):

- TRUE
- FALSE

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTBOLD=TRUE
```

3.17 Fig.: Legend with ITEMFONTBOLD=TRUE

ITEMFONTSIZE

This parameter specifies the font size for rendering layer title in point.

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTSIZE=20
```

3.18 Fig.: Legend with ITEMFONTSIZE=30
ITEMFONITALIC

This parameter specifies whether the item label is rendered in italic. Available values are (not case sensitive):

- TRUE
- FALSE

URL example:

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONITALIC=TRUE

3.19 Fig.: Legend with ITEMFONITALIC=TRUE

ITEMFONTCOLOR

This parameter specifies the item label color. The color may be literal (red, green, ...) or in hexadecimal notation (0xFF0000, 0x00FF00, ...).

URL example:

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetLegendGraphic
&LAYERS=airports,places
&BBOX=43.20,-2.93,49.35,8.32
&CRS=EPSG:4326
&TRANSPARENT=TRUE
&ITEMFONTCOLOR=0x5f9930

3.20 Fig.: Legend with ITEMFONTCOLOR=0x5f9930
3.2.5 GetStyle(s)

Standard parameters for the GetStyle (or GetStyles) request according to the OGC WMS 1.1.1 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugas pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetStyle or GetStyles)</td>
</tr>
<tr>
<td>LAYERS</td>
<td>Yes</td>
<td>Layers to query</td>
</tr>
</tbody>
</table>

The GetStyle request supports as well the following vendor parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetStyles
&LAYERS=mylayer1,mylayer2

3.2.6 DescribeLayer

Standard parameters for the DescribeLayer request according to the OGC WMS 1.1.1 and 1.3.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugos pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (DescribeLayer)</td>
</tr>
<tr>
<td>LAYERS</td>
<td>Yes</td>
<td>Layers to describe</td>
</tr>
<tr>
<td>SLD_VERSION</td>
<td>Yes</td>
<td>SLD version</td>
</tr>
</tbody>
</table>

The DescribeLayer request supports as well the following vendor parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=DescribeLayer
&SLD_VERSION=1.1.0
&LAYERS=mylayer1

The XML document looks like:

```xml
  xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld/1.1.0/DescribeLayer.xsd">
  <Version>1.1.0</Version>
</DescribeLayerResponse>
```

(continues on next page)
SLD_VERSION

This parameter allows to specify the version of SLD. Only the value 1.1.0 is available.

3.2.7 GetPrint

QGIS Server has the capability to create print layout output in pdf or pixel format. Print layout windows in the published project are used as templates. In the GetPrint request, the client has the possibility to specify parameters of the contained layout maps and labels.

The GetPrint request supports redlining, external WMS layers as well as the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugų pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetPrint)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>LAYERS</td>
<td>No</td>
<td>Layers to display</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>Yes</td>
<td>Layout template to use</td>
</tr>
<tr>
<td>SRS / CRS</td>
<td>Yes</td>
<td>Coordinate reference system</td>
</tr>
<tr>
<td>FORMAT</td>
<td>No</td>
<td>Output format</td>
</tr>
<tr>
<td>ATLAS_PK</td>
<td>No</td>
<td>Atlas features</td>
</tr>
<tr>
<td>STYLES</td>
<td>No</td>
<td>Layers' style</td>
</tr>
<tr>
<td>TRANSPARENT</td>
<td>No</td>
<td>Transparent background</td>
</tr>
<tr>
<td>OPACITIES</td>
<td>No</td>
<td>Opacity for layer or group</td>
</tr>
<tr>
<td>SELECTION</td>
<td>No</td>
<td>Highlight features</td>
</tr>
<tr>
<td>mapX:EXTENT</td>
<td>No</td>
<td>Extent of the map X</td>
</tr>
<tr>
<td>mapX:LAYERS</td>
<td>No</td>
<td>Layers of the map X</td>
</tr>
<tr>
<td>mapX:STYLES</td>
<td>No</td>
<td>Layers' style of the map X</td>
</tr>
<tr>
<td>mapX:SCALE</td>
<td>No</td>
<td>Layers' scale of the map X</td>
</tr>
<tr>
<td>mapX:ROTATION</td>
<td>No</td>
<td>Rotation of the map X</td>
</tr>
<tr>
<td>mapX:GRID_INTERVAL_X</td>
<td>No</td>
<td>Grid interval on x axis of the map X</td>
</tr>
<tr>
<td>mapX:GRID_INTERVAL_Y</td>
<td>No</td>
<td>Grid interval on y axis of the map X</td>
</tr>
</tbody>
</table>

URL example:
Note that the layout template may contain more than one map. In this way, if you want to configure a specific map, you have to use mapX: parameters where X is a positive number that you can retrieve thanks to the GetProjectSettings request.

For example:

```xml
<WMS_Capabilities>
  ...
  <ComposerTemplates xsi:type="wms:_ExtendedCapabilities">
    <ComposerTemplate width="297" height="210" name="Druckzusammenstellung 1">
      <ComposerMap width="171" height="133" name="map0"/>
    </ComposerTemplate>
    <ComposerTemplate width="49" height="46" name="map1"/>
  </ComposerTemplates>
  ...
</WMS_Capabilities>
```

**TEMPLATE**

This parameter can be used to specify the name of a layout template to use for printing.

**FORMAT**

This parameter specifies the format of map image. Available values are:

- `png` (default value)
- `image/png`
- `jpg`
- `jpeg`
- `image/jpeg`
- `svg`
- `image/svg`
- `image/svg+xml`
- `pdf`
- `application/pdf`

If the `FORMAT` parameter is different from one of these values, then an exception is returned.
**ATLAS_PK**

This parameter allows activation of Atlas rendering by indicating which features we want to print. In order to retrieve an atlas with all features, the * symbol may be used (according to the maximum number of features allowed in the project configuration).

When `FORMAT` is `pdf`, a single PDF document combining the feature pages is returned. For all other formats, a single page is returned.

**mapX:EXTENT**

This parameter specifies the extent for a layout map item as `xmin,ymin,xmax,ymax`.

**mapX:ROTATION**

This parameter specifies the map rotation in degrees.

**mapX:GRID_INTERVAL_X**

This parameter specifies the grid line density in the X direction.

**mapX:GRID_INTERVAL_Y**

This parameter specifies the grid line density in the Y direction.

**mapX:SCALE**

This parameter specifies the map scale for a layout map item. This is useful to ensure scale based visibility of layers and labels even if client and server may have different algorithms to calculate the scale denominator.

**mapX:LAYERS**

This parameter specifies the layers for a layout map item. See `GetMap Layers` for more information on this parameter.

**mapX:STYLES**

This parameter specifies the layers' styles defined in a specific layout map item. See `GetMap Styles` for more information on this parameter.

### 3.2.8 GetProjectSettings

This request type works similar to `GetCapabilities`, but it is more specific to QGIS Server and allows a client to read additional information which are not available in the `GetCapabilities` output:

- initial visibility of layers
- information about vector attributes and their edit types
- information about layer order and drawing order
- list of layers published in WFS
- show if a group in the layer tree is mutually exclusive
The **GetProjectSettings** request supports the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugos pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetProjectSettings)</td>
</tr>
</tbody>
</table>

### 3.2.9 GetSchemaExtension

The **GetSchemaExtension** request allows to retrieve optional extended capabilities and operations of the WMS service such as implemented by QGIS Server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Paslaugos pavadinimas (WMS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetSchemaExtension)</td>
</tr>
</tbody>
</table>

URL example:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetSchemaExtension
```

The XML document looks like:

```

targetNamespace="http://www.qgis.org/wms"
elementFormDefault="qualified" version="1.0.0">
  <import namespace="http://www.opengis.net/wms" schemaLocation="http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd"/>
  <element name="GetPrint" type="wms:OperationType" substitutionGroup="wms:_ExtendedOperation"/>
  <element name="GetStyles" type="wms:OperationType" substitutionGroup="wms:_ExtendedOperation"/>
</schema>
```

### 3.2.10 External WMS layers

QGIS Server allows including layers from external WMS servers in WMS **GetMap** and WMS **GetPrint** requests. This is especially useful if a web client uses an external background layer in the web map. For performance reasons, such layers should be directly requested by the web client (not cascaded via QGIS server). For printing however, these layers should be cascaded via QGIS server in order to appear in the printed map.

External layers can be added to the LAYERS parameter as EXTERNAL_WMS:<layername>. The parameters for the external WMS layers (e.g. url, format, dpiMode, crs, layers, styles) can later be given as service parameters <layername>:<parameter>. In a **GetMap** request, this might look like this:

```
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&LAYERS=EXTERNAL_WMS:basemap,layer1,layer2
&OPACITIES=100,200,255
&STYLES=,
&basemap:url=http://externalserver.com/wms.fcgi
&basemap:format=image/jpeg
```

(continues on next page)
Similarly, external layers can be used in *GetPrint* requests:

```xml
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetPrint
&TEMPLATE=A4
&map0:layers=EXTERNAL_WMS:basemap,layer1,layer2
&map0:EXTENT=<minx,miny,maxx,maxy>
&OPACITIES=100,200,255
&basemap:url=http://externalserver.com/wms.fcgi
&basemap:format=image/jpeg
&basemap:dpiMode=7
&basemap:crs=EPSG:2056
&basemap:layers=orthofoto
&basemap:styles=default
```

### 3.2.11 Redlining

This feature is available and can be used with *GetMap* and *GetPrint* requests.

The redlining feature can be used to pass geometries and labels in the request which are overlapped by the server over the standard returned image (map). This permits the user to put emphasis or maybe add some comments (labels) to some areas, locations etc. that are not in the standard map.

The *GetMap* request is in the format:

```xml
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetMap
&HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620, 2500000 508000, 2500000 5647000))
&HIGHLIGHT_SYMBOL=<StyledLayerDescriptor><UserStyle><Name>Highlight</Name><FeatureTypeStyle><Rule><Name>Symbol</Name><LineSymbolizer><Stroke><SvgParameter name="stroke" value="#ea1173"/><SvgParameter name="stroke-opacity" value="1"/><SvgParameter name="stroke-width" value="1.6"/></Stroke></LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
&HIGHLIGHT_LABELSTRING=Write label here
&HIGHLIGHTLABELSIZE=16
&HIGHLIGHTLABELCOLOR=%23000000
&HIGHLIGHTLABELBUFFERCOLOR=%23FFFFFF
&HIGHLIGHTLABELBUFFERSIZE=1.5
```

The *GetPrint* equivalent is in the format (note that `mapX:` parameter is added to tell which map has redlining):

```xml
http://localhost/qgisserver?
SERVICE=WMS
&REQUEST=GetPrint
&map0:HIGHLIGHT_GEOM=POLYGON((590000 5647000, 590000 6110620, 2500000 6110620, 2500000 508000, 2500000 5647000))
&map0:HIGHLIGHT_SYMBOL=\<StyledLayerDescriptor\><UserStyle\><Name>Highlight</Name><FeatureTypeStyle\><Rule\><Name>Symbol\</Name><LineSymbolizer\><Stroke\><SvgParameter name="stroke" value="#ea1173"/><SvgParameter name="stroke-opacity" value="1"/><SvgParameter name="stroke-width" value="1.6"/></Stroke></LineSymbolizer></Rule></FeatureTypeStyle></UserStyle></StyledLayerDescriptor>
&map0:HIGHLIGHT_LABELSTRING=Write label here
&map0:HIGHLIGHTLABELSIZE=16
&map0:HIGHLIGHTLABELCOLOR=%23000000
&map0:HIGHLIGHTLABELBUFFERCOLOR=%23FFFFFF
&map0:HIGHTLIGTHEATLABELBUFFERSIZE=1.5
```

(continues on next page)
Here is the image outputed by the above request in which a polygon and a label are drawn on top of the normal map:

You can see there are several parameters in this request to control the redlining feature. The full list includes:

- **HIGHLIGHT_GEOM**: You can add POINT, MULTILINESTRING, POLYGON etc. It supports multipart geometries. Here is an example: `HIGHLIGHT_GEOM=MULTILINESTRING((0 0, 0 1, 1 1))`. The coordinates should be in the CRS of the GetMap/GetPrint request.

- **HIGHLIGHT_LABELBUFFERCOLOR**: This parameter controls the label buffer color.

- **HIGHLIGHT_LABELBUFFERSIZE**: This parameter controls the label buffer size.

- **HIGHLIGHT_LABELCOLOR**: This parameter controls the label color.

- **HIGHLIGHT_LABEL_DISTANCE**: controls the distance between feature (e.g. point or line) and the label in mm

- **HIGHLIGHT_LABELFONT**: This parameter controls the font of the label (e.g. Arial)
- **HIGHLIGHT_LABEL_HORIZONTAL_ALIGNMENT**: places the label horizontally on a point using the specified alignment (e.g. 'left', 'center', 'right')
- **HIGHLIGHT_LABEL_ROTATION**: controls the label rotation in degrees
- **HIGHLIGHT_LABELSIZE**: This parameter controls the size of the label.
- **HIGHLIGHT_LABELSTRING**: You can pass your labeling text to this parameter.
- **HIGHLIGHT_LABEL_VERTICAL_ALIGNMENT**: places the label vertically on a point using the specified alignment (e.g. 'top', 'half', 'bottom')
- **HIGHLIGHT_SYMBOL**: This controls how the geometry is outlined and you can change the stroke width, color and opacity.

### 3.3 Tinklo Geoobjektų Paslauga - Web Feature Service (WFS)

QGIS Server įgyvendinti 1.0.0 ir 1.1.0 WFS standartai teikia HTTP sąsają gauti geografinius geoobjektus iš QGIS projekto. Tipinėje WFS užklausojų struktūroje nurodomas QGIS projektas ir sluoksnis, kuriamo turi būti vykdoma užklausa.

Specifikacijos dokumentas pagal paslaugos versiją:
- WFS 1.0.0
- WFS 1.1.0

Standartinės QGIS Serverio teikiamos užklausos:

<table>
<thead>
<tr>
<th>Užklausa</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>GetFeature</td>
<td>Returns a selection of features</td>
</tr>
<tr>
<td>DescribeFeatureType</td>
<td>Returns a description of feature types and properties</td>
</tr>
<tr>
<td>Transaction</td>
<td>Allows features to be inserted, updated or deleted</td>
</tr>
</tbody>
</table>

#### 3.3.1 GetCapabilities

Standard parameters for the GetCapabilities request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WFS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetCapabilities)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>
**VERSION**

This parameter allows to specify the version of the service to use. Available values for the `VERSION` parameter are:

- 1.0.0
- 1.1.0

If no version is indicated in the request, then 1.1.0 is used by default.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&...
```

### 3.3.2 GetFeature

Standard parameters for the **GetFeature** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WFS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetFeature)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>TYPENAME</td>
<td>No</td>
<td>Name of layers</td>
</tr>
<tr>
<td>FEATUREID</td>
<td>No</td>
<td>Filter the features by ids</td>
</tr>
<tr>
<td>OUTPUTFORMAT</td>
<td>No</td>
<td>Output Format</td>
</tr>
<tr>
<td>RESULTTYPE</td>
<td>No</td>
<td>Type of the result</td>
</tr>
<tr>
<td>PROPERTYNAME</td>
<td>No</td>
<td>Name of properties to return</td>
</tr>
<tr>
<td>MAXFEATURES</td>
<td>No</td>
<td>Maximum number of features to return</td>
</tr>
<tr>
<td>SRSNAME</td>
<td>No</td>
<td>Coordinate reference system</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>OGC Filter Encoding</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>Map Extent</td>
</tr>
<tr>
<td>SORTBY</td>
<td>No</td>
<td>Sort the results</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>STARTINDEX</td>
<td>No</td>
<td>Paging</td>
</tr>
<tr>
<td>GEOMETRYNAME</td>
<td>No</td>
<td>Type of geometry to return</td>
</tr>
<tr>
<td>EXP_FILTER</td>
<td>No</td>
<td>Expression filtering</td>
</tr>
</tbody>
</table>

**TYPENAME**

This parameter allows to specify layer names and is mandatory if `FEATUREID` is not set.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&TYPENAME=countries
```
FEATUREID

This parameter allows to specify the ID of a specific feature and is formed like `typename.fid,typename.fid,...`.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&FEATUREID=countries.0,places.1
```

XML response:

```
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd http://www.qgis.org/gml http://192.168.1.15/qgisserver?SERVICE=WFS&VERSION=1.1.0&REQUEST=DescribeFeatureType&TYPENAME=countries,places&OUTPUTFORMAT=text/xml; subtype%3Dgml/3.1.1">
  <gml:boundedBy>
  ...  
  </gml:boundedBy>
  <gml:featureMember>
    <qgs:countries gml:id="countries.1">
      ...
    </qgs:countries>
  </gml:featureMember>
  <gml:featureMember>
    <qgs:places gml:id="places.1">
      ...
    </qgs:places>
  </gml:featureMember>
</wfs:FeatureCollection>
```

OUTPUTFORMAT

This parameter may be used to specify the format of the response. If `VERSION` is greater or equal than `1.1.0`, GML3 is the default format. Otherwise GML2 is used.

Available values are:

- `gml2`
- `text/xml; subtype=gml/2.1.2`
- `gml3`
- `text/xml; subtype=gml/3.1.1`
- `geojson`
- `application/vnd.geo+json`,
- `application/vnd.geo json`
- `application/geo+json`
- `application/geo json`
- `application/json`

URL example:
GeoJSON response:

```
{
  "type":"FeatureCollection",
  "bbox": [  
    -180,  
    -90,  
    180,  
    83.6236  
  ],
  "features": [
    {
      "bbox": [  
        -61.891113,  
        16.989719,  
        -61.666389,  
        17.724998  
      ],
      "geometry": {  
        "coordinates": [  
          "..."  
        ],
        "type": "MultiPolygon"
      },
      "id": "countries.1",
      "properties": {
        "id": 1,
        "name": "Antigua and Barbuda"
      },
      "type": "Feature"
    }
  ]
}
```

**RESULTTYPE**

This parameter may be used to specify the kind of result to return. Available values are:

- **results**: the default behavior
- **hits**: returns only a feature count

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&FEATUREID=countries.0
&OUTPUTFORMAT=geojson
```

---

3.3. Tinklo Geoobjektų Paslauga - Web Feature Service (WFS) 61
**PROPERTYNANE**

This parameter may be used to specify a specific property to return. A property needs to be mapped with a **TYPENAME** or a **FEATUREID**:

Valid URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&PROPERTYNAME=name
&TYPENAME=places
```

On the contrary, the next URL will return an exception:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&PROPERTYNAME=name
&TYPENAME=places,countries
```

```xml
<ServiceExceptionReport xmlns="http://www.opengis.net/ogc" version="1.2.0">
  <ServiceException code="RequestNotWellFormed">There has to be a 1:1 mapping between each element in a **TYPENAME** and the **PROPERTYNAME** list</ServiceException>
</ServiceExceptionReport>
```

**MAXFEATURES**

This parameter allows to limit the number of features returned by the request.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&MAXFEATURES=1000
```

**Pastaba:** This parameter may be useful to improve performances when underlying vector layers are heavy.

**SRSNAME**

This parameter allows to indicate the response output Spatial Reference System as well as the BBOX CRS and has to be formed like `EPSG:XXXX`.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&SRSNAME=EPSG:32620
```
FILTER

This parameter allows to filter the response with the Filter Encoding language defined by the OGC Filter Encoding standard. For example:

```
http://localhost/qgisserver?
SERVICE=WFS
REQUEST=GetFeature
TYPENAME=places
FILTER=<Filter><PropertyIsEqualTo><PropertyName>name</PropertyName><Literal>Paris</Literal></PropertyIsEqualTo></Filter>
```

In case of multiple typenames, filters have to be enclosed in parentheses:

```
http://localhost/qgisserver?
SERVICE=WFS
REQUEST=GetFeature
TYPENAME=places,countries
FILTER=(<Filter><PropertyIsEqualTo><PropertyName>name</PropertyName><Literal>Paris</Literal></PropertyIsEqualTo></Filter>)
```

Filter features that intersect with a polygon:

```
http://localhost/qgisserver?
SERVICE=WFS
REQUEST=GetFeature
VERSION=1.1.0
TYPENAME=places
FILTER=<Filter xmlns="http://www.opengis.net/ogc">
   <Intersects>
      <PropertyName>geometry</PropertyName>
      <Polygon xmlns="http://www.opengis.net/gml" srsName="EPSG:4326">
         <exterior>
            <LinearRing>
               <posList>
                  -0.6389 42.5922
                  10.2683 51.9106
                  14.5196 41.0320
                  -0.6389 42.5922
               </posList>
            </LinearRing>
         </exterior>
      </Polygon>
   </Intersects>
</Filter>
```

BBOX

This parameter allows to specify the map extent with units according to the current CRS. Coordinates have to be separated by a comma.

The SRSNAME parameter may specify the CRS of the extent. If not specified, the CRS of the layer is used.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
REQUEST=GetFeature
TYPENAME=places
BBOX=-11.84,42.53,8.46,50.98
```
The **FEATUREID** parameter cannot be used with the **BBOX**. Any attempt will result in an exception:

```
<ServiceExceptionReport xmlns="http://www.opengis.net/ogc" version="1.2.0">
  <ServiceException code="RequestNotWellFormed">FEATUREID FILTER and BBOX parameters are mutually exclusive</ServiceException>
</ServiceExceptionReport>
```

**SORTBY**

This parameter allows to sort resulting features according to property values and has to be formed like `propertyname SORTRULE`.

Available values for **SORTRULE** in case of descending sorting:
- D
- +D
- DESC
- +DESC

Available values for **SORTRULE** in case of ascending sorting:
- A
- +A
- ASC
- +ASC

**URL example:**

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places
&PROPERTYNAME=name
&MAXFEATURES=3
&SORTBY=name DESC
```

The corresponding result:

```
  <gml:boundedBy>...
  </gml:boundedBy>
  <gml:featureMember>
    <qgs:places gml:id="places.90">
      <qgs:name>Zagreb</qgs:name>
    </qgs:places>
  </gml:featureMember>
  <gml:featureMember>
    <qgs:places gml:id="places.113">
      <qgs:name>Yerevan</qgs:name>
    </qgs:places>
  </gml:featureMember>
</wfs:FeatureCollection>
```

(continues on next page)
GEOMETRYNAME

This parameter can be used to specify the kind of geometry to return for features. Available values are:

- extent
- centroid
- none

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&GEOMETRYNAME=centroid
&...
```

STARTINDEX

This parameter is standard in WFS 2.0, but it's an extension for WFS 1.0.0.

Actually, it can be used to skip some features in the result set and in combination with MAXFEATURES, it provides the ability to page through results.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=GetFeature
&STARTINDEX=2
&...
```

EXP_FILTER

This parameter allows to filter the response with QGIS expressions. The ; character is used to separate filters in case of multiple typenames.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&REQUEST=GetFeature
&TYPENAME=places,countries
&EXP_FILTER="name"='Paris';"name"='France'
```
3.3.3 DescribeFeatureType

Standard parameters for the DescribeFeatureType request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WFS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (DescribeFeatureType)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>OUTPUTFORMAT</td>
<td>No</td>
<td>Format of the response</td>
</tr>
<tr>
<td>TYPENAME</td>
<td>No</td>
<td>Name of layers</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=DescribeFeatureType
&TYPENAME=countries

Output response:

```xml
<schema xmlns:ogc="http://www.opengis.net/ogc" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <element type="qgs:countriesType" substitutionGroup="gml:_Feature" name="countries"/>
  <complexType name="countriesType">
    <complexContent>
      <extension base="gml:AbstractFeatureType">
        <sequence>
          <element minOccurs="0" type="gml:MultiPolygonPropertyType" maxOccurs="1" name="geometry"/>
          <element type="long" name="id" nillable="true" type="string" name="name"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</schema>
```
3.3.4 Transaction

This request allows to update, delete or add one or several features thanks to a XML document. The *delete* action may be achieved with a POST request as well as with the *OPERATION* parameter while the *add* and the *update* operations may be achieved through POST request only.

Standard parameters for the **Transaction** request according to the OGC WFS 1.0.0 and 1.1.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WFS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (Transaction)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>FILTER</td>
<td>No</td>
<td>OGC Filter Encoding</td>
</tr>
<tr>
<td>BBOX</td>
<td>No</td>
<td>Map Extent</td>
</tr>
<tr>
<td>FEATUREID</td>
<td>No</td>
<td>Filter the features by ids</td>
</tr>
<tr>
<td>TYPENAME</td>
<td>No</td>
<td>Name of layers</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
<tr>
<td>OPERATION</td>
<td>No</td>
<td>Specify the operation</td>
</tr>
<tr>
<td>EXP_FILTER</td>
<td>No</td>
<td>Expression filtering</td>
</tr>
</tbody>
</table>

**OPERATION**

This parameter allows to delete a feature without using a POST request with a dedicated XML document.

URL example:

```
http://localhost/qgisserver?
SERVICE=WFS
&VERSION=1.1.0
&REQUEST=Transaction
&OPERATION=DELETE
&FEATUREID=24
```

**Pastaba:** FEATUREID, BBOX and FILTER parameters are mutually exclusive and prioritized in this order.

**Add features**

POST request example:

```
wget --post-file=add.xml "http://localhost/qgisserver?SERVICE=WFS&
REQUEST=Transaction"
```

with the *add.xml* document:

```
<wfs:Transaction service="WFS" version="1.0.0" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd">
```

(continues on next page)
Update features

POST request example:

```
wget --post-file=update.xml "http://localhost/qgisserver?SERVICE=WFS&REQUEST=Transaction"
```

with the `update.xml` document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <wfs:Update typeName="places">
    <wfs:Property>
      <wfs:Name>name</wfs:Name>
      <wfs:Value>Lutece</wfs:Value>
    </wfs:Property>
    <ogc:Filter>
      <ogc:FeatureId fid="24"/>
    </ogc:Filter>
  </wfs:Update>
</wfs:Transaction>
```

Delete features

POST request example:

```
wget --post-file=delete.xml "http://localhost/qgisserver?SERVICE=WFS&REQUEST=Transaction"
```

with the `delete.xml` document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <wfs:Update typeName="places">
    <wfs:Property>
      <wfs:Name>name</wfs:Name>
      <wfs:Value>Locmaria-Plouzané</wfs:Value>
    </wfs:Property>
    <ogc:Filter>
      <ogc:FeatureId fid="24"/>
    </ogc:Filter>
  </wfs:Update>
</wfs:Transaction>
```
3.4 Web Coverage Service (WCS)

The 1.0.0 and 1.1.1 WCS standards implemented in QGIS Server provide a HTTP interface to access raster data, referred to as coverage, coming from a QGIS project.

Specifikacijos

- WCS 1.0.0
- WCS 1.1.1

Standartinės QGIS Serverio teikiamos užklausos:

<table>
<thead>
<tr>
<th>Užklausa</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>DescribeCoverage</td>
<td>Retrieves XML document about additional information about coverages</td>
</tr>
<tr>
<td>GetCoverage</td>
<td>Retrieves coverage</td>
</tr>
</tbody>
</table>

3.4.1 GetCapabilities

Standard parameters for the GetCapabilities request according to the OGC WCS 1.1.1 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WCS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetCapabilities)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/qgisserver?
SERVICE=WCS
&VERSION=1.1.1
&REQUEST=GetCapabilities

XML document example when a single raster layer (named T20QPD_20171123T144719_TCI) is published in the QGIS project for the WCS service:

(continues on next page)
VERSION

This parameter allows to specify the version of the service to use. Currently, the version value is not internally used and always fallback to 1.1.1.

3.4.2 DescribeCoverage

This request allows to retrieve additional information about coverages like the format of the underlying datasource, the number of bands, … Standard parameters for the DescribeCoverage request according to the OGC WCS 1.1.1 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WCS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (DescribeCoverage)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>COVERAGE</td>
<td>No</td>
<td>Specify coverage layers (WCS 1.0.0)</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>No</td>
<td>Specify coverage layers (WCS 1.1.1)</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

The XML document for a 3 bands GeoTIFF raster layer looks like:

```
<CoverageDescription xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.opengis.net/wcs" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" updateSequence="0" version="1.0.0">...
```

(continues on next page)
**COVERAGE**

This parameter, defined in WCS 1.0.0, allows to specify the layers to query for additional information. Names have to be separated by a comma.

In addition, QGIS Server introduced an option to select layers by its short name. The short name of a layer may be configured through Properties ► Metadata in layer menu. If the short name is defined, then it’s used by default instead of the layer’s name:

```
http://localhost/qgisserver?
SERVICE=WCS
&REQUEST=DescribeCoverage
&COVERAGE=mylayer1name,mylayer2shortname
```

**Pastaba:** COVERAGE is mandatory if IDENTIFIER is not set.
IDENTIFIER

This parameter replaces the COVERAGE parameter in WCS 1.1.1. But QGIS Server does not filter according to the VERSION parameter so IDENTIFIER and COVERAGE have the same effect.

Pastaba: IDENTIFIER is mandatory if COVERAGE is not set. If both IDENTIFIER and COVERAGE parameters are defined, COVERAGE is always used in priority.

3.4.3 GetCoverage

This request allows to retrieve the coverage according to specific constraints like the extent or the CRS. Standard parameters for the DescribeCoverage request according to the OGC WCS 1.1.1 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WCS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetCoverage)</td>
</tr>
<tr>
<td>VERSION</td>
<td>No</td>
<td>Version of the service</td>
</tr>
<tr>
<td>COVERAGE</td>
<td>No</td>
<td>Specify coverage layers (WCS 1.0.0)</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>No</td>
<td>Specify coverage layers (WCS 1.1.1)</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes</td>
<td>Width of the response in pixels</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Yes</td>
<td>Height of the response in pixels</td>
</tr>
<tr>
<td>BBOX</td>
<td>Yes</td>
<td>Map extent in CRS units</td>
</tr>
<tr>
<td>CRS</td>
<td>Yes</td>
<td>Coordinate reference system of the extent</td>
</tr>
<tr>
<td>RESPONSE_CRS</td>
<td>No</td>
<td>Coordinate reference system of the response</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

BBOX

This parameter allows to specify the map extent in the units of the current CRS. Coordinates have to be separated by a comma. The BBOX parameter is formed like minx,miny,maxx,maxy.

URL example:

```
http://localhost/qgisserver?
SERVICE=WCS
&REQUEST=GetCoverage
&IDENTIFIER=T2OQP0D_20171123T144719_TCI
&BBOX=647533,1805950,660987,1813940
&CRS=EPSG:32620
```
CRS

This parameter allows to indicate the Spatial Reference System of the BBOX parameter and has to be formed like EPSG:XXXX.

RESPONSE_CRS

This parameter allows to indicate the output response Spatial Reference System and has to be formed like EPSG:XXXX. The CRS of the corresponding coverage layer is used by default.

WIDTH

This parameter allows to specify the width in pixels of the output image. The resolution of the response image depends on this value.

HEIGHT

This parameter allows to specify the height in pixels of the output image. The resolution of the response image depends on this value.

3.22 Fig. : From left to right: WIDTH=20 & HEIGHT=20, WIDTH=50 & HEIGHT=50, WIDTH=100 & HEIGHT=100

3.5 Web Map Tile Service (WMTS)

The 1.0.0 WMTS standard implemented in QGIS Server provides a HTTP interface to request tiled map images generated from a QGIS project. A typical WMTS request defined the QGIS project to use, some WMS parameters like layers to render, as well as tile parameters.

Specifications document of the service:

- WMTS 1.0.0

Standartinės QGIS Serverio teikiamos užklausos:

<table>
<thead>
<tr>
<th>Užklausa</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCapabilities</td>
<td>Returns XML metadata with information about the server</td>
</tr>
<tr>
<td>GetTile</td>
<td>Returns a tile</td>
</tr>
<tr>
<td>GetFeatureInfo</td>
<td>Grąžina duomenis (geometriją ir reikšmes) pikselio vietai</td>
</tr>
</tbody>
</table>
3.5.1 GetCapabilities

Standard parameters for the GetCapabilities request according to the OGC WMTS 1.0.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMTS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetCapabilities)</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/?
SERVICE=WMTS
&REQUEST=GetCapabilities
&MAP=/home/qgis/projects/world.qgs

3.5.2 GetTile

Standard parameters for the GetTile request according to the OGC WMTS 1.0.0 specifications:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMTS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetTile)</td>
</tr>
<tr>
<td>LAYER</td>
<td>Yes</td>
<td>Layer identifier</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Yes</td>
<td>Output format of the tile</td>
</tr>
<tr>
<td>TILEMATRIXSET</td>
<td>Yes</td>
<td>Name of the pyramid</td>
</tr>
<tr>
<td>TILEMATRIX</td>
<td>Yes</td>
<td>Meshing</td>
</tr>
<tr>
<td>TILEROW</td>
<td>Yes</td>
<td>Row coordinate in the mesh</td>
</tr>
<tr>
<td>TILECOL</td>
<td>Yes</td>
<td>Column coordinate in the mesh</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:

http://localhost/?
SERVICE=WMTS
&REQUEST=GetTile
&MAP=/home/qgis/projects/world.qgs
&LAYER=mylayer
&FORMAT=image/png
&TILEMATRIXSET=EPSG:4326
&TILEROW=0
&TILECOL=0
FORMAT

This parameter may be used to specify the format of tile image. Available values are:

- jpg
- jpeg
- image/jpeg
- image/png

If the FORMAT parameter is different from one of these values, then the default format PNG is used instead.

TILEMATRIXSET

This parameter defines the CRS to use when computing the underlying pyramid. Format: EPSG:XXXX.

TILEMATRIX

This parameter allows to define the matrix to use for the output tile.

TILEROW

This parameter allows to select the row of the tile to get within the matrix.

TILECOL

This parameter allows to select the column of the tile to get within the matrix.

3.5.3 GetFeatureInfo

Standard parameters for the GetFeatureInfo request according to the OGC WMTS 1.0.0 specification:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Yes</td>
<td>Name of the service (WMTS)</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Yes</td>
<td>Name of the request (GetFeatureInfo)</td>
</tr>
<tr>
<td>LAYER</td>
<td>Yes</td>
<td>Layer identifier</td>
</tr>
<tr>
<td>INFOFORMAT</td>
<td>No</td>
<td>Output format</td>
</tr>
<tr>
<td>I</td>
<td>No</td>
<td>X coordinate of a pixel</td>
</tr>
<tr>
<td>J</td>
<td>No</td>
<td>Y coordinate of a pixel</td>
</tr>
<tr>
<td>TILEMATRIXSET</td>
<td>Yes</td>
<td>Name of the pyramid</td>
</tr>
<tr>
<td>TILEMATRIX</td>
<td>Meshing</td>
<td></td>
</tr>
<tr>
<td>TILEROW</td>
<td>Yes</td>
<td>Row coordinate in the mesh</td>
</tr>
<tr>
<td>TILECOL</td>
<td>Yes</td>
<td>Column coordinate in the mesh</td>
</tr>
</tbody>
</table>

In addition to the standard ones, QGIS Server supports the following extra parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>Yes</td>
<td>QGIS project file</td>
</tr>
</tbody>
</table>

URL example:
INFOFORMAT

This parameter allows to define the output format of the result. Available values are:

- text/xml
- text/html
- text/plain
- application/vnd.ogc.gml

The default value is text/plain.

I

This parameter allows to define the X coordinate of the pixel for which we want to retrieve underlying information.

J

This parameter allows to define the Y coordinate of the pixel for which we want to retrieve underlying information.

3.6 OGC API Features (Geoobjektai)

OGC API Features (OAPIF) is the first implementation of the new generation of OGC protocols. It is described by the OGC API - Features - Part 1: Core document.

The API can be reached on typical installations via http://localhost/qgisserver/wfs3

Here is a quick informal summary of the most important differences between the well known WFS protocol and OAPIF:

- OAPIF is based on a REST API
- OAPIF must follow the OPENAPI specifications
- OAPIF supports multiple output formats but it does not dictate any (only GeoJSON and HTML are currently available in QGIS OAPIF) and it uses content negotiation to determine which format is to be served to the client
- JSON and HTML are first class citizens in OAPIF
- OAPIF is self-documenting (through the /api endpoint)
- OAPIF is fully navigable (through links) and browsable

Svarbu: While the OGC API Features implementation in QGIS can make use of the MAP parameter to specify the project file, no extra query parameters are allowed by the OPENAPI specification. For this reason it is strongly recommended that MAP is not exposed in the URL and the project file is specified in the environment by other means (i.e. setting QGIS_PROJECT_FILE in the environment through a web server rewrite rule).
Pastaba: The API endpoint provides comprehensive documentation of all supported parameters and output formats of your service. The following paragraphs will only describe the most important ones.

3.6.1 Resource representation

The implementation of OGC API Features in QGIS Server currently supports the following resource representation (output) formats:

- HTML
- JSON

The format that is actually served will depend on content negotiation, but a specific format can be explicitly requested by appending a format specifier to the endpoints.

Supported format specifier extensions are:

- .json
- .html

Additional format specifier aliases may be defined by specific endpoints:

- .openapi: alias for .json supported by the API endpoint
- .geojson: alias for .json supported by the Features and Feature endpoints

3.6.2 Endpoints

The API provides a list of endpoints that the clients can retrieve. The system is designed in such a way that every response provides a set of links to navigate through all the provided resources.

Endpoints points provided by the QGIS implementation are:

<table>
<thead>
<tr>
<th>Pavadinimis</th>
<th>Path</th>
<th>Aprašymas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Landing</td>
<td>/</td>
<td>General information about the service and provides links to all available endpoints</td>
</tr>
<tr>
<td>Page</td>
<td>/conformance</td>
<td>Information about the conformance of the service to the standards</td>
</tr>
<tr>
<td>API</td>
<td>/api</td>
<td>Full description of the endpoints provided by the service and the returned documents structure</td>
</tr>
<tr>
<td>Collection</td>
<td>/collections</td>
<td>List of all collections (i.e. 'vector layers') provided by the service</td>
</tr>
<tr>
<td></td>
<td>{collectionId}</td>
<td></td>
</tr>
<tr>
<td>Collection</td>
<td>/collections/</td>
<td>Information about a collection (name, metadata, extent etc.)</td>
</tr>
<tr>
<td></td>
<td>{collectionId}</td>
<td></td>
</tr>
<tr>
<td>Savybės</td>
<td>/collections/</td>
<td>List of the features provided by the collection</td>
</tr>
<tr>
<td></td>
<td>{collectionId}/items</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>/collections/</td>
<td>Information about a single feature</td>
</tr>
<tr>
<td></td>
<td>{collectionId}/items/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{featureId}</td>
<td></td>
</tr>
</tbody>
</table>

Similar to WFS-T (transactional Web Feature Service), it is possible to add, update and delete features (CRUD). The respective requests are described on „/api“.
Landing Page

The main endpoint is the **Landing Page**. From that page it is possible to navigate to all the available service endpoints. The **Landing Page** must provide links to:

- the API definition (path `/api link relations service-desc and service-doc`),
- the Conformance declaration (path `/conformance, link relation conformance`), and
- the Collections (path `/collections, link relation data`).

### API Definition

The **API Definition** is an OPENAPI-compliant description of the API provided by the service. In its HTML representation it is a browsable page where all the endpoints and their response formats are accurately listed and documented. The path of this endpoint is `/api`.

The API definition provides a comprehensive and authoritative documentation of the service, including all supported parameters and returned formats.

**Pastaba:** This endpoint is analogue to WFS's GetCapabilities

### Collections list

The collections endpoint provides a list of all the collections available in the service. Since the service „serves“ a single QGIS project the collections are the vector layers from the current project (if they were published as WFS in the project properties). The path of this endpoint is `/collections/`.

3.23 Fig.: Server OAPIF landing page

3.24 Fig.: Server OAPIF collections list page
Collection detail

While the collections endpoint does not provide detailed information about each available collection, that information is available in the `/collections/{collectionId}` endpoints. Typical information includes the extent, a description, CRSs and other metadata.

The HTML representation also provides a browsable map with the available features.

![Collection detail](image)

3.25 Fig. : Server OAPIF collection detail page

Features list

This endpoint provides a list of all features in a collection knowing the collection ID. The path of this endpoint is `/collections/{collectionId}/items`.

The HTML representation also provides a browsable map with the available features.

**Pastaba:** This endpoint is analogue to GetFeature in WFS 1 and WFS 2.
Feature detail

This endpoint provides all the available information about a single feature, including the feature attributes and its geometry. The path of this endpoint is /collections/{collectionId}/items/{itemId}.

The HTML representation also provides a browsable map with the feature geometry.

3.26 Fig. : Server OAPIF features list page

3.27 Fig. : Server OAPIF feature detail page
3.6.3 Pagination

Pagination of a long list of features is implemented in the OGC API through the `next` and `prev` links, QGIS server constructs these links by appending `limit` and `offset` as query string parameters.

**URL example:**

```
```

**Pastaba:** The maximum acceptable value for `limit` can be configured with the `QGIS_SERVER_API_WFS3_MAX_LIMIT` server configuration setting (see: Environment variables).

3.6.4 Feature filtering

The features available in a collection can be filtered/searched by specifying one or more filters.

**Date and time filter**

Collections with date and/or datetime attributes can be filtered by specifying a `datetime` argument in the query string. By default the first date/datetime field is used for filtering. This behavior can be configured by setting a „Date“ or „Time“ dimension in the QGIS Server ► Dimension section of the layer properties dialog.

The date and time filtering syntax is fully described in the API Definition and also supports ranges (begin and end values are included) in addition to single values.

**URL examples:**

- Returns only the features with date dimension matching **2019-01-01**
  ```
  ```

- Returns only the features with datetime dimension matching **2019-01-01T01:01:01**
  ```
  ```

- Returns only the features with datetime dimension in the range **2019-01-01T01:01:01 - 2019-01-01T12:00:00**
  ```
  http://localhost/qgisserver/wfs3/collection_one/items.json?datetime=2019-01-01T01:01:01/2019-01-01T12:00:00
  ```

**Bounding box filter**

A bounding box spatial filter can be specified with the `bbox` parameter:

The order of the comma separated elements is:

- Lower left corner, WGS 84 longitude
- Lower left corner, WGS 84 latitude
- Upper right corner, WGS 84 longitude
- Upper right corner, WGS 84 latitude
**Pastaba:** The OGC specifications also allow a 6 item bbox specifier where the third and sixth items are the Z components, this is not yet supported by QGIS server.

URL example:

```plaintext
http://localhost/qgisserver/wfs3/collection_one/items.json?bbox=-180,-90,180,90
```

If the **CRS** of the bounding box is not **WGS 84**, a different CRS can be specified by using the optional parameter **bbox-crs**. The CRS format identifier must be in the **OGC URI** format:

URL example:

```plaintext
```

### Attribute filters

Attribute filters can be combined with the bounding box filter and they are in the general form: `<attribute name>=<attribute value>`. Multiple filters can be combined using the **AND** operator.

URL example:

filters all features where attribute **name** equals „my value”

```plaintext
http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=my%20value
```

Partial matches are also supported by using a * („star“) operator:

URL example:

filters all features where attribute **name** ends with „value”

```plaintext
http://localhost/qgisserver/wfs3/collection_one/items.json?attribute_one=*value
```

### 3.6.5 Feature sorting

It is possible to order the result set by field value using the **sortby** query parameter.

The results are sorted in ascending order by default. To sort the results in descending order, a boolean flag (**sortdesc**) can be set:

```plaintext
http://localhost/qgisserver/wfs3/collection_one/items.json?sortby=name&sortdesc=1
```

### 3.6.6 Attribute selection

The feature attributes returned by a **Features list** call can be limited by adding a comma separated list of attribute names in the optional **properties** query string argument.

URL example:

returns only the **name** attribute

```plaintext
http://localhost/qgisserver/wfs3/collection_one/items.json?properties=name
```
3.6.7 Customize the HTML pages

The HTML representation uses a set of HTML templates to generate the response. The template is parsed by a template engine called *inJa*. The templates can be customized by overriding them (see: Template overrides). The template has access to the same data that are available to the JSON representation and a few additional functions are available to the template:

**Custom template functions**

- `path_append(path)`: appends a directory path to the current url
- `path_chomp(n)`: removes the specified number „n“ of directory components from the current url path
- `json_dump()`: prints the JSON data passed to the template
- `static(path)`: returns the full URL to the specified static path. For example: „static(„/style/black.css“)“ with a root path „http://localhost/qgissserver/wfs3“ will return „http://localhost/qgissserver/wfs3/static/style/black.css“.
- `links_filter(links, key, value)`: Returns filtered links from a link list
- `content_type_name(content_type)`: Returns a short name from a content type, for example „text/html“ will return „HTML“
- `nl2br(text)`: Returns the input text with all newlines replaced by „<br>“ tags
- `starts_with(string, prefix)`: returns true if a string begins with the provided string prefix, false otherwise

**Template overrides**

Templates and static assets are stored in subdirectories of the QGIS server default API resource directory (/usr/share/qgis/resources/server/api/ on a Linux system), the base directory can be customized by changing the environment variable QGIS_SERVER_API_RESOURCES_DIRECTORY.

A typical Linux installation will have the following directory tree:

```
/usr/share/qgis/resources/server/api/
├── ogc
│   └── static
│       ├── jsonFormatter.min.css
│       ├── jsonFormatter.min.js
│       └── style.css
└── templates
    ├── describeCollection.html
    └── links.html
```

To override the templates you can copy the whole tree to another location and point QGIS_SERVER_API_RESOURCES_DIRECTORY to the new location.
The QGIS Server Catalog is a simple catalog that shows the list of QGIS projects served by the QGIS Server. It provides a user-friendly fully browsable website with basic mapping capabilities to quickly browse the datasets exposed through those QGIS projects.

The QGIS Server catalog uses the variables QGIS_SERVER_LANDING_PAGE_PROJECTS_DIRECTORIES and QGIS_SERVER_LANDING_PAGE_PROJECTS_PG_CONNECTIONS (see Environment variables)
You can consult the metadata associated to a project and the services that it provides. Links to those services are also given.

By browsing a project, it is listed the dataset that it serves.
4.3 Fig.: Browsing a dataset served by a project in the Server Catalog

Use Right click on a layer to display the attribute table associated to it.

4.4 Fig.: Attribute table associated to a layer

It is possible to consult information of the elements in the map as shown in the image below:
4.5 Fig.: Consulting information of a map element
5.1 List of plugins

Plugins can also be installed on QGIS Server.

Some plugins designed for server can be found on the official QGIS repository.

Install only plugins you need for your own purpose. On QGIS server, plugins are like hooks into QGIS server, they can alter inputs or outputs of QGIS server. They can produce unexpected result if you don't know how the plugin works. Please refer to their respective documentation or the application that needs QGIS server plugins to know which plugin can be useful for you.

5.2 Location of plugins

By default, on Debian based systems, QGIS Server will look for plugins located in /usr/lib/qgis/plugins. The default value is displayed when QGIS Server is starting, in the logs. It's possible to set a custom path by defining the environment variable QGIS_PLUGINPATH in the web server configuration.

5.3 Įdiegimas

5.3.1 Manually with a ZIP

As an example, to install the HelloWorld plugin for testing the server, using a specific folder, you first have to create a directory to hold server plugins. This will be specified in the virtual host configuration and passed on to the server through an environment variable:

```
mkdir -p /var/www/qgis-server/plugins
cd /var/www/qgis-server/plugins
wget https://github.com/elpaso/qgis-helloserver/archive/master.zip
unzip master.zip
mv qgis-helloserver-master HelloServer
```
According to its description, HelloServer plugin is designed for development and demonstration purposes. Do not keep this plugin for production if you don’t need it.

5.3.2 With a command line tool

If you need to install and regularly upgrade plugins which are stored in the QGIS plugin repository, you may use the QGIS-Plugin-Manager. It’s a tool to help you manage plugins from the command line.

The installation is using pip. Installing in a virtual environment is a good practice but not required:

```bash
pip3 install qgis-plugin-manager
```

To upgrade the tool:

```bash
pip3 install --upgrade qgis-plugin-manager
```

Then, you can use the `qgis-plugin-manager` executable from the command line:

```bash
cd /var/www/qgis-server/plugins
qgis-plugin-manager list
```

QGIS server version 3.19.0
List all plugins in /var/www/qgis-server/plugins

<table>
<thead>
<tr>
<th>Folder</th>
<th>Name</th>
<th>Version</th>
<th>Experimental</th>
</tr>
</thead>
<tbody>
<tr>
<td>wfsOutputExtension</td>
<td>wfsOutputExtension</td>
<td>1.6.2</td>
<td></td>
</tr>
<tr>
<td>qgis_server_render_geojson</td>
<td>GeoJson Renderer</td>
<td>v0.4</td>
<td></td>
</tr>
<tr>
<td>DataPlotly</td>
<td>Data Plotly</td>
<td>3.7.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Matteo Ghetta (Faunalia)</td>
<td></td>
<td>Upgrade to 3.8.1</td>
</tr>
</tbody>
</table>

We suggest you to read the full documentation in the readme file to know how to install or upgrade plugins with this tool.

5.4 HTTP Server configuration

5.4.1 Apache

To be able to use server plugins, FastCGI needs to know where to look. So, we have to modify the Apache configuration file to indicate the `QGIS_PLUGINPATH` environment variable to FastCGI:

```
FcgidInitialEnv QGIS_PLUGINPATH "/var/www/qgis-server/plugins"
```

Moreover, a basic HTTP authorization is necessary to play with the HelloWorld plugin previously introduced. So we have to update the Apache configuration file a last time:

```
# Needed for QGIS HelloServer plugin HTTP BASIC auth
<IfModule mod_fcgid.c>
```
Then, restart Apache:

```
systemctl restart apache2
```

## 5.5 How to use a plugin

Test the server with the HelloWorld plugin:

```
wget -q -O - "http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=HELLO"
HelloServer!
```

You can have a look at the default GetCapabilities of the QGIS server at:

```
http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
```
6.1 Logging

To log requests sent to the server, you have to set the following environment variable:

- `QGIS_SERVER_LOG_STDERR`

With the following variables the logging can be further customized:

- `QGIS_SERVER_LOG_LEVEL`
- `QGIS_SERVER_LOG_PROFILE`

6.2 Environment variables

You can configure some aspects of QGIS Server by setting environment variables. According to the HTTP server and how you run QGIS Server, there are several ways to define these variables. This is fully described in Apache HTTP Server.

<table>
<thead>
<tr>
<th>Pavadinimas</th>
<th>Aprašymas</th>
<th>Default</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>QGIS_OPTIONS_PATH</td>
<td>Specifies the path to the directory with settings. It works the same way as QGIS application <code>--optionspath</code> option. It is looking for settings file in <code>&lt;QGIS_OPTIONS_PATH&gt;/QGIS/QGIS3.ini</code>.</td>
<td><code>.</code></td>
<td>All</td>
</tr>
<tr>
<td>QGIS_PLUGINPATH</td>
<td>Useful if you are using Python plugins for the server, this sets the folder that is searched for Python plugins.</td>
<td><code>.</code></td>
<td>All</td>
</tr>
<tr>
<td>Pavadinimas</td>
<td>Aprašys</td>
<td>Default</td>
<td>Services</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>QGIS_PROJECT_FILE</td>
<td>The <code>.qgs</code> or <code>.qgz</code> project file, normally passed as a parameter in the query string (with MAP), you can also set it as an environment variable (for example by using <code>mod_rewrite</code> Apache module). Note that you may also indicate a project stored in PostgreSQL, e.g. <code>postgresql://localhost:5432?sslmode=disable&amp;dbname=mydb&amp;schema=myschema&amp;project=myproject</code>.</td>
<td>;</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_ALLOWED_EXTRA_SQL_TOKENS</td>
<td>Comma separated list of strings that represent the allowed extra SQL tokens accepted as components of a feature filter.</td>
<td>;</td>
<td>WMS</td>
</tr>
<tr>
<td>QGIS_SERVER_API_RESOURCES_DIRECTORY</td>
<td>Base directory for all OGC API (such as OAPIF/WFS3) static resources (HTML templates, CSS, JS, …)</td>
<td>depends on packaging</td>
<td>OAPIF/WFS3</td>
</tr>
<tr>
<td>QGIS_SERVER_API_WFS3_MAX_LIMIT</td>
<td>Maximum value for <code>limit</code> in a OAPIF/WFS3 features request.</td>
<td>10000</td>
<td>OAPIF/WFS3</td>
</tr>
<tr>
<td>QGIS_SERVER_CACHE_DIRECTORY</td>
<td>Specifies the network cache directory on the filesystem.</td>
<td>cache in profile directory</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_CACHE_SIZE</td>
<td>Sets the network cache size in MB.</td>
<td>50 MB</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_DISABLE_GETPRINT</td>
<td>This is an option at the project level to improve project read time by disabling loading of layouts. Activating this option disables the QGIS WMS GetPrint request. Set this QGIS project flag to not load layouts.</td>
<td>false</td>
<td>WMS</td>
</tr>
<tr>
<td>QGIS_SERVER_FORCE_READONLY_LAYERS</td>
<td>Force QGIS Server to open all layers in a read only mode</td>
<td>false</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_IGNORE_BAD_LAYERS</td>
<td>„Bad“ layers are layers that cannot be loaded. The default behavior of QGIS Server is to consider the project as not available if it contains a bad layer. The default behavior can be overridden by setting this variable to 1 or <code>true</code>. In this case, „bad“ layers will just be ignored, and the project will be considered valid and available.</td>
<td>false</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_LANDING_PAGE_PREFIX</td>
<td>Prefix of the path component of the landing page base URL</td>
<td>„“</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_LANDING_PAGE_PROJECTS_DIRECTORIES</td>
<td>Directories used by the landing page service to find <code>.qgs</code> and <code>.qgz</code> projects</td>
<td>„“</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_LANDING_PAGE_PROJECTS_PG_CONNECTIONS</td>
<td>PostgreSQL connection strings used by the landing page service to find projects</td>
<td>„“</td>
<td>All</td>
</tr>
<tr>
<td>continues on next page</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Skyrius 6. Advanced configuration
### Table 6.1 – tęsinys iš praeito puslapio

<table>
<thead>
<tr>
<th>Pavadinimas</th>
<th>Aprašymas</th>
<th>Default</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>QGIS_SERVER_LOG_FILE</td>
<td>Specify path and filename. Make sure that server has proper permissions for writing to file. File should be created automatically, just send some requests to server. If it’s not there, check permissions.</td>
<td>:’:</td>
<td>All</td>
</tr>
<tr>
<td><strong>Ispėjimas:</strong></td>
<td>QGIS_SERVER_LOG_FILE is deprecated since QGIS 3.4, use QGIS_SERVER_LOG_STDERR instead. File logging support will be removed in QGIS 4.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QGIS_SERVER_LOG_LEVEL</td>
<td>Specify desired log level. Available values are:</td>
<td>0</td>
<td>All</td>
</tr>
</tbody>
</table>
|                                   | • 0 or INFO (log all requests)  
• 1 or WARNING  
• 2 or CRITICAL (log just critical errors, suitable for production purposes)                                                    |         |          |
| QGIS_SERVER_LOG_PROFILE           | Add detailed profile information to the logs, only effective when QGIS_SERVER_LOG_LEVEL=0                                                                                                              | false   | All      |
| QGIS_SERVER_LOG_STDERR            | Activate logging to stderr. This variable has no effect when QGIS_SERVER_LOG_FILE is set.                                                                                                             | false   | All      |
|                                   | • 0 or false (case insensitive)  
• 1 or true (case insensitive)                                                      |         |          |
| QGIS_SERVER_MAX_THREADS           | Number of threads to use when parallel rendering is activated. If value is -1 it uses the number of processor cores.                                                                                   | -1      | All      |
| QGIS_SERVER_OVERRIDE_LOCALE       | Sets LOCALE to be used by QGIS server. The default value is empty (no override). Example: de_CH.utf8                                                                                                           | :’:     | All      |

continues on next page
### Table 6.1 – tęsinys iš praeito puslapio

<table>
<thead>
<tr>
<th>Pavadinimas</th>
<th>Aprašymas</th>
<th>Default</th>
<th>Services</th>
</tr>
</thead>
</table>
| QGIS_SERVER_PARALLEL_RENDERING | Activates parallel rendering for WMS GetMap requests. It's disabled (false) by default. Available values are:  
- 0 or false (case insensitive)  
- 1 or true (case insensitive) | false | WMS |
| QGIS_SERVER_PROJECT_CACHE_CHECK_INTERVAL | Controls the periodic strategy interval for cache invalidation | filesystem | All |
| QGIS_SERVER_PROJECT_CACHE_STRATEGY | Defines method for invalidating the project cache. Available strategies are:  
- filesystem: uses the file system watcher strategy  
- periodic: uses the last modified value of a project for checking changes on project configuration. Convenient on atypical file systems, such as NFS, or when the project file is stored in a database system like PostgreSQL  
- off: disables internal cache invalidation completely | filesystem | All |
| QGIS_SERVER_SERVICE_URL | This is an option to set the service URL if it is not present in the project. The service URL is defined from (in order of precedence):  
- Value defined in the project per service  
- The QGIS_SERVER_<service>_SERVICE_URL environment variable  
- The QGIS_SERVER_SERVICE_URL environment variable  
- The X-Qgis-<service>-Service-Url header  
- The X-Qgis-<service>-Service-Url header  
- Build from the Forwarded header  
- Build from the X-Forwarded-Host and X-Forwarded-Proto headers  
- Build from the Host header and the server protocol  
- Build from the server name and the server protocol. In the last four cases, the resulting Service URL is based on the MAP parameter provided in the query string and on the incoming path request. | ; | All |

continues on next page
<table>
<thead>
<tr>
<th>Pavadinimas</th>
<th>Aprašymas</th>
<th>Default</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>QGIS_SERVER_SHOW_GROUP_SEPARATOR</td>
<td>Defines whether a group separator (e.g. thousand separator) should be used for numeric values (e.g. in GetFeatureInfo responses). The default value is 0.</td>
<td>false</td>
<td>WMS</td>
</tr>
<tr>
<td>QGIS_SERVER_TRUST_LAYER_METADATA</td>
<td>This is an option at the project level to improve project read time by using the vector layer extents defined in the project metadata and disabling the check for PostgreSQL/PostGIS layer primary key uniqueness. Trusting layer metadata can be forced by setting this variable to 1 or true. The vector layer’s extent will then be the one defined in the project, and the PostgreSQL/PostGIS layer's primary key defined in the data source is considered as unique without a check. Do not use it if layers' extent is not fixed during the project’s use.</td>
<td>false</td>
<td>All</td>
</tr>
<tr>
<td>QGIS_SERVER_WCS_SERVICE_URL</td>
<td>This is an option to set the service URL if it is not present in the project. See QGIS_SERVER_SERVICE_URL for more information.</td>
<td>;</td>
<td>WCS</td>
</tr>
<tr>
<td>QGIS_SERVER_WFS_SERVICE_URL</td>
<td>This is an option to set the service URL if it is not present in the project. See QGIS_SERVER_SERVICE_URL for more information.</td>
<td>;</td>
<td>WFS</td>
</tr>
<tr>
<td>QGIS_SERVER_WMS_MAX_HEIGHT / QGIS_SERVER_WMS_MAX_WIDTH</td>
<td>Maximum height/width for a WMS request. The most conservative between this and the project one is used. If the value is -1, it means that there is no maximum set.</td>
<td>-1</td>
<td>WMS</td>
</tr>
<tr>
<td>QGIS_SERVER_WMS_SERVICE_URL</td>
<td>This is an option to set the service URL if it is not present in the project. See QGIS_SERVER_SERVICE_URL for more information.</td>
<td>;</td>
<td>WMS</td>
</tr>
<tr>
<td>QGIS_SERVER_WMTS_SERVICE_URL</td>
<td>This is an option to set the service URL if it is not present in the project. See QGIS_SERVER_SERVICE_URL for more information.</td>
<td>;</td>
<td>WMTS</td>
</tr>
</tbody>
</table>

continues on next page
### 6.3 Settings summary

When QGIS Server is starting, you have a summary of all configurable parameters thanks to environment variables. Moreover, the value currently used and the origin is also displayed.

For example with spawn-fcgi:

```bash
export QGIS_OPTIONS_PATH=/home/user/.local/share/QGIS/QGIS3/profiles/default/
export QGIS_SERVER_LOG_STDERR=1
export QGIS_SERVER_LOG_LEVEL=2
spawn-fcgi -f /usr/lib/cgi-bin/qgis_mapserv.fcgi -s /tmp/qgisserver.sock -U www-
         -G www-data -G www-data -n

QGIS Server Settings:

- `QGIS_OPTIONS_PATH` / '/' (Override the default path for user configuration): '/
  home/user/.local/share/QGIS/QGIS3/profiles/default/' (read from ENVIRONMENT_VARIABLE)

- `QGIS_SERVER_PARALLEL_RENDERING` / '/qgis/parallel_rendering' (Activate/Deactivate parallel rendering for WMS getMap request): 'true' (read from INI_FILE)

- `QGIS_SERVER_MAX_THREADS` / '/qgis/max_threads' (Number of threads to use when parallel rendering is activated): '4' (read from INI_FILE)

- `QGIS_SERVER_LOG_LEVEL` / '/' (Log level): '2' (read from ENVIRONMENT_VARIABLE)

- `QGIS_SERVER_LOG_STDERR` / '/' (Activate/Deactivate logging to stderr): '1'

- `QGIS_PROJECT_FILE` / '/' (QGIS project file): '' (read from DEFAULT_VALUE)

- `MAX_CACHE_LAYERS` / '/' (Specify the maximum number of cached layers): '100'
```

(continues on next page)
In this particular case, we know that `QGIS_SERVER_MAX_THREADS` and `QGIS_SERVER_PARALLEL_RENDERING` values are read from the ini file found in `QGIS_OPTIONS_PATH` directory (which is defined through an environment variable). The corresponding entries in the ini file are `/qgis/max_threads` and `/qgis/parallel_rendering` and their values are `true` and `4` threads.

### 6.4 Connection to service file

In order to make apache aware of the PostgreSQL service file (see the `pg-service-file` section) you need to make your `*.conf` file look like:

```bash
SetEnv PGSERVICEFILE /home/web/.pg_service.conf

<Directory "/home/web/apps2/bin/">
  AllowOverride None
  ......
</Directory>
```

### 6.5 Add fonts to your linux server

Keep in mind that you may use QGIS projects that point to fonts that may not exist by default on other machines. This means that if you share the project, it may look different on other machines (if the fonts don’t exist on the target machine).

In order to ensure this does not happen you just need to install the missing fonts on the target machine. Doing this on desktop systems is usually trivial (double clicking the fonts).

For linux, if you don’t have a desktop environment installed (or you prefer the command line) you need to:

- **On Debian based systems:**
  ```bash
  sudo su
  mkdir -p /usr/local/share/fonts/truetype/myfonts && cd /usr/local/share/fonts/truetype/myfonts
  # copy the fonts from their location
  cp /fonts_location/* .
  chown root *
  cd .. && fc-cache -f -v
  ```

- **On Fedora based systems:**
  ```bash
  sudo su
  mkdir /usr/share/fonts/myfonts && cd /usr/share/fonts/myfonts
  # copy the fonts from their location
  cp /fonts_location/* .
  ```

(continues on next page)
chown root *
cd .. && fc-cache -f -v
A production installation and deployment of QGIS Server usually involves setting up a web server component (e.g. Apache or Nginx) that can forward the HTTP requests coming from the clients to the QGIS Server FastCGI binary application.

If you want to quickly test QGIS Server on your local machine without configuring and installing a full web server stack you can use the QGIS Development Standalone server. This is an independent application that provides a very simple web server ready to serve your project files.

**Ispėjimas:** The Standalone Development Server has not been developed with the purpose of being used in production, it was not checked for security vulnerabilities or for other stress conditions that normally will occur on a publicly exposed server.

To launch the server:

```
$ qgis_mapserver
```

The default port the Development Server listens to is 8000. Example output:

```
QGIS Development Server listening on http://localhost:8000
CTRL+C to exit
127.0.0.1 [lun gen 20 15:16:41 2020] 5140 103ms "GET /wfs3/?MAP=/tests/testdata/qgis_server/test_project.qgs HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 3298 2ms "GET /wfs3/static/jsonFormatter.min.js HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 1678 3ms "GET /wfs3/static/jsonFormatter.min.css HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:41 2020] 1310 5ms "GET /wfs3/static/style.css HTTP/1.1" 200
127.0.0.1 [lun gen 20 15:16:43 2020] 4285 13ms "GET /wfs3/collections?MAP=/tests/testdata/qgis_server/test_project.qgs HTTP/1.1" 200
```

The server has a few options that can be passed as command line arguments. You can see them all by invoking the server with \(-h\).

Usage: qgis_mapserver [options] [address:port]

QGIS Development Server

(continues on next page)
Options:
- `h, --help` Displays this help.
- `v, --version` Displays version information.
- `l <logLevel>` Sets log level (default: 0)
  - 0: INFO
  - 1: WARNING
  - 2: CRITICAL
- `p <projectPath>` Path to a QGIS project file (*.qgs or *.qgz),
  if specified it will override the query string MAP argument
  and the QGIS_PROJECT_FILE environment variable

Arguments:
- `addressAndPort` Listen to address and port (default: "localhost:8000")
  address and port can also be specified with the environment
  variables QGIS_SERVER_ADDRESS and QGIS_SERVER_PORT
Containerized deployment

There are many ways to use containerized application, from the most simple (simple Docker images) to sophisticated (Kubernetes and so on).

**Pastaba:** This kind of deployment needs the docker application to be installed and running. Check this [tutorial](#).

**Patarimas:** Docker run pre packaged application (aka images) which can be retrieved as sources (Dockerfile and resources) to build or already built from registries (private or public).

**Pastaba:** QGIS Debian-Ubuntu package downloads need a valid gpg authentication key. Please refer to the [installation pages](#) to update the following Dockerfile.

### 8.1 Simple docker images

As the docker image does not exist in a public registry, you will need to build it. To do so create a directory `qgis-server` and within its directory:

- create a file `Dockerfile` with this content:

```bash
FROM debian:bullseye-slim
ENV LANG=en_EN.UTF-8

RUN apt-get update \
  && apt-get install --no-install-recommends --no-install-suggests --allow-unauthenticated -y \
  gnupg \n  ca-certificates \n  wget \n  locales \
  && localedef -i en_US -f UTF-8 en_US.UTF-8 \
```

(continues on next page)
# Add the current key for package downloading
# Please refer to QGIS install documentation (https://www.qgis.org/fr/site/

```bash
mkdir -m755 -p /etc/keyrings
wget -O /etc/apt/keyrings/qgis-archive-keyring.gpg https://download.qgis.org/downloads/qgis-archive-keyring.gpg
```

# Add repository for latest version of qgis-server
# Please refer to QGIS repositories documentation if you want other version
```bash
echo "deb [signed-by=/etc/apt/keyrings/qgis-archive-keyring.gpg] https://qgis.org/debian bullseye main" | tee /etc/apt/sources.list.d/qgis.list
```

```bash
RUN useradd -m qgis
```

```bash
ENV TINI_VERSION v0.19.0
```

```bash
ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/tini /tini
```

```bash
RUN chmod +x /tini
```

```bash
ENV QGIS_PREFIX_PATH /usr
```

```bash
ENV QGIS_SERVER_LOG_STDERR 1
```

```bash
ENV QGIS_SERVER_LOG_LEVEL 2
```

```bash
COPY cmd.sh /home/qgis/cmd.sh
```

```bash
RUN chmod -R 777 /home/qgis/cmd.sh
```

```bash
RUN chown qgis:qgis /home/qgis/cmd.sh
```

```bash
USER qgis
```

```bash
WORKDIR /home/qgis
```

```bash
ENTRYPOINT ["/tini", "--"
```

```bash
CMD ["/home/qgis/cmd.sh"]
```

- create a file `cmd.sh` with this content:

```bash
#!/bin/bash
[[ $DEBUG = "1" ]] && env
exec /usr/bin/xvfb-run --auto-servernum --server-num=1 /usr/bin/spawn-fcgi -p 5555 -n -d /home/qgis -- /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

- build the image with:

```bash
docker build -f Dockerfile -t qgis-server ./
```
8.1.1 First run

To run the server you will need a QGIS project file. You can use one of yours or pick this sample.

To do so, create a directory `data` within the directory `qgis-server` and copy your file in it. To comply with the following explanations, rename it to `osm.qgs`.

Pastaba: You may need to add advertised URLs under the QGIS Server tab of the Project ➤ Properties if the GetCapabilities are broken. For example if your server is exposed on port 8080, you will put this for advertised URL http://localhost:8080/qgis-server/. More information available in section Configure your project and subsequent.

Now, you can run the server with:

```
docker network create qgis

docker run -d --rm --name qgis-server --net=qgis --hostname=qgis-server \
  -v $(pwd)/data:/data:ro -p 5555:5555 \
  -e "QGIS_PROJECT_FILE=/data/osm.qgs" \
  qgis-server
```

Options used:

- `-d`: run in the background
- `--rm`: remove the container when it is stopped
- `--name`: name of the container to be created
- `--net`: (previously created) sub network
- `--hostname`: container hostname, for later referencing
- `-v`: local data directory to be mounted in the container
- `-p`: host/container port mapping
- `-e`: environment variable to be used in the container

To check, type `docker ps | grep qgis-server` and you should see a line with `qgis-server`:

```none
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS         PORTS NAMES
--------------- ----------- --------------- --------------- ------------- --------------- ---------------
4de8192da76e   qgis-server   "/tini -- /home/qgis..." 3 seconds ago Up 2 seconds...
```

8.1.2 Usable sample

As the server is only accepting fastcgi connections, you need an HTTP server that handles this protocol. To do so we have to create a simple Nginx configuration file and start a Nginx image.

Create a file `nginx.conf` in the current directory with this content:

```
server {
    listen 80;
    server_name _;
    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
    }
    location /qgis-server {
        proxy_buffers 16 16k;
        proxy_buffer_size 16k;
        gzip off;
    }
}
```

(continues on next page)
include fastcgi_params;
fastcgi_pass qgis-server:5555;
}
}

And type this command:

docker run -d --rm --name nginx --net=qgis --hostname=nginx \
- v ${pwd}/nginx.conf:/etc/nginx/conf.d/default.conf:ro -p 8080:80 \
nginx:1.13

To check capabilities availability, type in a browser http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities

8.1.3 Cleanup

To cleanup the running images, type:

docker stop qgis-server nginx

8.2 Docker stacks

The previous method is scriptable, but not easily packageable nor standardized or easily manageable.

To work with a docker image set you could use a docker stack managed by an orchestrator. In a stack, the images are working in the same private network, and you can start / stop the whole stack or deploy the stack to other workers. There are many orchestrators, for example Swarm, Kubernetes and Mesos.

In the following, we will present simple configurations for testing purposes. They are not suitable for production.

8.2.1 Swarm/docker-compose

Docker now has its own orchestrator: Swarm (compatible with docker-compose files). You have to enable it (the Mac version will also work with Linux).

Stack description

Now that you have Swarm working, create the service file (see Deploy to Swarm) qgis-stack.yaml:

```
version: '3.7'

services:
  qgis-server:
    # Should use version with utf-8 locale support:
    image: qgis-server:latest
    volumes:
    - REPLACE_WITH_FULL_PATH/data:/data:ro
    environment:
    - LANG=en_EN.UTF-8
    - QGIS_PROJECT_FILE=/data/osm.qgs
    - QGIS_SERVER_LOG_LEVEL=0 # INFO (log all requests)
    - DEBUG=1 # display env before spawning QGIS Server
  nginx:
    image: nginx:1.13

(continues on next page)```
To deploy (or update) the stack, type:

```
docker stack deploy -c qgis-stack.yaml qgis-stack
```

Check the stack deployment status until you obtain 1/1 in the `replicas` column:

```
docker stack services qgis-stack
```

Something like:

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>MODE</th>
<th>REPLICAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>gmx7ewlvwsqt</td>
<td>nginx:1.13</td>
<td>replicated</td>
<td>1/1</td>
</tr>
<tr>
<td>l0v2e7c143u3</td>
<td>qgis-qgis-server:latest</td>
<td>replicated</td>
<td>1/1</td>
</tr>
</tbody>
</table>

To check WMS capabilities, type in a web browser `http://localhost:8080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities`

## Cleanup

To cleanup, type:

```
docker stack rm qgis-stack
```

### 8.2.2 Kubernetes

#### Įdiegimas

If you have a Docker Desktop installation, using Kubernetes (aka k8s) is pretty straightforward: enable k8s.

If not, follow the minikube tutorial or microk8s for Ubuntu.

As Kubernetes installation can be really complex, we will only focus on aspects used by this demo. For further / deeper information, check the official documentation.

#### microk8s

microk8s needs extra steps: you have to enable the registry and tag the qgis-server image in order to have Kubernetes to find the created images.

First, enable the registry:

```
microk8s enable dashboard dns registry
```

Then, tag and push the image to your newly created registry:

```
docker tag qgis-server 127.0.0.1:32000/qgis-server && docker push 127.0.0.1:32000/qgis-server
```
Finally, add or complete the `/etc/docker/daemon.json` to have your registry `127.0.0.1:32000` listed in the `insecure-registries` field:

```json
{
    "insecure-registries": ["127.0.0.1:32000"]
}
```

### Creating manifests

Kubernetes describes the objects to deploy in yaml manifests. There are many different kinds, but we will only use deployments (handle pods, i.e. docker images) and services to expose the deployments to internal or external purposes.

### Deployment manifests

Create a file `deployments.yaml` with this content:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qgis-server
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-server
  template:
    metadata:
      labels:
        myLabel: qgis-server
    spec:
      containers:
      - name: qgis-server
        image: localhost:32000/qgis-server:latest
        imagePullPolicy: IfNotPresent
        env:
          - name: LANG
            value: en_EN.UTF-8
          - name: QGIS_PROJECT_FILE
            value: /data/osm.qgs
          - name: QGIS_SERVER_LOG_LEVEL
            value: "0"
          - name: DEBUG
            value: "1"
        ports:
        - containerPort: 5555
        volumeMounts:
        - name: qgis-data
          mountPath: /data/
      volumes:
      - name: qgis-data
        hostPath:
          path: REPLACE_WITH_FULL_PATH/data
---
apiVersion: apps/v1
kind: Deployment
metadata:
```

(continues on next page)
```yaml
name: qgis-nginx
namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      myLabel: qgis-nginx
  template:
    metadata:
      labels:
        myLabel: qgis-nginx
    spec:
      containers:
      - name: qgis-nginx
        image: nginx:1.13
        ports:
        - containerPort: 80
      volumeMounts:
      - name: nginx-conf
        mountPath: /etc/nginx/conf.d/default.conf
      volumes:
      - name: nginx-conf
        hostPath:
          path: REPLACE_WITH_FULL_PATH/nginx.conf
```

### Service manifests

Create a file `services.yaml` with this content:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: qgis-server
  namespace: default
spec:
  type: ClusterIP
  selector:
    myLabel: qgis-server
  ports:
  - port: 5555
    targetPort: 5555
---

apiVersion: v1
kind: Service
metadata:
  name: qgis-nginx
  namespace: default
spec:
  type: NodePort
  selector:
    myLabel: qgis-nginx
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30080
```

8.2. Docker stacks
Deploying manifests

To deploy the images and services in Kubernetes, one can use the dashboard (click on the + on the upper right) or the command line.

**Pastaba:** When using the command line with microk8s you will have to prefix each command with `microk8s`.

To deploy or update your manifests:

```
kubectl apply -k ./
```

To check what is currently deployed:

```
kubectl get pods,services,deployment
```

You should obtain something like:

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pod/qgis-nginx-54845ff6f6-8skp9</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>27m</td>
</tr>
<tr>
<td>pod/qgis-server-75df6dd89-c7t7s</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>27m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service/Kubernetes</td>
<td>ClusterIP</td>
<td>10.152.183.1</td>
<td>&lt;none&gt;</td>
<td>443/TCP</td>
<td>5h51m</td>
</tr>
<tr>
<td>service/qgis-exec-server</td>
<td>ClusterIP</td>
<td>10.152.183.218</td>
<td>&lt;none&gt;</td>
<td>5555/TCP</td>
<td>35m</td>
</tr>
<tr>
<td>service/qgis-nginx</td>
<td>NodePort</td>
<td>10.152.183.234</td>
<td>&lt;none&gt;</td>
<td>80:30080/TCP</td>
<td>27m</td>
</tr>
<tr>
<td>service/qgis-server</td>
<td>ClusterIP</td>
<td>10.152.183.132</td>
<td>&lt;none&gt;</td>
<td>5555/TCP</td>
<td>27m</td>
</tr>
</tbody>
</table>

To read nginx/qgis logs, type:

```
kubectl logs -f POD_NAME
```

To check WMS capabilities, type in a web browser `http://localhost:30080/qgis-server/?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities`

**Cleanup**

To clean up, type:

```
kubectl delete -n default service/qgis-server service/qgis-nginx deployment/qgis-nginx deployment/qgis-server
```
8.3 Cloud deployment

Managing your own cluster of servers to handle the deployment of containerized applications, is a complex job. You have to handle multiple issues, such as hardware, bandwidth and security at different levels.

Cloud deployment solutions can be a good alternative when you do not want to focus on infrastructure management.

A cloud deployment may use proprietary mechanisms, but they are also compatible with the stages explained previously (docker images and stack management).

8.3.1 AWS usecase

With Amazon AWS, through ECS (Elastic Container Service) functionalities, you can use docker-compose or Kubernetes compatible wrappers to manage your stack. You will have to create an image registry for your custom images to be accessible.

To use docker-compose alike functionalities, you need to install the ecs-cli client and have proper permissions / roles. Then, with the help of the ecs-cli compose commands, you can reuse the stack description.

To use Kubernetes, you can use the AWS web console or the command line tool eksctl and have the proper permissions / roles. Then with a well configured kubectl environment, you can reuse the Kubernetes manifests.
Frequently Asked Question

- **What are the differences between QGIS Desktop and QGIS Server?**
  
  QGIS Desktop has a graphical user interface and allows you to create and modify maps. QGIS Server is a server application serving your QGIS project files to end user applications via OGC web services like WMS, WFS, etc.

- **What is OGC?**
  
  The [OGC (Open Geospatial Consortium)](https://www.opengeospatial.org) is an international not for profit organization committed to making quality open standards for the global geospatial community.

- **Name some other web mapping servers?**
  
  ArcGIS server, GeoServer, Mapserver, Mapnik etc.

- **How to compare QGIS server to other web mapping servers?** (2021/01/01)

<table>
<thead>
<tr>
<th>Savybės</th>
<th>QGIS Server</th>
<th>GeoServer</th>
<th>ArcGIS Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2006</td>
<td>2001</td>
<td>1999</td>
</tr>
<tr>
<td>Licence</td>
<td>GPL</td>
<td>GPL</td>
<td>commercial</td>
</tr>
<tr>
<td>Commercial support</td>
<td>Multiple companies</td>
<td>Multiple companies</td>
<td>ESRI and its vendors network</td>
</tr>
<tr>
<td>Technology</td>
<td>C++/python</td>
<td>Java</td>
<td>C++</td>
</tr>
<tr>
<td>Tile cache</td>
<td>yes</td>
<td>yes (via GeoWebCache)</td>
<td>yes</td>
</tr>
<tr>
<td>3D</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Querying</td>
<td>FES (2.0) and OGC (1.0) filters</td>
<td>CQL and OGC filters</td>
<td>OGC filters</td>
</tr>
<tr>
<td>Report generation</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Server administration</td>
<td>yes via third parties (LizMap, QWC2, etc.)</td>
<td>web + API REST</td>
<td>web + API REST</td>
</tr>
<tr>
<td>GIS project</td>
<td>complete via dedicated GUI</td>
<td>simple via web interface</td>
<td>complete via dedicated GUI</td>
</tr>
<tr>
<td>Layer/symbology edition</td>
<td>complete via dedicated GUI</td>
<td>simple via web interface</td>
<td>complete via dedicated GUI</td>
</tr>
</tbody>
</table>

- **What are the OGC specification versions implemented in QGIS server compared to other web mapping servers?** (2021/01/01)
**What is a tile cache?**

Maps are often static. As most mapping clients render WMS (Web Map Service) data every time they are queried, this can result in unnecessary processing and increased wait times.

The tile cache optimizes this experience by saving (caching) map images, or tiles, as they are requested, in effect acting as a proxy between client (such as OpenLayers or Google Maps) and server (any WMS-compliant server). As new maps and tiles are requested, QGIS server intercepts these calls and returns pre-rendered tiles if stored, or calls the QGIS engine to render new tiles as necessary. Thus, once tiles are stored, the speed of map rendering increases by many times, creating a much improved user experience.

**What is PostgreSQL?**

PostgreSQL is a powerful, open source object-relational database companion for QGIS.

**What is PostGIS?**

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.