BarcodeAPI.org

Technical User Manual

1 About

Designed with scale in mind, the BarcodeAPI.org[1] web server was built to handle even the most demanding environments. With some instances having generated over 100 million barcodes, the server can be used for nearly and use-cases. Using their favorite web browser a user can navigate to an elegant website and quickly create barcodes of many different types. For advanced users who wish to automate the generation of barodes, the server also features a full rendering API via the HTTP protocol; this is useful where barcode generation libraries might not be available or where system resources are limited.

2 API Server

The server will generate a barcode for any content passed to the /api endpoint; this can be done using a web browser, fetched through a user script, or even simply with cURL.

curl / wget

```
curl https://barcodeapi.org/api/A_Barcode > gen.png

wget -o gen.png https://barcodeapi.org/api/A_Barcode
```

HTML / JS

```
<img id="barcode"/>

<script>
    document.getElementById("barcode").src = "https://barcodeapi.org/api/A_Barcode";
</script>
```

2.1 Type Selection

The API offers two types of barcode type selection, automatic or manual; users of the web-based interface might prefer the barcode type be selected based on what they type, while direct callers of the API might want to have control over which barcode format they will receive back from the server.

2.1.1 Automatic Type Detection

When calling the API endpoint without specifying an explicit code type the server will make its best judgment as to which code type will be best suited for the supplied data.

```
curl https://barcodeapi.org/api/abc123
curl https://barcodeapi.org/api/auto/abc123
```

2.1.2 Manual Type Selection

A specific barcode type may be requested by using the type string in the request URL.

```
curl https://barcodeapi.org/api/128/abc123
curl https://barcodeapi.org/api/qr/abc123
```

2.2 Web Server

The server comes with a minimal set of static HTML and Javascript files that allow users to generate barcodes in their web browser with ease. The UI allows users to quickly download, print, or copy the images for use in other applications.

2.2.1 Multi-Barcode Generation

Some users will want to generate a large number of barcodes with one request - a basic JavaScript utility is provided at /multi.html which will generate as many images as requested then prepare the file to be printed.

```
https://barcodeapi.org/multi.html?Barcode1&Barcode2&dm/A%20Data%20Matrix&qr/And%20QR
```

2.2.2 Forced Redirect

The server provides a set of static resources served from the root url of the server, in the event a request is made which is not targeted directly at the API nor references a known resources, a 403 redirect will be provided to send the send request to the API handler.

```
$ curl --head https://barcodeapi.org/abc123
...
Location: https://barcodeapi.org/api/auto/abc123
```

2.3 Response Headers

The server will add several headers related to the barcode including the type and encoded contents.

```
$ curl --head https://barcodeapi.org/api/auto/abc123
...
X-Barcode-Type: Code128
```

```
X-Barcode-Content: abc123
Content-Type: image/png;charset=utf-8
Content-Disposition: filename=abc123.png
```

Cache Control
In an attempt to minimize the number of requests

```
Cache-Control: max-age=86400, public
```

Error Headers
In the event of a render error, an additional header will be available containing details about the exception.

```
X-Error-Message: (exception.toString())
```

3 Barcode Generation

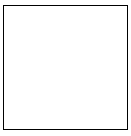
The primary purpose of the server is barcode image generation behind a RESTful endpoint. The server has been designed to support a wide variety of barcode types and offers additional configuration options for most.

3.1 Barcode Types

A list of all supported barcodes types is available by calling the /types/ endpoint; this will be a JSON Array containing details for each type including a regex format to pre-validate the data before each request.

Parameter	Description
name	The name of the Barcode format
target	The target name to use in the API request
pattern	The format pattern accepted by the barcode
description	The description of the barcode

3.1.1 Aztec



Does not require a quiet-zone, making it a smaller size.

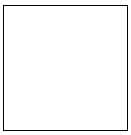
Alias `aztec`

Format `[ !"#$%&'()*+,-.\/0-9:;<=>?@A-Z\[\\\]^_`a-z{|}~]+`

Parameter	Unit	Default
size	px	300

margin	scalar	2
layers	(-4 <> 32)	0
correction	(0 <> 8)	4

3.1.2 Codabar

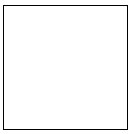


Alias codabar

Format `[0-9-:$\\/.+]+`

Parameter	Unit	Default
dpi	ppi	200
scale	scalar	1
qz	px	scale * 10
qzV	px	scale * 2
text	(top|bottom|none)	bottom

3.1.3 Code39

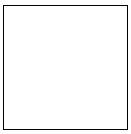


Alias 39, code-39, code39

Format `^[A-Z*0-9 -$%.\\/+]+$`

Parameter	Unit	Default
dpi	ppi	200
scale	scalar	1
qz	px	scale * 10
qzV	px	scale * 2
text	(top|bottom|none)	bottom

3.1.4 Code128



Alias 128, code-128, code128

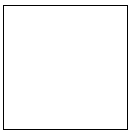
Format `^[ !\"#$%&'()*+,-.\\/0-9:;<=>?@A-Z\\[\\\\\\]^_`a-z{|}~]+$`

Parameter	Unit	Default
dpi	ppi	200
scale	scalar	1
qz	px	scale * 10
qzV	px	scale * 2

text	(top|bottom|none)	bottom

Note
The minimum width of the Quiet Zone to the left and right of the 128 Bar Code is 10x, where x is the minimum width of a module. It is mandatory at the left and right side of the barcode.[8]

3.1.5 Data Matrix

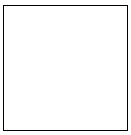


Alias dm, data-matrix, datamatrix, matrix, data

Format `^[ !\"#$%&'()*+,-.\\/0-9:;<=>?@A-Z\\[\\\\\\]^_`a-z{|}~]{1,2335}$`

Parameter	Unit	Default
dpi	ppi	200
scale	scalar	1
qz	scalar	scale * 2

3.1.6 EAN-8

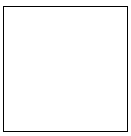


Alias 8, ean-8, ean8

Format `^[0-9]{7,8}$`

Parameter	Unit	Default

3.1.7 EAN-13

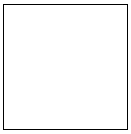


Alias 13, ean-13, ean13

Format `^[0-9]{12,13}$`

Parameter	Unit	Default

3.1.8 PDF-417

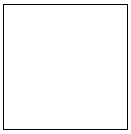


Alias 417, pdf417, pdf

Format `^[ !\"#$%&'()*+,-.\\/0-9:;<=>?@A-Z\\[\\\\\\]^_`a-z{|}~]{1,2335}$`

Parameter	Unit	Default

3.1.9 QR Code

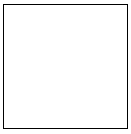


Alias qr, qr-code, qrcode

Format `^.{1,65535}$`

Parameter	Unit	Default
size	px	300
qz	scalar	2

3.1.10 UPC-A

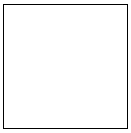


Alias a, upc-a, upca, upc

Format `^(?=.*0)[0-9]{11,12}$`

Parameter	Unit	Default

3.1.11 UPC-E



Alias e, upc-e, upce

Format `^(?=.*0)[0-9]{7,8}$`

Parameter	Unit	Default
dpi	ppi	150

3.2 Control Characters

Barcodes will frequently contain control characters for various reasons; as they can be difficult to enter in a text field the API server has implemented a special mechanism for allowing users to easily generate barcodes containing these characters. Similar to the standard caret notation[9], these special symbols can be requested for supported barcodes by using the prefix $$ followed by the proper character representation.

Supported barcode formats:

- Aztec

- Code128
- DataMatrix
- PDF417
- QR Code

Special	HEX	Raw	Description
-	-	-	-
$$@	0x00	NUL	**Null** Often used as a string terminator
$$A	0x01	SOH	**Start of Heading** Delimits the start of a message header
$$B	0x02	STX	**Start of Text** Terminates a message header and begins the body
$$C	0x03	ETX	**End of Text** Terminates the body of the message
$$D	0x04	EOT	**End of Transmission** Delimits the end of a message / stream / file
$$E	0x05	ENQ	**Enquiry** A ping to the server
$$F	0x06	ACK	**Acknowledge** A pong to the client to signal it is ready
$$G	0x07	BEL	**Bell** May trigger an alert or notification
$$H	0x08	BS	**Backspace** May overprint the previous character
$$I	0x09	TAB	**Character Tab** Move to the next character tab-stop
$$J	0x0A	LF	**Line Feed** Move down one line without changing character position
$$K	0x0B	VT	**Line Tab** Move to the next line tab-stop
$$L	0x0C	FF	**Form Feed** On printers this will load the next page, often used as a page break or to clear a terminal.
$$M	0x0D	CR	**Carriage Return** Mark the end of a line (Enter key)
$$N	0x0E	SO	**Shift Out** Switch to an alternate character-set
$$O	0x0F	SI	**Shift In** Switch back to the standard character-set
$$P	0x10	DLE	**Data Link Escape** Signals the following bytes are to be interpreted as raw data
$$Q	0x11	DC1	**Device Control 1 (XON)** Reserved for device control
$$R	0x12	DC2	**Device Control 2** Reserved for device control

$$S	0x13	DC3	**Device Control 3 (XOFF)** Reserved for device control
$$T	0x14	DC4	**Device Control 4** Reserved for device control
$$U	0x15	NAK	**Negative Acknowledge** A pong to the client to signal it is not ready
$$V	0x16	SYN	**Synchronous Idle** Used to signal synchronization in a stream
$$W	0x17	ETB	**End of Transmission Block** Used to signal the end of a transmission in a stream, often used as an end of paragraph marker.
$$X	0x18	CAN	**Cancel** Used to signal that the data preceding is invalid
$$Y	0x19	EM	**End of Medium** Indicates the end of a usable medium has been reached, often used as the first character of a paragraph
$$Z	0x1A	SUB	**Substitute** In Windows/DOS, often used to indicate the end of a file
$$[	0x1B	ESC	**Escape** Represents the escape character on the keyboard
$$\	0x1C	FS	**File Separator** May be used separate data fields in a stream
$$]	0x1D	GS	**Group Separator** May be used separate data fields in a stream
$$^	0x1E	RS	**Record Separator** May be used separate data fields in a stream
$$_	0x1F	US	**Unit Separator** May be used separate data fields in a stream

4 Server Management

4.1 Launch Arguments

Argument	Description
`--port $port`	
`--no-web`	

4.2 Server Statistics

The server will keep counters for all handlers and caches, these statistics are available at the /stats/ endpoint.

Metric	Description
request.count.total	
request.count.$handler	
render.count.total	
render.count.$type	
render.time.total	
render.time.$type	
render.fail.total	

4.3 Caching

The server implements a basic cache for rendered images, provided the cached object has not expired and been evicted, a request for a previously rendered barcode will will be rapidly served from memory instead of being rendered; the cache will only render requests matching a certain criteria.

Metric	Description
cache.$type.hit	
cache.$type.miss	
cache.$type.add	
cache.$type.remove	

4.4 Session Management

A simple session cache will track all user actions; this can be used to provide the user with a list of their most used barcodes; this information is available at the /session/ endpoint.

4.5 Tasks

4.5.1 Watchdog

1 minute

4.5.2 Statistics

5 minutes

4.5.3 Cache Cleanup

Frequency: 1 hour
Life: 3 days

The cache cleanup task ensures that stale objects do not remain lingering in the cache if they are not being utilized. Evaluated each hour, each item in the cache will be checkd for it's last access time; any barcode object which has not been accessed in the last 3 days is evicted from the cache.

4.5.4 Session Cleanup

Frequency: 15 minutes
Life: 6 hours

Like the cache cleanup task, the session cleanup task is periodically evaluated to expire old user sessions. Evaluated every 15 minutes the task will evict any sessions older then 6 hours; this means that user sessions are very transient and will likely only last a single working day for web-based users.

4.6 Blacklist

Based on the needs of some organizations it may be necessary to blacklist certain formats of codes from being generated. Located in the resources directory, server administrators can customize the blacklist by appending entries to the 'blacklist.conf' file. Entries in this file are evaluated as regular expressions which allow a wide range of format matching possibilities.

4.7 Logging

Being fairly verbose, the server will log all important events and user requests; to help with post-processing, the logs are automatically split based on their category.

4.7.1 Server Log

The main server log will include log entries for important events.

Generic server messages
Task executions

4.7.2 Error Log

Render errors

4.7.3 Request Log

All requests to the Jetty server

Format

```
$handler : $target : $source : $from [ : $proxy ]
```

Parameter	Description
`$handler`	The API handler receiving the request
`$target`	The user requested target
`$source`	The source of the request (API / URL)
`$from`	The source of the request (IP)
`$via`	The proxy between the endpoints.

4.7.4 Barcode Log

All requests to the render engine

Format

```
Rendered [ $type ] with [ $data ] size [ $bytes ] in [ $time ]
```

Parameter	Description
`$type`	
`$data`	
`$bytes`	
`$time`	

4.7.5 Log Rollover

5 Free, Open Source Software

5.1 License

```
Copyright 2020, BarcodeAPI.org
```

```
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

5.2 Third Party

BarcodeAPI.org is only made possible with the use of third-party software.

Jetty[4], Apache 2.0

- The BarcodeAPI server was built around the Jetty web server framework.

Barcode4J[5], Apache 2.0

- Barcode4J is an open source barcode generator; it is used for the generation of the following code types:

ZXing[6], Apache 2.0

- ZXing is a barcode processing library that makes QR code generation possible.

6 Resources

Reference	Title	Target
[1]	BarcodeAPI.org	BarcodeAPI
[2]	MClarkDev.com GIT Repository	MClarkDev
[3]	C0 and C1 control codes	Wikipedia
[4]	Jetty	
[5]	Barcode4J	
[6]	ZXing	
[7]	Apache 2.0 License	
[8]	Code 128 - Quiet Zone	Wikipedia
[9]	Caret Notation	Wikipedia
[10]	C1 and C0 Control Codes	Wikipedia