# MapFish Print Documentation

***Release 2.0-SNAPSHOT***

**Camptocamp**

December 09, 2013

# Contents

MapFish Print allows printing maps as PDFs.

MapFish Print is written in Java. It is typically executed as a servlet in a servlet container such as Apache Tomcat.

MapFish Print is released under the GPLv3 license.

GeoExt includes print components for use with MapFish Print.

Contents:

CHAPTER 1

# Installation

## 1.1 Compilation

To build this project, you need:

- JDK >= 1.5, make sure you use Sun's JDK (sun-java6-jdk on Ubuntu 8.04) and not GCJ
- Download MapFish Print from GitHub
  - On systems with git installed:

```
Go to https://github.com/mapfish/mapfish-print
Click the "Fork" button to fork a version to your user
Assuming your username is myuser type

git clone git@github.com:myuser/mapfish-print.git
```

The other external libs are taken care of by maven.

The build command to use is:

- On Linux:

```
cd %MAPFISH_DIR%/trunk/
./gradlew build
```

- On Windows:

```
cd %MAPFISH_DIR%/trunk/
gradlew.bat build
```

You might get an error like:

```
....
.....
FAILURE: Could not determine which tasks to execute.

* What went wrong:

Task 'build' not found in root project 'src'.

* Try:
```

```
Run with -t to get a list of available tasks.

BUILD FAILED
....
.....
```

Then try:

```
cd %MAPFISH_DIR%/trunk/
./gradlew clean build
```

## 1.2 Print module installation

The PDF generation requires some code to run on your web server. This page describes how to install it.

Depending on what language/framework you want to use on your server and what performance you need, multiple solutions are available. The next sections will describe them.

In all cases, **you have to modify the print.yaml file** to customize it. At least to add the list of hosts that can be used to get the maps from.

### 1.2.1 Python

1. Edit the development.ini file and add two lines like that to the [DEFAULT] section (change the strings "%....%" by something relevant):

   ```
   print.jar = %MAPFISHDIR%/trunk/build/libs/print-standalone-%VERSION%.jar
   print.config = %PROJECT_DIR%/print/config.yaml
   ```

2. Edit the config.yaml file in your %PROJECT_DIR%/print/ directory as documented in the Print Module Server page.

3. Edit the MY_PROJECT/config/routing.py file and add those 2 lines (just after the "CUSTOM ROUTES HERE"):

   ```
   from mapfish.controllers import printer
   printer.addRoutes(map, '/print/', 'printer')
   ```

4. Create a MY_PROJECT/controllers/printer.py file with this single line as content:

   ```
   from mapfish.controllers.printer import PrinterController
   ```

5. If you use Apache mod_wsgi, you can add the following before the "loadapp" call in the WSGI script:

   ```
   # configure the logging system
   from paste.script.util.logging_config import fileConfig
   fileConfig('%PASTE_CONFIG%')
   ```

   where `%%PASTE_CONFIG%` is the path to the application's configuration file (.ini).

### 1.2.2 Java servlet

This method will give you the best performance.

### The separate WAR way

For that you need to install tomcat5.5 or higher.

Note: some people have experienced problems when deploying the MapFish Print WAR in the Tomcat packaged in Ubuntu, it looks like these problems are due to additional security constraints in the Ubuntu package. If you have such problems and don't know how to solve them you can download Tomcat from the Apache web site, for example MapFish Print is known to work well with apache-tomcat-6.0.20.tar.gz.

In my system, tomcat is installed in **/var/lib/tomcat5.5** and the %WEBAPPS% directory is in **/var/lib/tomcat5.5/webapps**.

1. copy the WAR file in your %WEBAPPS% directory. This is can be obtained from *%MAPFISH_DIR%/trunk/build/libs/print-servlet-*.war* if you compiled the print module or you can get it from the maven repository.
2. Edit the config.yaml file in your *%WEAPPS%/print-servlet-*/* directory as documented in the Print Module Server page.
3. If needed, edit the web.xml and log4j.properties files.

If your web pages are not served by the tomcat server, you may need to configure a proxy to work around the same origin policy. See Configure Proxy for examples on how you may do that.

### Included into another web application

If you are developping a web application, you can include the print module in it. This explanation works only if you use Maven to build your project. For others, you'll have to include the print-lib JAR file and all its dependencies.

1. Add the print-lib dependency to your WAR project by adding these two elements to your pom.xml file (adjust the version):

```
...
<dependencies>
  <dependency>
    <groupId>org.mapfish.print</groupId>
    <artifactId>print-lib</artifactId>
    <version>%RELEASE%</version>
  </dependency>
  ...
</dependencies>
...
<repositories>
  <repository>
    <id>geotools</id>
    <url>http://download.osgeo.org/webdav/geotools</url>
  </repository>
         <!-- The next repository is only required if the release is a snapshot -->
  <repository>
    <id>maven snapshots</id>
    <url>https://oss.sonatype.org/content/repositories/snapshots/</url>
    <snapshots>
                        <enabled>true</enabled>
                </snapshots>
  </repository>
  ...
</repositories>
...
```

2. Add these three sections (correct the paths and URLs in function of your needs) to your WEB-INF/web.xml file:

```
...
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:mapfish-spring-application-context.xml,classpath:*-mapfish-spring-app
</context-param>
...
<servlet>
  <servlet-name>mapfish.print</servlet-name>
  <servlet-class>org.mapfish.print.servlet.MapPrinterServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>config.yaml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>mapfish.print</servlet-name>
  <url-pattern>/pdf/*</url-pattern>
</servlet-mapping>
 ...
```

3. Add a config.yaml file (from the samples, for example) to the root of your web application.

#### Configuring Servlet Temp Directory

By default the default servlet temporary directory will be used but that behaviour can be overridden setting the init-param *tempdir*. If this parameter is set the servlet must have write access to the directory.

### 1.2.3 Command line

For debugging or calling from other environments, you can run the print module from the command line.

Examples are assumed to be ran from the directory %MAPFISHDIR%/server/java/print.

The fastest way is to call to gradle to run the application.

```
./gradlew run -Dconfig=samples/config.yaml -Dspec=samples/spec.json -Doutput=/tmp/print-out.pdf
```

Help about the command line's option can be obtained like that:

./gradlew run

If you have previously built you can run the existing jar using the following commands:

```
java -Djava.awt.headless=true -cp build/libs/print-standalone-%VERSION%.jar org.mapfish.print.ShellMa
```

Help about the command line's option can be obtained like that:

```
java -Djava.awt.headless=true -cp build/libs/print-standalone-%VERSION%.jar org.mapfish.print.ShellMa
```

### 1.2.4 With Image output line

At the moment to have robust image (png, gif, etc...) output options the image magick command line tool must be used. The tool is cross platform and available on linux, windows and mac osx. Before installing mapfish print first

install both:

- ImageMagick (specifically ensure that the convert tool has been installed)
- Ghostscript

Once ImageMagick is installed (and Ghostscript) then the spring configuration needs to be updated so that org.mapfish.print.output.NativeProcessOutputFactory object will be used as an option for creating output files. This OutputFactory uses native commandline processes for converting a PDF to an alternate format. The mapfish-spring-application-context.xml can be directly edited to configure the OutputFactory or the file https://github.com/mapfish/mapfish-print/blob/master/sample-spring/imagemagick/WEB-INF/classes/imagemagick-spring-application-context-override.xml can be copied into the mapfish-print/WEB-INF/classes folder (if a servlet is being used).

Since the ImageMagick support is a commonly requested configuration there is an IMAGEMAGICK artifact prebuild with the correct configuration. To use this artifact change your maven dependency from: print-servlet-1.2-SNAPSHOT to print-servlet-1.2-SNAPSHOT-IMG-MAGICK (you will likely have to change the 1.2-SNAPSHOT portion to the version of mapfish that you are using).

By default ImageMagick will try to find the convert tool in /usr/bin/convert. You will want to find the path to your convert tool and update imagemagick-spring-application-context-override.xml if that file is included in your build, or mapfish-spring-application-context.xml if you manually changed the mapfish-spring-application-context.xml file to include the imagemagick configuration.

CHAPTER 2

# Configuration

The server side uses a YAML configuration file that defines the page layouts and allowed values. This file is usually called config.yaml.

Here is the general structure:

```
dpis:
  - 254
  - 190
  {...}

?maxSvgWidth: 2048 # set the maximum dimensions to 2048 points, this is useful when using MapServer a
?maxSvgHeight: 2048
?integerSvg: false # the library in MapServer <= 5.6 does not support floating point values in the SV

?formats:
  - pdf
  - png
  {...}

scales:
  - 25000
  - 50000
  {...}

hosts:
  - {HOST_WHITELIST_DEFINITION}
  {...}

?localHostForward: # For request on map.example.com we build an http request on localhost with the he
?  from:
?    - map.example.com
?  https2http: True # For above hosts on request on https we build a request on http

?headers: ['Cookie', 'Referer'] # The header that will be copyed to the tiles http requests

?keys:
?  - !key
?    host: !dnsMatch
?      host: maps.google.com
?      port: 80
```

```
?    domain: !dnsMatch
?      host: localhost
?    key: 1234456
?    id: gmd-xyz


?fonts:
? - {PATH}

?globalParallelFetches: 5
?perHostParallelFetches: 5
?tilecacheMerging: false
?connectionTimeout: 30000           MF_V1.2
?socketTimeout: 180000              MF_V1.2
?outputFilename: Mapfish-print      MF_V1.2
?disableScaleLocking: false
?brokenUrlPlaceholder: default      MF_V2.0
?proxyBaseUrl: http://mapfishprint.org  MF_V2.0

?security:
?  - !basicAuth
?      matcher: !dnsMatch
?        host: www.camptocamp.com
?        post: 443
?      username: xyz
?      password: zyx
?      preemptive: true
?  - !basicAuth
?    username: abc
?    password: bca

layouts:
   {LAYOUT_NAME}:
?   : Mapfish-print.pdf  MF_V1.2
?   metaData:
?     {METADATA_DEFINITION}
?   titlePage:
?     {PAGE_DEFINITION}
    mainPage:
?     rotation: false
      {PAGE_DEFINITION}
?   lastPage:
?     {PAGE_DEFINITION}
  {...}
```

Optional parts are shown with a question mark in the left margin. The question marks must not be put in the configuration file. Their default values is shown.

Note: Sets of values like DPI can be entered in one of two forms: .. code-block:: yaml

dpi: [1,2,3,...]

or

```
dpis:
  - 254
  - 190
```

A chosen DPI value from the above configuration is used in WMS GetMap requests as an added format_options (GeoServer) or map_resolution (MapServer) parameter. This is used for symbol/label-rescaling suitable for high res-

olution printouts, see Geoserver format_options specification (Geoserver 2.1) and MapServer defresolution keyword (MapServer 5.6) for more information.

In general, PDF dimensions and positions are specified in points. 72 points == 1 inch == 25.4 mm.

The list of {HOST_WHITELIST_DEFINITION} defines the allowed URLs for getting maps. Its format will be defined in the next sub-section.

The formats element lists the values formats that the server permits. If omitted only 'pdf' is permitted. If the single element '*' (quotes are required) is present then all formats that the server can produce can be requested. The formats the server can produce depends to a large degree on how the Java is configured. PDF is supported on all systems but for image output formats JAI and ImageIO is used which means both must be on the server for them to be available. You can get the list of supported formats by running the standalone client with the –clientConfig flag enabled (you will need to supply a yaml config file as well). If you are using the servlet then do a get info request to see the list of formats (with the '*' as the outputFormats parameter in the config file).

You can have as many layouts as you want. Their name must be unique and will be used on the client side. A layout can have a "titlePage" that will be added at the beginning of the generated document. It cannot contain any map. Same for the "lastPage", but for the end of the document. The "mainPage" section is mandatory and will be used once for each page requested. The details of a {PAGE_DEFINITION} section can be found in another sub-section of this document.

If you want to let the user rotate the map (for a given layout), you have to set the "rotate" field to "true" in the corresponding "mainPage" section.

"globalParallelFetches" and "perHostParallelFetches" are used to tune the parallel loading of the map tiles/images. If you want to disable the parallel loading, set "globalParallelFetches" to 1.

New versions of tilecache added the support for merging multiple layers in a single WMS request. If you want to use this functionality, set the "tilecacheMerging" attribute to true.

"connectionTimeout" and "socketTimeout" (only since MapFish v1.2) can be used to tune the timeouts for reading tiles from map servers.

If the 'outputFilename' parameter is defined in the main body then that name will be used by the MapPrintServlet when sending the pdf to the client. It will be the name of the file that the client downloads. If the 'outputFilename' parameter is defined in a layout then that value will override the default name. In both cases the .pdf is optional; if not present the server will append .pdf to the name. In all cases the json request can override the filename defined in the configuration file by posting a 'outputFilename' attribute in the posted JSON. If the outputFilename has ${date}, ${time} or ${dateTime} in it, it will be replaced with the current date using the related DateFormat.get*Instance().format() method. If a pattern is provided it will be passed to SimpleDataFormat for processing. A few examples follow:

- outputFilename: "host-${yyyyMMdd}.pdf" # results in host-20111213.pdf
- outputFilename: "host-${date}" # results in host-Dec_13_2011.pdf (actual output depends on local of server)
- outputFilename: "host-${dateTime}" # results in host-Dec_13_2011_1:10:50_PM.pdf (actual output depends on local of server)
- outputFilename: "host-${time}.pdf" # results in host-1:11:14_PM.pdf (actual output depends on local of server)
- outputFilename: "host-${yyMMdd-hhmmss}"# results in host-111213-011154.pdf (actual output depends on local of server)

"disableScaleLocking" allows you to bypass the choosing of scale from the available factors, and simply use the suggested value produced inside MapBlock.java.

"brokenUrlPlaceholder" the placeholder image to use in the case of a broken url. By default, when a url request fails, an error is thrown and the pdf process terminates. However if this parameter is set then instead a placeholder image is returned. Non-null values are:

- "default" - use the system default image.

- "throw" - throw an exception.
- <url> - obtain the image from the supplied url. If this url is broken then an exception will be thrown. This can be anytype of valid url from a file url to https url.

"proxyBaseUrl" the optional url of the proxy between mapfish-print and the internet. This is the url base that will be in the info.json response. On occasion the url or port of the web server containing mapfish-print is not the server that is public to the internet and the requests are proxied to the mapfish-print webserver. In this case it is important for the info.json request to return the public URL instead of the url of the webserver.

## 2.1 Security

Both Keys and Security are options for accessing protected services. Keys are currently for Google maps premium accounts and Security is for other types and is more general Currently only BasicAuth is supported but other strategies can easily be added

```
security:
    - !basicAuth
        matcher: !dnsMatch
          host: www.camptocamp.com
          post: 443
        username: xyz
        password: zyx
        preemptive: true
    - !basicAuth
      username: abc
      password: cba
```

The above example has 2 security configuration. Each option is tested (in order) to see if it can be used for the given URI and if it applies it is used to configure requests for the URI. In the above example the first configuration will be used if the URI matches the hostmatcher provided if not then the second configuration will be applied. The last configuration has no host matcher so it is applied to all URIs.

A basicAuth security configuration consists of 4 options

- matcher - a host matcher for determining which requests need the security to be applied
- username - username for basicauth
- password - password for basicauth
- preemptive - optional, but for cases where the credentials need to be sent without the challenge

## 2.2 Keys

Google maps currently requires a private key to be used (we only support users Google maps premium accounts).

The keys section allows a key to be mapped to hosts. The hosts are identified with host matchers that are described in the <configuration.html#host-whitelist-definition> sub-section.

In addition a domain hostmatcher can be used to select a key based on the domain of the local server. This can be useful if the same configuration is used in a test environment and a production environment with differing domains. For example mapfish.org and mapfish.net.

Finally google maps (for example) requires a client id as well that is associated with the private key. There for in the case of google premium services a legal key would be:

```
keys:
  - !key
    key: yxcvyxvcyxvyx
    id: gme-xxxcs
```

Thanks to the hosts and domain matcher it is possible to have a key for google maps and (for future proofing) a different key for a different service.

## 2.3 Fonts definition

The "fonts" section is optional. It contains the path of the fonts you want to use. The entries can point to files (TTF, OTF, TTC, AFM, PFM) or directories. Don't point to directories containing too many files since it will slow down the start time. By default, PDF gives you access to the following fonts (Cp1252 encoding only):

- Courrier (-Bold, -Oblique, -BoldOblique)
- Helvetica (-Bold, -Oblique, -BoldOblique)
- Times (-Roman, -Bold, -Oblique, -BoldOblique)
- Symbol
- ZapfDingbats

## 2.4 Host whitelist definition

In this section, you can put as many entries as you want, even for the same type of filter. If at least one matches, the Map server can be used.

This section is not for defining which client can request maps. It is just here to avoid having the print module used as a proxy to access documents from computers behind firewalls.

There are 3 ways to whitelist a host.

### 2.4.1 Allowing every local services:

```
- !localMatch
  dummy: true
```

The "dummy" parameter is ignored, but mandatory to avoid a limitation in the YAML format.

### 2.4.2 Allowing by DNS name:

```
- !dnsMatch
  host: labs.metacarta.com
```

### 2.4.3 Allowing by IP address:

```
  - !ipMatch
    ip: www.camptocamp.org
?   mask: 255.255.255.255
```

The “ip” parameter can be a DNS name that will be resolved or directly an IP address.

All the methods accept the following optional parameters:

- port: to limit to a certain TCP port
- pathRegexp: a regexp that must match the path part of the URL (before the ‘?’).

## 2.5 Metadata definition

Allow to add some metadata to the generated PDF. They are visible in acroread in the File->Properties menu.

The structure is like that:

```
    metaData:
?     title: ''
?     author: ''
?     subject: ''
?     keywords: ''
?     creator: ''
?     supportLegacyReader: false
```

All fields are optional and can use global variables, as defined in the Block definition chapter. Page specific variables are not accessible.

## 2.6 Page definition

The structure is like that:

```
      pageSize: A4
?     landscape: false
?     marginLeft: 40
?     marginRight: 40
?     marginTop: 20
?     marginBottom: 20
?     backgroundPdf: template.pdf
?     condition: null
?     header:
        height: 50
        items:
          - {BLOCK_DEFINITION}
          {...}
      items:
        - {BLOCK_DEFINITION}
        {...}
?     footer:
        height: 50
        items:
          - {BLOCK_DEFINITION}
          {...}
```

With the “condition” we can completely hide a page, same behavior than in block.

If “backgroundPdf” is specified, the first page of the given PDF file will be added as background of every page.

The “header” and “footer” sections are optional. If the “items” that are in the main section are too big, more pages are generated. The header and footer will be drawn on those pages as well.

Here is a short list of supported **pageSizes**:

| name | width | height |
|---|---|---|
| LETTER | 792 | 612 |
| LEGAL | 1008 | 612 |
| A4 | 842 | 595 |
| A3 | 1191 | 842 |

The complete list can be found in http://api.itextpdf.com/itext/com/itextpdf/text/PageSize.html. If you want to use a custom page size, you can set **pageSize** to the width and the height separated by a space.

## 2.7 Block definition

The next sub-sections document the possible types of blocks.

In general, text values or URLs can contain values taken from the **spec** structure coming with the client's request. A syntax similar to shell is used: ${variableName}. If the current page is a **titlePage**, only the root values are taken. If it's a **mainPage**, the service will first look in the current **page** section then in the root values. Here is how to use this functionality:

```
text: 'The value of mapTitle is: ${mapTitle}'
```

Some virtual variables can be used:

- ${pageNum}: The current page number.
- ${pageTot}: The total number of pages. Can be used only in text blocks.
- ${now}: The current date and time as defined by the machine's locale.
- ${now FORMAT}: The current date and time as defined by the FORMAT string. The syntax is here: http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html.
- ${configDir}: The absolute path to the directory of the configuration file.
- ${format PRINTF VAR}: Format the value of VAR using the provided PRINTF format (for example: %,d).

All the blocks can have a condition attribute that takes a spec attribute name. If the attribute name exists and is not equal to "false" or "0", the block is drawn. Otherwise, it is ignored. An exclamation mark may precede the condition to invert it, exclamation mark is part of yaml syntax, than the expression should be in quotes.

Example: show text block only if in the spec the attribute name "showText" is given, is not equal to "false" and not equal to "0":

```
- !text
  text: 'mytext'
  condition: showText
```

## 2.8 Text block

```
        - !text
?         font: Helvetica
?         fontSize: 12
?         fontEncoding: Cp1252
?         fontColor: black
?         spacingAfter: 0
?         align: left
?         vertAlign: middle
?         backgroundColor: #FFFFFF
          text: 'Blahblah'
```

Typical "fontEncoding" values are:

- Cp1250
- Cp1252
- Cp1257
- Identity-H (horizontal UTF-8)
- Identity-V (vertical UTF-8)
- MacRoman

The "font" must refer to a standard PDF font or a declared font.

## 2.9 Image block

```
        - !image
          maxWidth: 200
          maxHeight: 100
?         spacingAfter: 0
?         align: left
?         vertAlign: middle
          url: http://trac.mapfish.org/trac/mapfish/chrome/site/img/mapfish.png
```

Supported formats are PNG, GIF, Jpeg, Jpeg2000, BMP, WMF (vector), SVG and TIFF.

The original aspect ratio will be respected. The url can contain "${}" variables.

## 2.10 Columns block

```
        - !columns
?         config: {TABLE_CONFIG}
?         widths: [25,25,25,25]
?         backgroundColor: #FFFFFF
?         absoluteX: null
?         absoluteY: null
?         width: {PAGE_WIDTH}
?         spacingAfter: 0
?         nbColumns: -1
          items:
            - {BLOCK_DEFINITION}
            {...}
```

Can be called **!table** as well.

By default, the width of the columns will be equal.

Each item will be in its own column.

If the **absoluteX**, **absoluteY** and **width** are given, the columns block will be floating on top of the page at the specified position.

The **widths** attribute can be used to change the width of the columns (by default, they have the same width). It must contain one integer for each column. The width of a given column is *tableWidth*columnWeight/sum(columnWeight)*.

Every block type is allowed except for **map** if the column has an absolute position.

Look at <http://trac.mapfish.org/trac/mapfish/wiki/PrintModuleServer#Tableconfiguration to know how to specify the **config** field.

## 2.11 Map block

Allowed only within a **mainPage**.

```
        - !map
          width: ?
          height: ?
?         name: map
?         spacingAfter: 0
?         align: left
?         vertAlign: middle
?         absoluteX: null
?         absoluteY: null
?         overviewMap: null
?         backgroundColor: #FFFFFF
```

**width** and **height** are mandatory. You can use variable substitution in this part, but if you do so, the browser won't receive the map size when it calls **info.json**. You'll have to **override mapfish.widgets.print.Base.configReceived** and set the map width and height of your layouts.

If the **absoluteX** and **absoluteY** are given, the map block will be floating on top of the page at the specified position.

The **name** is what will be displayed in the Acrobat's reader layer panel. The map layers will be displayed bellow it.

If **overviewMap** is specified, the map will be an overview of the extent augmented by the given factor. There are few cases to consider with map overviews:

1. If there is no overview overrides and no OL.Control.MapOverview, then all the layers will figure in the PDF map overview.
2. If there are overview overrides, the OL map overview control is ignored.
3. If there are no overview overrides and there is an OL.Control.MapOverview (takes the first one), then the layers defined in the control are taken into account. By default it is the current base layer.

## 2.12 Scalebar block

Display a scalebar.

Allowed only within a **mainPage**.

```
        - !scalebar
          maxSize: 150
?         type: line
?         intervals: 3
?         subIntervals: false
?         units: m
?         barSize: 5
?         lineWidth: 1
?         barDirection: up
?         textDirection: up
?         labelDistance: 3
?         font: Helvetica
?         fontSize: 12
?         fontColor: black
?         color: #000000
?         barBgColor: null
?         spacingAfter: 0
?         align: left
?         vertAlign: middle
?         backgroundColor: #FFFFFF
?         lockUnits: true
```

The scalebar, will adapt its width up to *maxSize* (includes the labels) in order to have a multiple of 1, 2 or 5 values at each graduation. For example:

- 0, 1, 2, ...
- 0, 2, 4, ...
- 0, 5, 10, ...
- 0, 10, 20, ...

The *barSize* is the thickness of the bar or the height of the tick marks on the line. The *lineWith* is for the thickness of the lines (or bar border).

Units can be any of:

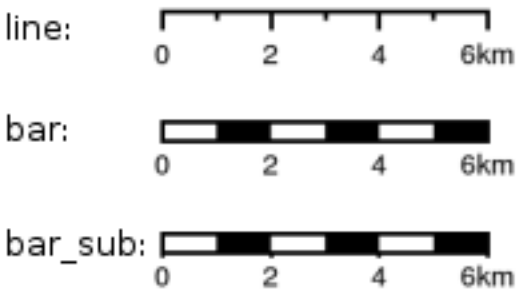
- m (mm, cm, m or km)
- ft (in, ft, yd, mi)
- degrees (min, sec, °)

If the value is too big or too small, the module will switch to one of the unit in parenthesis (the same unit is used for every intervals). If this behaviour is not desired, the lockUnits parameter will force the declared unit (or map unit if no unit is declared) to be used for the scalebar.

The number of *intervals* can be set to anything >=2. Labels are drawn only at main intervals. If there is no space to display a label at a certain interval, this label won't be displayed. If *subIntervals* are enabled, their number will depend on the length of an interval.

The type can be:

- line: A simple line with graduations
- bar: A thick bar with a suite of color and barBgColor blocks.
- bar_sub: Like bar, but with little lines for labels.




The bar and/or text orientation can be set to “up”, “down”, “left” or “right”.

The *align* attribute is for placing the whole scalebar withing the surrounding column or page. The *vertAlign* attribute is used only when placed in a column.

Labels are always centered on the graduation, at a distance specified by labelDistance.

## 2.13 Attributes block

Allows to display a table of the displayed feature’s attributes.

Allowed only within a *mainPage*.

```
        - !attributes
          source: results
?         tableConfig: {TABLE_CONFIG}
          columnDefs:
            {COLUMN_NAME}:
?             columnWeight: 0        MF_V1.2
              header: {BLOCK_DEFINITION}
              cell: {BLOCK_DEFINITION}
            {...}
```

Look here for how to specify the *tableConfig* field.

The *columnWeigth* (MF_V1.2 only) allows to define a weight for the column width. If you specify it for one column, you have to specify it for all of them. The width of a given column is tableWidth*columnWeight/sum(columnWeight).

The **source** value defines the name of the entry in the root of the client’s **spec**. For example, it would look like that:

```
{
  ...
  pages: [
    {
      ...
      results: {
        data: [
          {id:1, name: 'blah', icon: 'icon_pan'},
          ...
        ],
        columns: ['id', 'name', 'icon']
      }
    }
  ]
  ...
}
```

With this spec you would have to define 3 columnDefs with the names **id**, **name** and **icon**. Each cell definition blocks have access to all the values of the current row.

The spec part is filled automatically by the 2 MapFish widgets when their grids parameter is set.

Here is a crazy example of columnDef that will show the name of the icon and it's bitmap side-by-side inside a single column:

```
columnDefs:
  icon:
    header: !text
      text: Symbol
      backgroundColor: #A0A0A0
    cell: !columns
      items:
        - !text
          text: '${icon}'
        - !image
          align: center
          maxWidth: 15
          maxHeight: 15
          url: 'http://www.mapfish.org/svn/mapfish/trunk/MapFish/client/mfbase/mapfish/img/${icon}.pr
```

A more complex example can be found in SVN: config.yaml spec.json

The print widgets are able to fill the spec for you based on a dictionary of **Ext.grid.GridPanel**. Just pass them through the grids parameter.

## 2.14 Legends block

Display each layers along with its classes (icons and labels).

```
        - !legends
?         backgroundColor: #FFFFFF
?         borders: false
?         horizontalAlignment: center
?         maxWidth: 0
?         maxHeight: 0
?         iconMaxWidth: 0
?         iconMaxHeight: 8
?         iconPadding: 8 7 6 5
?         textMaxWidth: 8
?         textMaxHeight: 8
?         textPadding: 8 7 6 5
?         defaultScale: 1.0
?         inline: true
?         classIndentation: 20
?         layerSpaceBefore: 5
?         layerSpace: 5
?         classSpace: 2
?         layerFont: Helvetica
?         layerFontSize: 10
?         classFont: Helvetica
?         classFontSize: 8
?         fontEncoding: Cp1252
?         columnMargin: 3
```

**borders** is mainly for debugging purpouses and shows all borders in the legend tables. This can be either 'true' or 'false'.

**horizontalAlignment** can be left, right or center (default) and aligns all items left, right or in the center.

**iconMaxWidth**, **iconMaxHeight**, **defaultScale** with value of 0 indicate that the value will be ignored, i.e. the values are automatically set to the equivalent of Infinity, Infinity and 1 respectively. If the legends URL passed to MapFish (see http://mapfish.org/doc/print/protocol.html#print-pdf) are obtained from a WMS GetLegendGraphic request, the width/height are only indicative (even more when a label text is included with LEGEND_OPTIONS/forceLabels parameter) and it would be safer, in order to preserve scale coherence between legends and map, to set **iconMaxWidth** and **iconMaxHeight** to zero.

**textMaxWidth/Height** and **iconMaxWidth/Height** define how wide/high the text/icon cells of a legend item can be. At this point textMaxHeight is ignored.

**textPadding** and **iconPadding** can be used like standard CSS padding. In the above example 8 is the padding top, 7 padding right, 6 padding bottom and 5 padding left.

if **inline** is true icons and text are rendered on the same line, BUT multicolumn is still enabled.

if **maxWidth** is set the whole legend gets a maximum width, just like other blocks. Note that **maxWidth** does not have any impact on icons size, thus icons may overflow outside the legends block.

if **maxHeight** is set the whole legend gets a maximum height. This forces more than one column to appear if the legend is higher than the specified value. This can be used to enable the multi-column layout. 0 makes the maxHeight= max value, i.e. the equivalent of infinity.

if **defaultScale** is non null it means that the legend image will be scaled so it doesn't take the full space. This can be overriden for individual classes in the spec JSON sent to the print module by adding an attribute called 'scale' and giving it a number. In conjunction with iconMaxWidth/Height this can be used to control average and also maximum width/height. If **defaultScale** equals 1, one pixel is scaled to one point (1/72 inch) in generated PDF. By default, as GeoServer legends are generated with ~90 dpi resolution (exactly 25.4/0.28), setting **defaultScale** value to 0.7937 (72*0.28/25.4) produces legend icons of same size as corresponding map icons. As the LEGEND_OPTIONS/dpi GeoServer parameter is not handled by MapFish, the resolution will necessary be ~91 dpi, which may cause visual quality difference with the map.

For this to work, you need to set the **layerTree** config option on MF print widgets, more precisely the legends should be present in the print.pdf JSON request.

**layerSpaceBefore** is to specify the space before the second and consecutive layers.

**layerSpace** and **classSpace** is to specify the line space to add after layers and classes.

**columnMaxWidth** maximum width of a column in multi-column layout. Not tested (at time of writing).

**classIndentation** amount of points to indent classes by.

**layerSpaceBefore** if a layer is after another one, this defines the amount of space to have before it. This will not be applied if the layer is the first item in its column in multi-column layout.

**layerFont** font of layer name legend items.

**layerFontSize** font size of layer name.

**classFont** font of class legend items.

**classFontSize** font size of class.

**fontEncoding** (see below)

## 2.15 Table configuration

The columns block and the attributes block can take a table configuration object like that:

```
    config:
?     borderWidth: 0
?     borderWidthLeft: 0
?     borderWidthRight: 0
?     borderWidthTop: 0
?     borderWidthBottom: 0
?     borderColor: black
?     borderColorLeft: black
?     borderColorRight: black
?     borderColorTop: black
?     borderColorBottom: black
?     cells:
?       - {CELL_CONFIGURATION}
```

A cell configuration looks like that:

```
?     row: {...}
?     col: {...}
?     borderWidth: 0
?     borderWidthLeft: 0
?     borderWidthRight: 0
?     borderWidthTop: 0
?     borderWidthBottom: 0
?     borderColor: black
?     borderColorLeft: black
?     borderColorRight: black
?     borderColorTop: black
?     borderColorBottom: black
?     padding: 0
?     paddingLeft: 0
?     paddingRight: 0
?     paddingTop: 0
?     paddingBottom: 0
?     backgroundColor: white
?     align: LEFT
?     vertAlign: TOP
```

The stuff configured at table level is for the table border, not every cell.

The **cells** list defines overrides for some cells. The cells an override is applied to is defined by the **row** and **col** attribute. Those attributes can have several formats:

- **0**: apply only to row or column 0 (the first)
- **0-10**: applies only the row or columns from 0 to 10
- or you can use any regular expression

Every matching overrides is applied in order and will override the values defined in the previous ones.

For example, if you want to draw an attribute block like that:

| Seuchen grouppe | Tiera |
|---|---|
| Auszurottende Seuchen | Ziege |
| Auszurottende Seuchen | Ziege |
| Zu bekämpfende Seuchen | Bienen |
| Zu bekämpfende Seuchen | Bienen |
| Zu bekämpfende Seuchen | Bienen |

You define that:

```
- !attributes
  tableConfig:
    borderWidth: 1
    cells:
      # match every cell (default cell formatting)
      - borderWidthBottom: 0.5
        borderWidthLeft: 0.5
        padding: 4
        paddingTop: 0
      # match every even cell (yellowish background)
      - row: '\d*[02468]'
        backgroundColor: #FFFFCC
      # for the header
      - row: 0
        borderWidthBottom: 1
        backgroundColor: #FA0002
        align: center
  {...}
```

CHAPTER 3

# Protocol

Four commands are available and are documented in the next sections.

Every command uses the HTTP status code to notify errors.

## 3.1 info.json

HTTP command:

```
GET {PRINT_URL}/info.json?url={PRINT_URL}%2Finfo.json&var=printConfig
```

Returns a JSON structure as such:

```
var printConfig = {
    "scales":[
        {"name":"25000"},
        {"name":"50000"},
        {"name":"100000"}
    ],
    "dpis":[
        {"name":"190"},
        {"name":"254"}
    ],
    "outputFormats":[
        {"name":"pdf"},
        {"name":"png"}
    ],
    "layouts":[
        {
            "name":"A4 portrait",
            "map":{
                "width":440,
                "height":483
            }
        }
    ],
    "printURL":"http:\/\/localhost:5000\/print\/print.pdf",
    "createURL":"http:\/\/localhost:5000\/print\/create.json"
}
```

This can be loaded through an HTML script tag like that:

```
<script type="text/javascript"
      src="http://localhost:5000/print/info.json?var=printConfig"></script>
```

or through an AJAX request, in this case the `var` query parameter will be omitted.

The "url" query parameter is here to help the print servlet to know what URL is used by the browser to access the servlet. This parameter is here because the servlet can be behind a proxy, hiding the real URL.

## 3.2 print.pdf

HTTP command:

```
GET {PRINT_URL}/print.pdf?spec={SPEC}
   or
POST {PRINT_URL}/print.pdf    with {SPEC} in the request body
```

The "SPEC" parameter is a JSON structure like that:

```
{
    layout: 'A4 portrait',
    ...CUSTOM_PARAMS...
    srs: 'EPSG:4326',
    units: 'degrees',
    geodetic: false,
    outputFilename: 'political-boundaries',
    outputFormat: 'pdf',
    layers: [
        {
            type: 'WMS',
            layers: ['basic'],
            baseURL: 'http://labs.metacarta.com/wms/vmap0',
            format: 'image/jpeg'
        }
    ],
    pages: [
        {
            center: [6, 45.5],
            scale: 4000000,
            dpi: 190,
            geodetic: false,
            ...CUSTOM_PARAMS...
        }
    ],
    legends: [
        {
            classes: [
                {
                    icons: [
                        'full url to the image'
                    ],
                    name: 'an icon name',
                    iconBeforeName: true
                }
            ],
            name: 'a class name'
        }
```

```
    ]
}
```

The location to show on the map can be specified with a **center** and a **scale** as show or with a **bbox** like that:

```
bbox: [5, 45, 6, 46]
```

The print module will use the nearest scale and will make sure the aspect ratio stays correct.

The geodetic parameter can be set to true so the scale of geodetic layers can correctly be calculated. Certain projections (Google and Latlong for example) are based on a spheroid and therefore require **geodetic: true** in order to correctly calculate the scale. If the geodetic parameter is not present it will be assumed to be false.

The outputFilename parameter is optional and if omitted the values used in the server's configuration will be used instead. If it is present it will be the name of the downloaded file. The suffix will be added if not left off in the parameter. The date can be substituted into the filename as well if desired. See configuration's outputFilename for more information and examples

The outputFormat parameter is optional and if omitted the value 'pdf' will be used. Only the formats returned in the info are permitted.

There are two locations where custom parameters can be added. Those will be ignored by the web service but, will be accessible from the layout templates.

For the format of the **layers** section, please look at the implementations pointed by mapfish.PrintProtocol.SUPPORTED_TYPES.

This command returns the PDF file directly.

## 3.3 create.json

HTTP command:

```
POST {PRINT_URL}/create.json?url={PRINT_URL}%2Fcreate.json
```

The spec defined in the "print.pdf" command must be included in the POST body.

Returns a JSON structure like that:

```
{
    getURL: 'http:\/\/localhost:5000\/print\/56723.pdf'
}
```

The URL returned can be used to retrieve the PDF file. See the next section.

## 3.4 {ID}.pdf

This command's URL is returned by the "create.json" command.

HTTP command:

```
GET {PRINT_URL}/{ID}.pdf
```

Returns the PDF. Can be called only during a limited time since the server side temporary file is deleted afterwards.

# Layers Params

## 4.1 Vector

Type: vector

Render vector layers. The geometries and the styling comes directly from the spec JSON.

- opacity (Defaults to `1.0`)
- geoJson (Required) the geoJson to render
- styleProperty (Defaults to '_style') Name of the property within the features to use as style name. The given property may contain a style object directly.
- styles (Optional) dictonary of styles. One style is defined as in OpenLayers.Feature.Vector.style.
- name (Defaults to `vector`) the layer name.

## 4.2 WMS

Type: wms

Support for the WMS protocol with possibilities to go through a WMS-C service (TileCache).

- opacity (Defaults to `1.0`)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- layers (Required)
- styles (Optional)
- format (Required)
- useNativeAngle (Defaults to false) it true transform the map angle to customParams.angle for GeoServer, and customParams.map_angle for MapServer.

## 4.3 WMTS

Type: wmts

Support for the protocol using directly the content of a WMTS tiled layer, support REST or KVP.

Two possible mode, standard or simple, the simple mode imply that all the topLeftCorner are identical.

Standard mode:

- opacity (Defaults to 1.0)
- baseURL the 'ResourceURL' available in the WMTS capabilities.
- customParams (Optional) Map, additional URL arguments
- layer (Required) the layer name
- version (Required) `1.0.0`
- requestEncoding (Required) `REST` or `KVP`
- style (Required) the style name
- dimensions (Optional) list of dimensions names
- params (Optional) dictionary of dimensions name (capital) => value
- matrixSet (Required) the name of the matrix set
- matrixIds (Required) array of matrix ids e.g.:

```
[{
    "identifier": "0",
    "matrixSize": [1, 1],
    "resolution": 4000,
    "tileSize": [256, 256],
    "topLeftCorner": [420000, 350000]
}, ...]
```

- format (Optional, Required id requestEncoding is `KVP`)

Simple mode:

- baseURL base URL without the version.
- layer (Required)
- version (Required)
- requestEncoding (Required) `REST`
- tileOrigin (Required)
- tileSize (Required)
- extension (Required)
- resolutions (Required)
- style (Required)
- tileFullExtent (Required)
- zoomOffset (Required)
- dimensions (Optional)

- params (Optional)
- formatSuffix (Required)

## 4.4 Tms

Type: tms

Support the TMS tile layout.

- opacity (Defaults to 1.0)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- maxExtent (Required) Array, extent coordinates `[420000, 30000, 900000, 350000]`
- tileSize (Required) Array, tile size e.g. `[256, 256]`
- format (Required)
- layer (Required)
- resolutions (Required) Array of resolutions
- tileOrigin (Optional) Object, tile origin. Defaults to `0,0`

Resources:

- Quick intro to TMS requests: http://geowebcache.org/docs/current/services/tms.html
- TMS Spec (Not an Official Standard): http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification

## 4.5 Xyz

Type: xyz

Support the tile layout z/x/y.<extension>.

- opacity (Defaults to 1.0)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- maxExtent (Required) Array, extent coordinates `[420000, 30000, 900000, 350000]`
- tileSize (Required) Array, tile size e.g. `[256, 256]`
- resolutions (Required) Array of resolutions (Required) Array of resolutions
- extension (Required) file extension (Required) file extension
- tileOrigin (Optional) Array, tile origine e.g. `[420000, 350000]`
- tileOriginCorner `tl` or `bl` (Defaults to `bl`)
- path_format (Optional) url fragment used to construct the tile location. Can support variable replacement of `${x}`, `${y}`, `${z}` and `${extension}`. Defaults to zz/x/y.extension format. You can use multiple "letters" to indicate a replacable pattern (aka, `${zzzz}` will ensure the z variable is 0 padded to have a length of AT LEAST 4 characters).

## 4.6 Osm

Type: osm

Support the OSM tile layout.

- opacity (Defaults to `1.0`)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- maxExtent (Required) Array, extent coordinates `[420000, 30000, 900000, 350000]`
- tileSize (Required) Array, tile size e.g. `[256, 256]`
- resolutions (Required) Array of resolutions
- extension (Required) file extension

## 4.7 TileCache

Type: tileCache

Support for the protocol using directly the content of a TileCache directory.

- opacity (Defaults to `1.0`)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- layer (Required)
- maxExtent (Required) Array, extent coordinates `[420000, 30000, 900000, 350000]`
- tileSize (Required) Array, tile size e.g. `[256, 256]`
- resolutions (Required) Array of resolutions
- extension (Required) file extension

## 4.8 Image

Type: image

- opacity (Defaults to `1.0`)
- name (Required)
- baseURL (Required) Service URL
- extent (Required)

## 4.9 MapServer

Type: mapServer

Support mapserver WMS server.

- opacity (Defaults to `1.0`)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- layers (Required)
- format (Required)

## 4.10 KaMap

Type: kaMap

Support for the protocol using the KaMap tiling method

- opacity (Defaults to `1.0`)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- map
- group
- maxExtent (Required) Array, extent coordinates `[420000, 30000, 900000, 350000]`
- tileSize (Required) Array, tile size e.g. `[256, 256]`
- resolutions (Required) Array of resolutions
- extension (Required) file extension

## 4.11 KaMapCache

Type: kaMapCache

Support for the protocol talking direclty to a web-accessible ka-Map cache generated by the precache2.php script.

- opacity (Defaults to `1.0`)
- baseURL (Required) Service URL
- customParams (Optional) Map, additional URL arguments
- map (Required)
- group (Required)
- metaTileWidth (Required)
- metaTileHeight (Required)
- units (Required)
- maxExtent (Required) Array, extent coordinates `[420000, 30000, 900000, 350000]`
- tileSize (Required) Array, tile size e.g. `[256, 256]`
- resolutions (Required) Array of resolutions
- extension (Required) file extension

## 4.12 Google

Type: google or tiledGoogle

They used the Google Map Static API, tiledGoogle will create tiles and google only one image.

The google map reader has several custom parameters that can be added to the request they are:

- opacity (Optional, Defaults to `1.0`)
- baseURL (Required, should be 'http://maps.google.com/maps/api/staticmap')
- customParams (Optional) Map, additional URL arguments
- maxExtent (Required, should be `[-20037508.34, -20037508.34, 20037508.34, 20037508.34]`)
- resolutions (Required, should be `[156543.03390625, 78271.516953125, 39135.7584765625, 19567.87923828125, 9783.939619140625, 4891.9698095703125, 2445.9849047851562, 1222.9924523925781, 611.4962261962891, 305.74811309814453, 152.87405654907226, 76.43702827453613, 38.218514137268066, 19.109257068634033, 9.554628534317017, 4.777314267158508, 2.388657133579254, 1.194328566789627, 0.5971642833948135, 0.29858214169740677, 0.14929107084870338, 0.07464553542435169]`)
- extension (Required, should be `png`)
- client (Optional)
- format (Optional)
- maptype (Required) - type of map to display: http://code.google.com/apis/maps/documentation/staticmaps/#MapTypes
- sensor (Optional) - specifies whether the application requesting the static map is using a sensor to determine the user's location
- language (Optional) - language of labels.
- markers (Optional) - add markers to the map: http://code.google.com/apis/maps/documentation/staticmaps/#Markers

```
markers: ['color:blue|label:S|46.5195933305192,6.566684726913701']
```

- path (Optional) - add a path to the map: http://code.google.com/apis/maps/documentation/staticmaps/#Paths

```
path: 'color:0x0000ff|weight:5|46.5095933305192,6.506684726913701|46.5195933305192,6.526684726913701
```

CHAPTER 5

# FAQ

**All I get in my PDF is: "ERROR: infinite table loop". What's wrong?** Something in your page is too big. For example, the width or the height of your !map block.

**I tried to print (pylons mode) and I get a "Java error". What's next?** Look in the apache error log, you'll find more information.

**What are the limitations of the type 2 layers?** It depends mostly on the map server you use. For the moment, GeoServer has not been extensively tested. With MapServer:

- The PDF output must be enabled when you compile and doesn't work in WMS mode, only in native MapServer mode. There are some limitations. on the styling. And you must use truetype fonts.
- The SVG output is limited regarding the stylings you can use. For example only plain polygon fillings are supported by MapServer. If a complex styling is used, your features may appear plain black.

**I tried to change the layout and half the Map is printed off the page on the right. Or I have an empty page added. Is it a bug?** It's mostly a feature ;-) . This kind of behavior can be seen in iText, when adding a block that is too big for the page size. Try to reduce the size of your map block.

**When I look at my generated PDF in Acrobat Reader, it looks good. But, when I print it, it misses some tiles/layers, some bitma** There are three possible explanations:

- Your printer has not enough memory: in Acrobat's print dialog, select "Save printer memory"
- Your printer firmware is buggy: upgrade it
- Your printer driver is buggy: upgrade it

**The module needs to go through a proxy to acces the map services.** It's so 90s... you should hire some fresh guys for your IT team. ;-)

You need to set some system properties (http.proxy*) when you start your java programs.

**On the browser, the scale is displayed with spaces to separate thousands and it's against my religion. How do I put my sacred se** By default, the browser's configured locale is used. You can force another locale in the print widget configuration:

```
{
  ...
  configUrl: 'print/info.json',
  serviceParams: { locale: 'fr_CH' },
  ...
}
```

**I copied the examples and the print widgets are not working.** First edit the client/examples/examples.js file and make sure the URLs are correct.

1. If you don't want to install the server side, make sure you installed a proxy (see Configure Proxy). For example, test (must return a JSON content, not the proxy.cgi script's content) it with an URL like that (adapt the hostname, port and path): *http://localhost/cgi-bin/proxy.cgi?url=http://demo.mapfish.org/mapfishsample/trunk/print/info.json*
2. If you installed the server side, make sure it works by calling the URL specified in the mapfish.SERVER_BASE_URL variable (must be the hostname/port your page is accessed through) added with */print/info.json*. For example, if you have *mapfish.SERVER_BASE_URL="http://localhost/mapfish": http://localhost/mapfish/print/info.json*

If it still doesn't work, use firefox, install firebug and check in the console panel that the AJAX request made by the print widget works fine.

CHAPTER 6

# Warranty disclaimer and license

The authors provide these documents “AS-IS”, without warranty of any kind either expressed or implied.

Document under Creative Common License Attribution-Share Alike 2.5 Generic.

Authors: MapFish developers.