Buddhi Vikasitha
Software Engineer, Graphic Designer, Gamer, Mahindian, University of Moratuwa Undergraduate.
May 21 · 5 min read

How to directly print from your browser using QZ Tray and use a signed certificate

Mostly, in developing web based cashier applications the next problem after setting up transaction logic is, “How to pass the information to a printer device”. This can be done by qz.io library provided by qz technologies. Actually this is a great tool to do communication between your web application and the printer.

https://qz.io/

Using qz library

Qz works in two parts. One, the qz client that we download as a setup and install on our computer. This must run in the background from the startup and this creates print jobs for our printer. Other part is the source code that we include in our project and try to issue commands to the client.

So let’s see how to make this happen with qz.io.

- First download the qz setup from their website and install it. Then you have what issues print commands to your printer. Make sure you download the qz tray 2.0 because that version is more compatible with what we are going to do.
- Second, include the source codes in your project. These are located at the application installation location.

```
<script type="text/javascript" src="js/dependencies/rsvp-
3.1.0.min.js"></script>

<script type="text/javascript" src="js/dependencies/sha-
256.min.js"></script>
```

```
<script type="text/javascript" src="js/qz-tray.js"></script>
```

- Then you can use the qz commands in the application.

```
qz.websocket.connect().then(function() {
  //next function after connecting to the service
});
```

- Apart from that there are a lot of commands and these steps are also well written in the getting started page.

Printing types

There are several types of feeding data to qz for printing. You can use so many types as raw data, HTML, base64, image types, etc. More information about printing is included in this page. The performance, quality and the content may vary depending on the type of data we are using for printing.

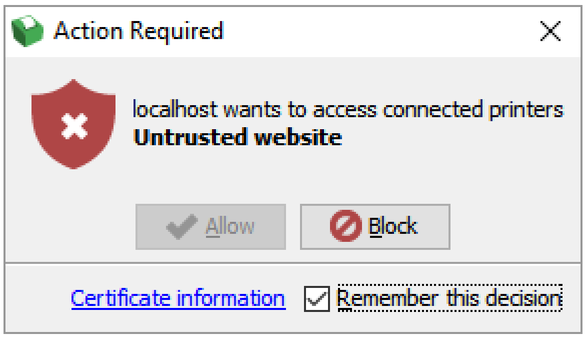
```
function print() {
   var config = qz.configs.create("your printer name");

   var data = [
      'Data\n',
      'Should be simple data\n',
      'To be printed\n'
   ];

   qz.print(config, data).catch(function(e) {
       console.error(e);
   });
}
```

By using a simple script like this, you can print something on your printer. You can use raw data for very fast printing like POS printing. Try to change options in printing for a better result.

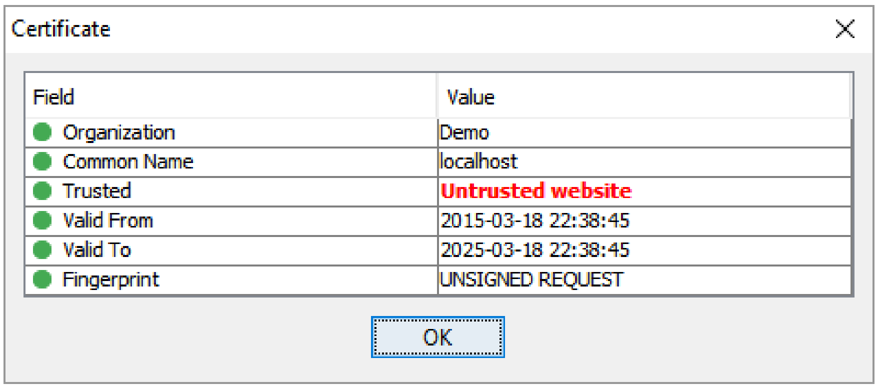
Now you got the setup and a basic printing right. Now let’s head on to the advanced part of this article.

Self-signed certificates

The only problem that is there is the warning dialogs that pops up in the way during each execution of print, connect commands. It only happens in free version of the client. This is really annoying in general use. This happens due to unsigned certificates that come with each request to the qz client. From version 2.0 the developers have provided a way to recompile the binaries with a generated self signed certificate and make the popups go away.



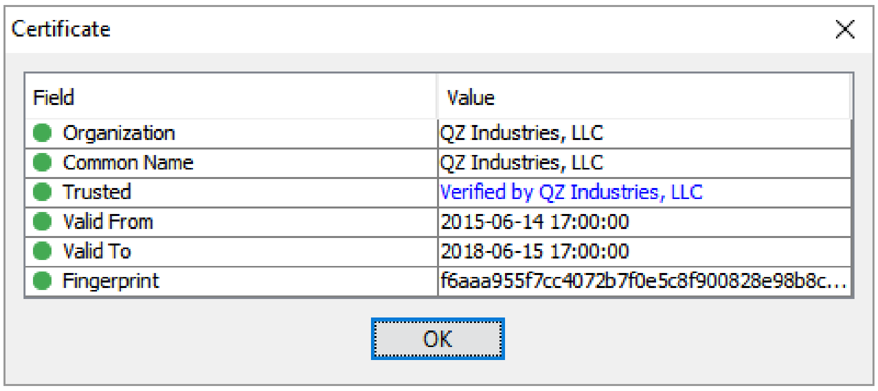
This confirmation pops up everytime


Certificate

Field	Value
● Organization	Demo
● Common Name	localhost
● Trusted	Untrusted website
● Valid From	2015-03-18 22:38:45
● Valid To	2025-03-18 22:38:45
● Fingerprint	UNSIGNED REQUEST

Untrusted details of the website without certificate

Our goal is to make the popup say that the request for the client is from a trusted location. To successfully do this, you'll need to recompile the client setup using a key and a certificate that are self-signed.


Certificate

Field	Value
● Organization	QZ Industries, LLC
● Common Name	QZ Industries, LLC
● Trusted	Verified by QZ Industries, LLC
● Valid From	2015-06-14 17:00:00
● Valid To	2018-06-15 17:00:00
● Fingerprint	f6aaa955f7cc4072b7f0e5c8f900828e98b8c…

We need to make the website info look like this

1. First we will recompile the binaries for the client. Before doing so, you should install the required dependencies and setup them.

- Install JDK 7 or higher. http://www.oracle.com/technetwork/java/javase/downloads/
- Setup Apache Ant. https://ant.apache.org/bindownload.cgi
- Setup following environmental variables. http://ant.apache.org/manual/install.html#setup

```
set ANT_HOME=c:\ant
set JAVA_HOME=c:\jdk1.7.0_51
set PATH=%PATH%;%ANT_HOME%\bin
```

- Install NSIS 3.0+. http://nsis.sourceforge.net/Download
- Install git. https://git-scm.com/download/win

2. Clone the source code for the application. https://github.com/qzind/tray

3. Install OpenSSL.

- OpenSSL is used to create the self signed certificate. You can download OpenSSL binaries and start creating a certificate and a key for your own. https://slproweb.com/products/Win32OpenSSL.html
- Use the following code to generate a key and certificate. Various inouts will be required in making this certificate though. We only need to enter those details in each step.

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out
cert.pem -days 11499 -nodes
```

- Following will be some example inputs input while running this command.

Country Name (2 letter code) [XX] *: LK*

***State or Province Name (full name) [Some-State]** : Western*

***Locality Name (eg, city) []** : Colombo*

***Organization Name (eg, company) [Internet Widgits Pty Ltd]**: University of Moratuwa*

***Organizational Unit Name (eg, section) []**: CSE*

Common Name (e.g. server FQDN or YOUR name) []**: THIS ENTRY IS IMPORTANT, this should be your domain name in wildcard format an example of this would be ".mywebsitedomain.com**": *.buddhiv.com*

***Email Address []**: 92buddhiv@gmail.com*

- Once we have the certificate and the key generated we need to do another round of modification.
- We need to convert our key to a different format using the following command. If you are prompted to use a password in this step, please use a secure password.

```
openssl pkcs12 -inkey key.pem -in cert.pem -export -out
privateKey.pfx
```

4. Recompile the binaries using the new certificate.

- Navigate to the location of the source code using the command prompt. Now run this command to build using ant and the newly created certificate file.

```
c:\ant\bin\ant nsis -Dauthcert.use="c:\OpenSSL-
Win64\bin\cert.pem"
```

- Make sure that you modify the command according to your file paths where ant and the new certificate file lies.
- The new installation files will be located at the "out" folder of your source code folder.

5. Now what you have to do is install the newly created installation file in the system.

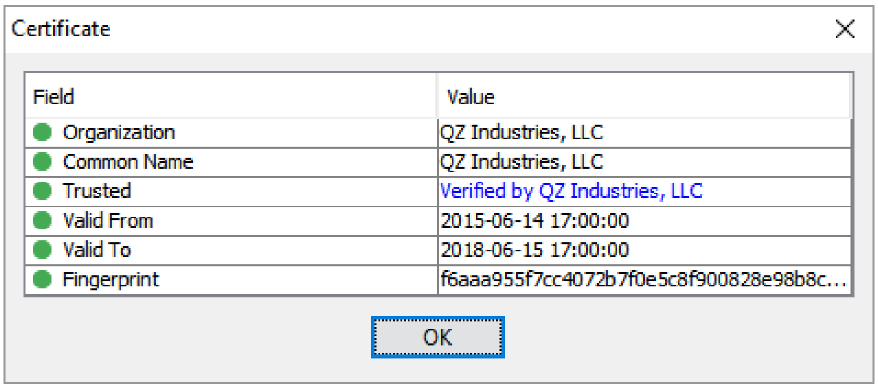
6. Next we will sign the messages from client side.

- First read the basic instructions for signing messages on this page. https://qz.io/wiki/2.0-signing-messages
- Here, I will show the easiest way of signing a message and get the qz tray popup done. It is by editing the qz.security.setSignaturePromise() function.

```
qz.security.setCertificatePromise(function (resolve, reject)
{
    resolve("-----BEGIN CERTIFICATE-----\n" +

"MIID5TCCAs2gAwIBAgIJALzQm+/XXbwMMA0GCSqGSIb3DQEBCwUAMIGIMQs
wCQYDVQQGEwJMSzEQMA4GA1UECAwHV2VzdGVybjEQMA4GA1UEBwwHQ29sb21
ibzEQMA4GA1UECgwHTWVlcHVyYTELMAkGA1UECwwCSVQxEjAQBgNVBAMMCWx
vY2FsaG9zdDEiMCAGCSqGSIb3DQEJARYTOTJidWRkaGl2QGdtYWlsLmNvbTA
eFw0xNzA1MDQxMjM4MjZaFw00ODEwMjcxMjM4MjZaMIGIMQswCQYDVQQGEwJ
MSzEQMA4GA1UECAwHV2VzdGVybjEQMA4GA1UEBwwHQ29sb21ibzEQMA4GA1U
ECgwHTWVlcHVyYTELMAkGA1UECwwCSVQxEjAQBgNVBAMMCWxvY2FsaG9zdDE
iMCAGCSqGSIb3DQEJARYTOTJidWRkaGl2QGdtYWlsLmNvbTCCASIwDQYJKoZ
IhvcNAQEBBQADggEPADCCAQoCggEBAOiWVtI4eLVNMVCLgb9GJrRTBhiTxZ7
KtC/4ydIh4ZNyb5vy9ykfzFHbLQKsfvbPArJdlEHkl5vu+gCV9i8B1wsF2bH
/GUwtUFshzkV58w2PQByIfeh/G5YtHV2N8LLy7W/4QRWClN7HtGHyIJBG5Wk
rXHJRAzxsX4BzEgQZA4gXGagOxhamWSs2XGnIdP7IqeLe0xFz+1m0cwuVDIk
JbUnKYGQjLlX1Xo8tstrgcerZmXrcWcqIXswtN4Mm9azmigpmHHGGzynqqaP
4i8VSOpwSS62Mz0YEb2Y4+Y9zPvs7UOX3aJY/LiXjYMcWrNulU3/O4utvJiG
T0/L+ILVfXgkCAwEAAaNQME4wHQYDVR0OBBYEFNKWk9dscGiV2CLFMMrteh8
tvmNzMB8GA1UdIwQYMBaAFNKWk9dscGiV2CLFMMrteh8tvmNzMAwGA1UdEwQ
FMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAK/iPW5iTcVJNd44g2Uc+IV7XPC
IY6fi+Hv3+0LbEhtThmQOGi2Zf2yv9BiMP8eVuhew3ovqlgXVQqgq+WM+Dwe
ba7vEhGpadltYNPlrbpLi1PMEIi1rdJ4AN4ALwmWSpqUucU3/Z4B+fJVuJ/z
+XkR7tzaJzsQcjLK/NZhxj26Yl2KwthWRpXzq3ZSTMP1Jqv7C0nI597lTV+C
3yObY/JfxE0blUSv0DZrg8JrHeTgW9fd3+4+UhRtX7YW+SYpOCbZP/HrCQum
CwRkZ7xb3Rbp8vp3ol40p+ka5BBEBADhVjUfq649O66o9M3HCfkUYQ+lIhjN
kMOMoD0H9pP7wXE4=\n" +
            "-----END CERTIFICATE-----");
});

var privateKey = "-----BEGIN PRIVATE KEY-----\n" +

"MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQDollbSOHi
1TTFQi4G/Ria0UwYYk8WeyrQv+MnSIeGTcm+b8vcpH8xR2y0CrH72zwKyXZR
B5Jeb7voAlfYvAdcLBdmx/xlMLVBbIc5FefMNj0AciH3ofxuWLR1djfCy8u1
v+EEVgpTex7Rh8iCQRuVpK1xyUQM8bF+AcxIEGQOIFxmoDsYWplkrNlxpyHT
+yKni3tMRc/tZtHMLlQyJCW1JymBkIy5V9V6PLbLa4HHq2Zl63FnKiF7MLTe
DJvWs5ooKZhxxhs8p6qmj+IvFUjqcEkutjM9GBG9mOPmPcz77O1Dl92iWPy4
l42DHFqzbpVN/zuLrbyYhk9Py/iC1X14JAgMBAAECggEBAKMRJHXm2dpi8Hx
EEweDq4cp3lBE6nzWKVao2vbUgk7aIJ35zoeqn5mUTQ5e2fU4Ve+v5E3+cr0
E44qdmSiD5bz4sRQ2ggoCiyAZp4DWay3KjWxz1bK3yyOTJc99wI/1+bpTF25
5St6WrUUueN4ulpERsZMEcXxfjuWDx9HPp8Y09YYgfxW+VFioFCdjlnMMRBB
qWYbRS8FwU11k+Ai9sfHPhdCcPtLAF2V2brgpOonZtjuCo7nwsQs6Q33VToW
j64xE7fPlAvVPouSH4hp93iv/ULSvBWT6w/wmjWckGEinDg49/GuRv3XRZ5F
FmWnWugwmLQTpRyNJEi1J7lCEPfECgYEA/UbP/GJOSr1clm3AGQPrmcxTU0n
V9Hvrc1pt/2L0aPTU7h95/mJ2Ue4f8kVZBogp6gU9LZQdP37Mz41MgZWLreM
UyQSdI9KZH3G6uVRc/LNg5xDET70uyeun+dw1Io+5e6rlGaHVKERYoeSgTTI
+eeZCJNoOi5hhPKbq5wJRlsMCgYEA6xaTYQlhNEx8b1ElAklAMjn9FJPJVir
OKijfFOrE7ZpTvaOQ1PDC2paXY9ZcA3LMNgnGT7BSJS+sUc627XHYUuo6mp/
0ILsq+9qvth/tDyR0tENIrRCfZZGBxM+avC8/1C0TXQIyeB8uLxR89JSQQhV
H3zwQjUXH92GrcP/y40MCgYEA+hyJm0RA6FGjMvHid1GFwXUi++a4IByXYGx
```

```
2n3JKxbKw6w2uXOVCzpmGdqrAxVCFg5H03iOb1m4TNwrj+DuDmg3bIr8ppox
7pa+boxSKVwmUsdm+4reBkujiEj3BQwYHNvaGEw/a/U6w7/5jxpfNVndp7hZ
fsr6hl1GGOuXxSB0CgYEAluYX1dqidWJ/ISjx24TPWy4TwCiYvOGfEjrH7vI
/U9CS3hBmv/iG6q5tIJ2Q1HnUkP83NyGTqODv+Fb63nEMDTTiRyxTFMtvbNh
Tn1Dg5q5c5vSlaslXt2dt57nmtdKSYwxH+JSXdrl0+K1rA8d0zaZBSw6QBU5
8a9Naq57u9mcCgYAanz5/Rw1VUbcViRMVueuopdPo6hgNv/9ciBsgOqhxq6s
rtoPcFEo3fNU5v5pdKQGaI8hfkjMhR4sYw3JbWcB7JIJTtjCJvUuDUJNrf62
+couuCX7WQUrBq7HVOtaFD92P86d6JGqjNYSYMarSAlMCgd0TtJKPK/gu7xD
NhdWahQ==\n" +
        "-----END PRIVATE KEY-----";

qz.security.setSignaturePromise(function (toSign) {
    return function (resolve, reject) {
        try {
            var pk = new RSAKey();

pk.readPrivateKeyFromPEMString(strip(privateKey));
            var hex = pk.signString(toSign, 'sha1');
            console.log("DEBUG: \n\n" +
stob64(hextorstr(hex)));
            resolve(stob64(hextorstr(hex)));
        } catch (err) {
            console.error(err);
            reject(err);
        }
    };
});

function strip(key) {
    if (key.indexOf('-----') !== -1) {
        return key.split('-----')[2].replace(/\r?\n|\r/g,
'');
    }
}
```

- With this, you will get a popup that you can check “Remember this decision” and make it go away forever.



Certificate

Field	Value
● Organization	QZ Industries, LLC
● Common Name	QZ Industries, LLC
● Trusted	Verified by QZ Industries, LLC
● Valid From	2015-06-14 17:00:00
● Valid To	2018-06-15 17:00:00
● Fingerprint	f6aaa955f7cc4072b7f0e5c8f900828e98b8c...

The certificate details will be like this after successfully doing the steps.

- There are examples for other languages in here. You can accomplish this over any preferred language.

7. Trusted certificate should be allowed only once and you are good to go with qz tray.

Happy coding :)