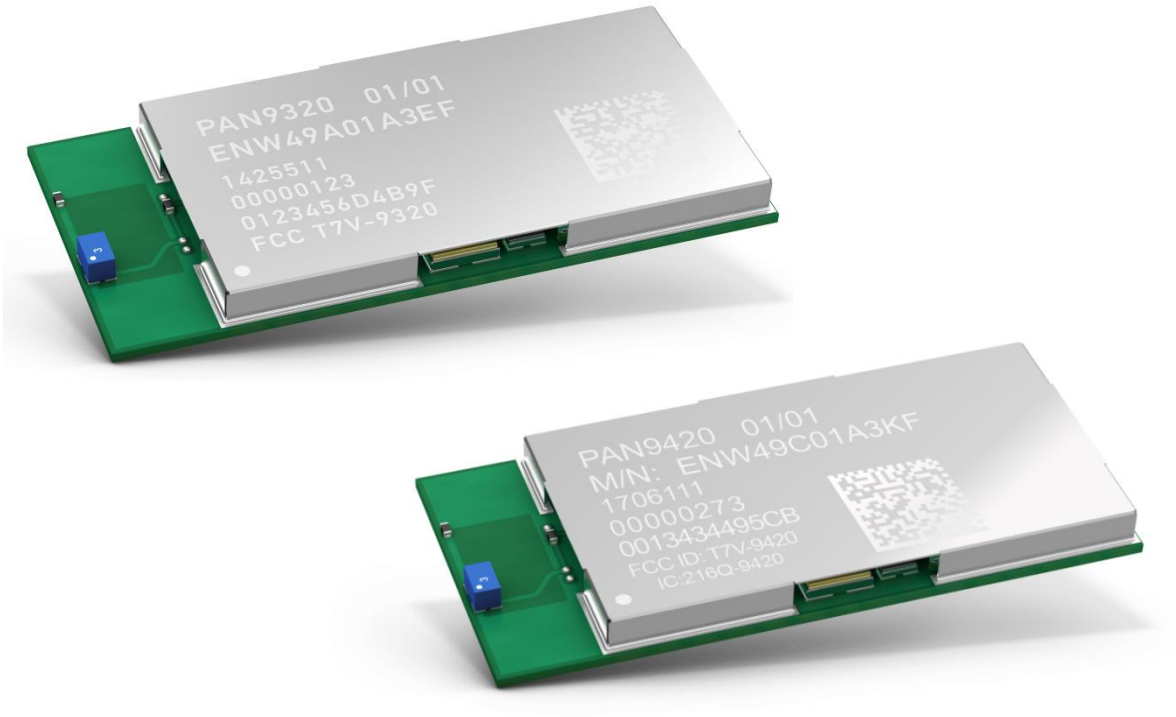
Panasonic

PAN9320/PAN9420

HTTP(S) Client Usage

Application Note

Rev. 1.0



Wireless Modules

By purchase of any of the products described in this document the customer accepts the document's validity and declares their agreement and understanding of its contents and recommendations. Panasonic Industrial Devices Europe GmbH (Panasonic) reserves the right to make changes as required at any time without notification.

This Application Note does not lodge the claim to be complete and free of mistakes.

The information contained herein is presented only as guidance for Product use. No responsibility is assumed by Panasonic for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.

Description of hardware, software, and other information in this document is only intended to illustrate the functionality of the referred Panasonic product. It should not be construed as guaranteeing specific functionality of the product as described or suitable for a particular application.

Any provided (source) code shall not be used or incorporated into any products or systems whose manufacture, use or sale is prohibited under any applicable laws or regulations.

Any outlined or referenced (source) code within this document is provided on an “as is” basis without any right to technical support or updates and without warranty of any kind on a free of charge basis according to § 516 German Civil Law (BGB) including without limitation, any warranties or conditions of title, non-infringement, merchantability, or fitness for a particular purpose. Customer acknowledges that (source) code may bear defects and errors.

The third-party tools mentioned in this document are offered by independent third-party providers who are solely responsible for these products. Panasonic has no responsibility whatsoever for the performance, product descriptions, specifications, referenced content, or any and all claims or representations of these third-party providers. Panasonic makes no warranty whatsoever, neither express nor implied, with respect to the goods, the referenced contents, or any and all claims or representations of the third-party providers.

To the maximum extent allowable by Law Panasonic assumes no liability whatsoever including without limitation, indirect, consequential, special, or incidental damages or loss, including without limitation loss of profits, loss of opportunities, business interruption, and loss of data.

Table of Contents

1 About This Document

1.1 Purpose and Audience

The purpose of this document is to explain how to set up and perform HTTP(S) client requests with the PAN9320/PAN9420 Wi-Fi modules using a local Apache web server.

The PAN9320/PAN9420 Wi-Fi modules are referred to as “the PAN9x20” or “the module” within this document.

The document is first and foremost intended for software engineers. However, it is generally useful for anyone wanting to understand the setup of the HTTP(S) client and the requests.

1.2 Revision History

Revision	Date	Modifications/Remarks
0.1	13.06.2017	Initial draft
0.2	10.07.2017	Updated document title, disclaimer, header, footer and the document version of related documents
1.0	16.08.2017	Added chapter “Module Basics” and configuration file description in chapter “Enabling HTTP(S) Communication”, editorial review

1.3 Use of Symbols

Symbol	Description
	**Note** Indicates important information for the proper use of the product. Non-observance can lead to errors.
	**Attention** Indicates important notes that, if not observed, can put the product’s functionality at risk.
	**Tip** Indicates useful information designed to facilitate working with the software.
⇨ [chapter number] [chapter title]	**Cross reference** Indicates cross references within the document. **Example:** Description of the symbols used in this document ⇨ 1.3 Use of Symbols.
✓	**Requirement** Indicates a requirement that must be met before the corresponding tasks can be completed.
➔	**Result** Indicates the result of a task or the result of a series of tasks.

Symbol	Description
This font	**GUI text** Indicates fixed terms and text of the graphical user interface. **Example:** Click **Save**.
Menu > **Menu item**	**Path** Indicates a path, e.g. to access a dialog. **Example:** In the menu, select **File** > **Setup page**.
`This font`	**File names, messages, user input** Indicates file names or messages and information displayed on the screen or to be selected or entered by the user. **Examples:** `main.c` contains the actual module initialization. The message `Failed to save your data` is displayed. Enter the value `Product 123`.
[Key]	**Key** Indicates a key on the keyboard, e.g. **[F10]**.

1.4 Related Documents

[1] Panasonic. PAN9320 Communication Specification Firmware V1.9.6.4

[2] Panasonic. PAN9420 Communication Specification Firmware V1.1.0.6

Please refer to the Panasonic website for more related documents ⇨ 10.1.2 Product Information.

2 Introduction

The PAN9x20 hosts an HTTP(S) client which enables an application to communicate with web services and remote APIs. These web services and APIs are mostly not in the hand of the embedded software engineer or not ready to be used, hence it can be helpful to have a webserver for testing and development purposes.

As there are several different types of web servers with different setup procedures and configurations, it is beyond the scope of this document to explain them all in detail. Thus, this document simply describes how to perform HTTP(S) GET and POST requests to a local web server. The setup and configuration can then be transferred to other web servers accordingly.

3 Prerequisites

This chapter describes which prerequisites have to be met to successfully work through the remaining document.

3.1 Web Server

To perform HTTP(S) client requests from the PAN9x20 to a web server it is inevitable to have some sort of web server. This server can either be online or just an offline local test server. In this document an Apache web server from within XAMPP is used.

3.2 Certificates

If a secure HTTP communication is desired, certificates are necessary. Hence, to create keys, certificate signing requests and certificates a TLS implementation is required. In this document OpenSSL is used to take over this task.

3.3 Terminal Software

When using a PAN9x20 ETU in combination with a motherboard, the module can be controlled from a computer using a terminal, such as

- HTerm
- Tera Term or
- HyperTerminal.

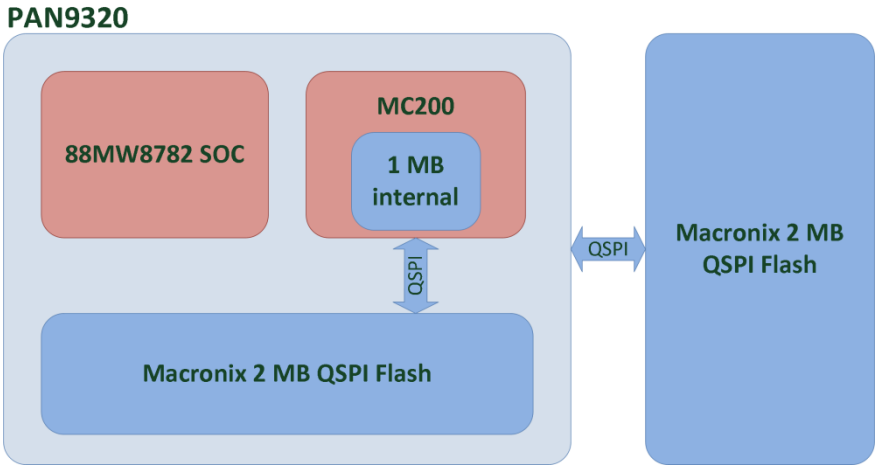
4 Module Basics

This chapter describes important PAN9x20 basics, which are essential to understand in order to successfully create update files.

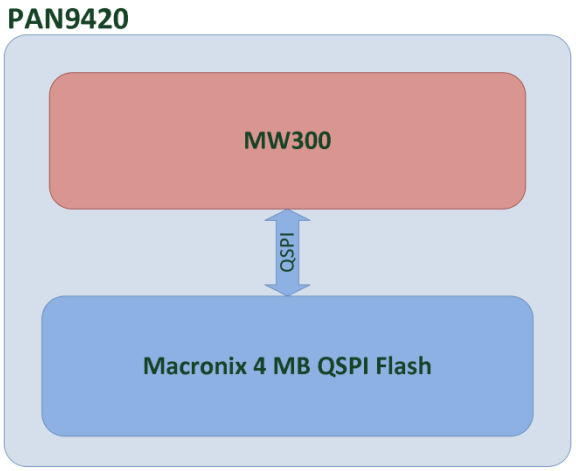
4.1 Memory Layout

The PAN9320 has 3 MB of internal flash memory, of which 1 MB is available for user content such as configuration files, webpages, certificates, etc. and 2 MB are reserved for module firmware and caching firmware uploads.

If more than 1 MB of flash memory is required for user content, additional 2 MB of external flash can be added to the application using the PAN9320 QSPI interface.



The PAN9420 has 4 MB of internal flash memory, of which 2 MB are available for user content and the remaining 2 MB are for reserved for internal usage.



4.2 File System

The PAN9x20 utilizes a file system in order to systematically store and find resources such as webpage files, certificates and others. The file system is used by the HTTP(S) webserver as well as the Transport Layer Security (TLS) of various software modules. The webserver uses it to search for webpage files, whereas the TLS uses it to search for valid certificates.

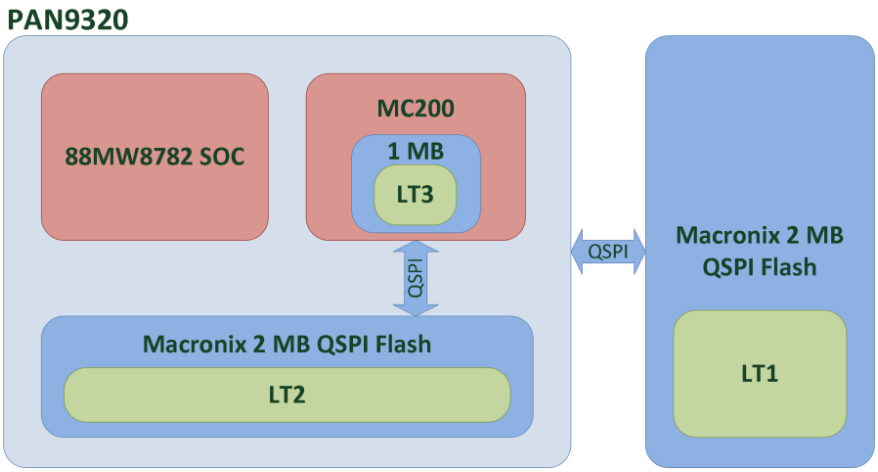
The user-editable folders in the file system are listed and described below:

Path	Description
/web	Websites and resources which are accessible via webserver. Note: The filename including the path must not exceed 80 characters.
/cert	Location for trusted webserver certificates. The certificates are used from the mail and HTTP(S) client module to establish TLS connections. Note: The file name of the certificate must be the subject key identifier of the certificate and it must have the .der format (e.g. 949c41491e583f48f20206dc743693c6f322ec9a.der)
/cert/own	Location for the PAN9x20 webserver certificates. Note: The certificate name must be `pan9320.der` respectively `pan9420.der` and the private key name must be `pan9320.key` respectively `pan9420.key`.
/cert/fwu	Location for trusted firmware update servers. Note: The file name of the certificate must be the subject key identifier of the certificate and it must have the .der format (e.g. 949c41491e583f48f20206dc743693c6f322ec9a.der)

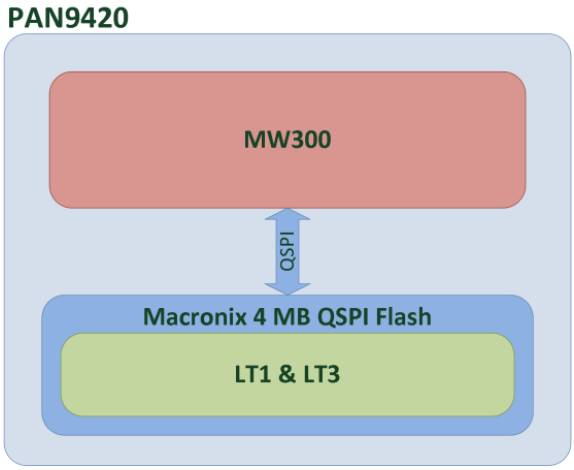
Assuming that external flash is connected, the file system will search for resources in all three file partitions, with the following prioritization:

1. The first search starts in LT 1 (look-up table).
2. If the data is not in LT1, search in LT2.
3. If the data is not in LT1 and LT2, load LT3.

To show the different locations of look-up tables in the PAN9320 memory, the initial memory layout can be extended as follows:



The memory layout of the PAN9420 can be extended as follows:



On the PAN9420 there is no LT1 due to missing external memory. In this case a resource search starts in LT2.

Beside the mentioned searching mechanism, it is possible to address PAN9x20 resources directly using a URL with a prefix.

Prefix	Flash Memory PAN9320	Flash Memory PAN9420
LT1/	External	Internal (PAN9420 module)
LT2/	Internal (PAN9320 module)	
LT3/	Internal (PAN9320 MC200 MCU)	Internal (PAN9420 module)

Example:

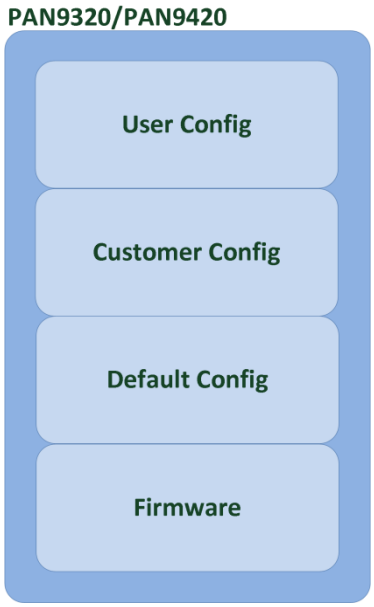
To directly access the PAN9x20 fallback webpage, enter the following URL into the browser:

http://192.168.1.1/LT3/index.html

4.3 Configuration Layout

The configuration of the PAN9x20 consists of three configuration options.

- Default Config: Contains basic configuration parameters.
- Customer Config: Configuration parameters set by the customer.
- User Config: Configuration parameters set by the user at runtime.



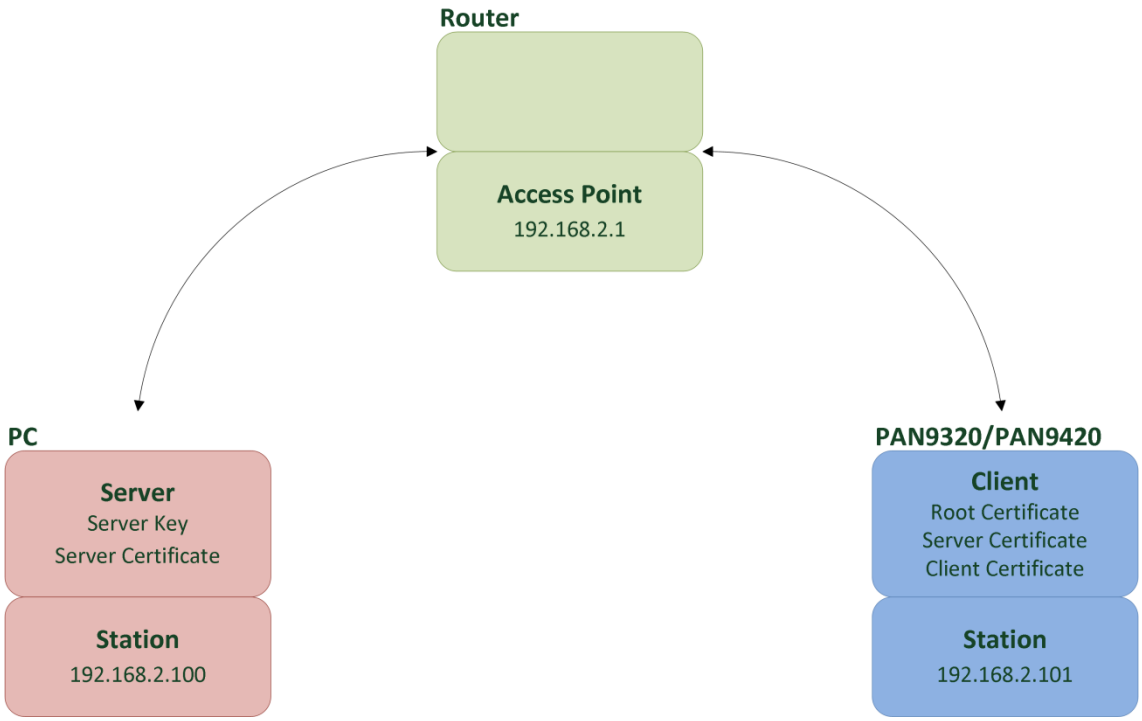
The basis of the PAN9x20 configuration layout is the module firmware with the bootloader and radio driver. The behavior of this firmware can be configured over three configuration layers of which only two can be used by the customer. The default configuration is done by Panasonic in order not to endanger the certification of the module and the end application.

The customer configuration enables to customize most of the PAN9x20 default settings of access point, station, UART interfaces, GPIOs, HTTP(S) client, firmware update server and others.

The user configuration is the configuration which is done by host and user at runtime and can address most of the previously mentioned settings depending on the user rights.

5 HTTP(S) Communication Setup

This chapter describes and depicts the local test setup used for the HTTP(S) test communication. The IP addresses shown in the figure below should help to understand the configuration of the setup process described in the following chapters. In a proper application this setup would obviously differ, while the server would be online and there would be multiple clients sending and requesting information.



As depicted above, a computer is running a server application, which again is hosting the resources for the HTTP(S) communication. This computer is connected to a network in order to serve its services to other computers and clients. From the security point of view, the server uses its server key as well as its server certificate to provide communication security using TLS.

On the client side, the PAN9x20 is running a client application sending and requesting data from the server. This node is connected to the network the server is on.

The full setup can be operated in a secure as well as in an insecure mode. This document will only show the secure mode as the insecure mode is much easier and self-evident.

6 Creating Certificates

In the following sections, OpenSSL is used to create a test Certification Authority (CA) as well as generate test certificates for the server and the client.

The process of creating a certificate is similar for all the three different instances. First, a private key is generated. Then, a certificate signing request (CSR) is generated, which contains the public key, identifying information and integrity protection. Finally, the certificate itself will be generated.

The configuration and extension files used during the process are attached to this document.

6.1 Certification Authority

Create a private key:

```
openssl genrsa -out ca.key 1024
```

Create the CSR using a config file:

```
openssl req -config ca.config -new -key ca.key -out ca.csr
```

Create the x509 certificate using an extension file (x509v3) and output the certificate as well as the extension information such as the subject key identifier.

```
openssl x509 -days 1095 -extfile ca.ext -signkey ca.key -in ca.csr -req -out ca.crt –text
```

Copy the subject key identifier from the output, convert the upper case characters to lower case characters, remove the colons and save the identifier to an additional file. It will be used in a bit.

Convert the certificate from the CRT format to the DER format.

```
openssl x509 -outform der -in ca.crt -out ca.der
```

Now, rename the certificate with the DER format with the stored subject key identifier.

6.2 Server Certificate

Create a private key:

```
openssl genrsa -out server.key 1024
```

Create the CSR using a config file:

```
openssl req -config server.config -new -key server.key -out server.csr
```

Create a file which stores the serials of the certificates issued by the CA:

```
echo -ne '01' > ca.serial
```

Create the x509 certificate using an extension file (x509v3) and output the certificate as well as the extension information such as the subject key identifier.

```
openssl x509 -days 730 -extfile server.ext -CA ca.crt -CAkey ca.key -CAserial ca.serial -in server.csr -req -out server.crt -text
```

Copy the subject key identifier from the output, convert the upper case characters to lower case characters, remove the colons and save the identifier to an additional file. It will be used in a bit.

Convert the certificate from the CRT format to the DER format.

```
openssl x509 -outform der -in server.crt -out server.der
```

Now, rename the certificate with the DER format with the stored subject key identifier.

6.3 Client Certificate

Create a private key:

```
openssl genrsa -out client.key 1024
```

Create the CSR using a config file:

```
openssl req -config client.config -new -key client.key -out client.csr
```

Create the x509 certificate using an extension file (x509v3) and output the certificate as well as the extension information such as the subject key identifier.

```
openssl x509 -days 730 -extfile client.ext -CA ca.crt -CAkey ca.key -CAserial ca.serial -in client.csr -req -out client.crt -text
```

Copy the subject key identifier from the output, convert the upper case characters to lower case characters, remove the colons and save the identifier to an additional file. It will be used in a bit.

Convert the certificate from the CRT format to the DER format.

```
openssl x509 -outform der -in client.crt -out client.der
```

Now, rename the certificate with the DER format with the stored subject key identifier.

7 Enabling HTTP(S) Communication

In order to enable the HTTP(S) communication, there are the following two approaches.

- API commands
- Customer configuration file

Depending on the save mode configuration (auto or manual) of the PAN9x20, which again depends on the use case, one option might be selected over the other.

Since the use of the API commands is straight forward and described later in the document, only the configuration with the customer configuration file will be described in this chapter.

The firmware file required to enable secure HTTP communication consists of two binary files, which will be generated separately first and combined later on. The binary files contain the customer configuration file with the HTTP(S) client configuration and the folder **cert** containing the CA, server and client certificates in DER format, named with the subject key identifier from within the certificates.

These are the detailed steps to generate the customer configuration binary file (if desired):

1. Amend the section [HTTP-CLIENT] according to your needs.

   Here is an example of the settings used for a local server:

```
[HTTP-CLIENT]
server_adr = 192.168.2.100
server_port = 443
server_ressource = Get.php
server_login = ""
server_password = ""
token = ""
# 0 = off 1= auto 2= always TLS
security = 2
# 1=off 2=on
state = 2
```

   The attribute server_adr can either be an IP address as shown above or a DNS name.

2. Generate a binary file using the `CustomerConfig.exe` and the following command:

```
CustomerConfig.exe –i customer.cfg
```

These are the detailed steps to generate the binary file containing the certificates (if desired):

1. Create a folder named **cert**.
2. Inside this folder place the CA, server and client certificate named with the subject key identifier of the certificate in lower case letters and in the DER format.
3. Create a binary file using the tool `Dir2Flash.exe` and the following command:

```
Dir2Flash.exe --name=serialflash.bin --input-path=.\ --output-path=.\
```

These two binary files need to be combined to the final firmware update file. In order to do that, create an additional file called `httpc.txt`. This file states where the files need to be stored in flash.

```
CustomerCfg.bin 0x0102 1 0
serialflash.bin 0x0101 2 0
```

The structure of this configuration file is: **file name**, **file type**, **file ID** and **version number**, which can be used multiple times to combine multiple files. A detailed description of the parameters can be found below.

Parameter	Description
File name	The file name including the path.
File type	The type of the file (firmware, configuration file, etc.).
File ID	The ID of the file. This specifies the desired location of the file in the memory.
Version	The version number of the file (currently not used by the tool `fwgen.exe`).

The possible file types and IDs can be found in the table below.

File Type	Description
0x0003	Host controller firmware. No file ID is used.
0x0101	Flash image. Images must by generated with `Dir2Flash.exe`. File IDs PAN9320: 0 = Internal Flash (Size: 987 KB) 1 = External Flash (Size: 2 MB) File IDs PAN9420: 0 = Internal Flash (Size: 2 MB)
0x0102	Configuration file. Config files must be generated with `CustomerConfig.exe`. File ID: 1 = Customer configuration file

Finally, create the firmware update file using the tool `fwgen.exe` with the following command for the PAN9320.

```
fwgen.exe -v 0.0.0.1 -c "Panasonic GmbH" -p PAN9320 -i httpc.txt
```

For the PAN9420 please use the following command.

```
fwgen.exe -v 0.0.0.1 -c "Panasonic GmbH" -p PAN9420 -i httpc.txt
```

Please note that the specified version (e.g. –v 0.0.0.1) has to be equal or higher than the one on the module. Otherwise, the module will not accept the file as a valid update file.

The current version can be requested using the following command:

```
get system version<\r><\n>
```

Now, the only thing left to do is to upload this file onto the PAN9x20 using the webserver address http://192.168.1.1/LT3/index.html.

8 XAMPP Apache Web Server

1. Install XAMPP with Apache – All other services are not used for this set up.
2. Go to **C:\xampp\apache\conf** and place the server certificate, CSR and private key in the appropriate folders.
3. Go to **C:\xampp\htdocs** and place the accompanying `Get.php` and `Post.php` files in the folder. These files simply read the HTTP(S) request header and return them to the module. The `Post.php` file additionally returns the posted data.
4. Go to **C:\xampp\apache\conf**, open the file `httpd.conf` and bind Apache to the specific IP address and port in order to prevent Apache from listening to all bound IP addresses:

```
#Listen 80
Listen 192.168.2.100:443
```

5. Go to **C:\xampp\apache\conf\extra**, open the file `httpd-vhosts.conf` and add the following lines with the appropriate settings for your host to the end of the file:

```
<VirtualHost *:443>
        ServerAdmin admin@testserver.com
         DocumentRoot "C:/xampp/htdocs/"
        ServerName 192.168.2.100
        ErrorLog "logs/testserver-error.log"
        CustomLog "logs/testserver-access.log" common
        SSLEngine on
        SSLCertificateFile "C:/xampp/apache/ssl.crt/server.crt"
        SSLCertificateKeyFile "C:/xampp/apache/ssl.key/server.key"
</VirtualHost>
```

6. Connect the computer running the server to the local network.
7. Start the Apache server.

Please note that the specified server name (192.168.2.100) has to match the actual IP address of your server.

9 Performing HTTP(S) Requests

This chapter describes how to set up the PAN9x20 and its HTTP(S) client as well as how to perform HTTP(S) requests to a server.

The PAN9x20 currently offers two modes for transferring an HTTP(S) payload:

1. The payload can be handed to the module in the request command using the CMD UART of the PAN9x20. This mode has a payload limitation of 299 bytes.
2. The payload is handed to the module using the BIN UART of the PAN9x20. This mode does not have any payload limitation.

Both modes will be covered for GET and POST requests in the following sections.

9.1 Setting up the PAN9x20

In order to actually perform HTTP(S) requests, the following tasks have to be taken care of first:

1. The module has to be connected to the network the server is on. If the server is online, the module has to be connected to a network with an internet connection.
2. The HTTP(S) client of the module has to be set up according to the application needs, when this has not already been done in the customer configuration file.
3. The time module of the PAN9x20 has to be properly set up in order for the module to validate the server certificate.

In order to connect the PAN9x20 to a local network, the WLAN station of the module has to be configured and the station has to be turned on.

Depending on the hardware setup used, the commands to perform these actions might reach the modules CMD UART from different sources. If a PAN9x20 ETU is connected via USB to a PC, the commands can be sent using a terminal, such as HTerm. If an embedded setup is used the commands have to be sent using a host controller.

Configuring the PAN9x20 wlan station

Template:	set wlan cfg sta <ssid> <psk> <security><\r><\n>
Example:	set wlan cfg sta testnetwork password 4<\r><\n>

Turning on the PAN9x20 wlan station

```
set wlan state sta on<\r><\n>
```

After the module has successfully connected to the network, the HTTP(S) client has to be set up.

9.2 Setting up the HTTP(S) Client

The PAN9x20's HTTP(S) client API offers four variables for configuration.

Variable	Description
Server	Configures which server the HTTP(S) client should connect to and with which encryption state.
Resource	Configures which resource on the server the HTTP(S) client should request.
Token	Configures in which path the resource is located (<token>/<resource>).
State	Configures the state of the client.

All these variables can be set by a host as shown in the following configuration example:

Server configuration – IP address: 192.168.2.100, Port: 443: Security: off

Template:	set httpc server <server-addr> <port> <security> <username> <password><\r><\n>
Example:	set httpc server 192.168.2.100 443 on<\r><\n>

Resource configuration – Get.php

Template:	set httpc resource <resource><\r><\n>
Example:	set httpc resource Get.php<\r><\n>

Token configuration – None

Template:	set httpc token <token><\r><\n>
Example:	set httpc token “”<\r><\n>

State configuration – On

Template:	set httpc state <state><\r><\n>
Example:	set httpc state on<\r><\n>

It is recommended to check the status of the HTTP(S) client after the configuration has been made to make sure the client was set up properly.

Request HTTP(S) client status

Request:	get httpc status<\r><\n>
Response:	get httpc status 0 1<\r><\n>

The different statuses can be found in the following status table.

Return Value	Description
0	Not configured
1	Ready to send and receive data
2	Data is sending
3	Error

9.3 Performing HTTP(S) GET Requests

The resource `Get.php` on the server is used for testing HTTP(S) GET requests. This script simply returns the HTTP(S) header. The resource has to be addressed using the HTTP(S) resource command as previously described:

Resource configuration – Get.php

```
set httpc resource Get.php<\r><\n>
```

9.3.1 GET Request via CMD UART

To perform a GET request using the CMD UART the following command is used:

Initiate HTTP(S) GET request via CMD UART

```
set httpc get<\r><\n>
```

In order to get the requested data from the module, the following command is used:

Get HTTP(S) GET data via CMD UART

```
get httpc get<\r><\n>
```

Considering the previous configuration and setup including the resource on the server (`Get.php`), the data received from the server on the CMD UART looks like this:

```
<\n>
PAN9X20 GET Test<\n>
Host: 192.168.2.100<\n>
User-Agent: HTTP-Client<\n>
```

9.3.2 GET Request via BIN UART

To perform a GET request using the BIN UART the following command is used:

Initiate HTTP(S) GET request via CMD UART

```
send httpc get<\r><\n>
```

The advantage of using the BIN UART for HTTP(S) communication is, that the GET data does not have to be prompted using a special command. The PAN9x20 will inform the host via the CMD UART about received data on the BIN UART.

HTTP(S) received data information via CMD UART

Template:	info httpc header <status-code> <mode> <datasize><\r><\n>
Template:	info httpc status <status> <datasize><\r><\n>
Example:	info httpc header 200 0 62<\r><\n>
Example:	info httpc status 1 62<\r><\n>

Considering the previous configuration and setup including the resource on the server (`Get.php`), the data received from the server on the BIN UART looks like this:

```
<\n>
PAN9X20 GET Test<\n>
Host: 192.168.2.100<\n>
User-Agent: HTTP-Client<\n>
```

9.4 Performing HTTP(S) POST Requests

The resource `Post.php` on the server is used for testing HTTP(S) POST requests. This script simply returns the HTTP(S) header and the posted data.

The resource has to be addressed using the HTTP(S) resource command:

Resource Configuration – Post.php

```
set httpc resource Post.php<\r><\n>
```

9.4.1 POST Request via CMD UART

To perform a POST request using the CMD UART the following command is used:

Initiate HTTP(S) POST Request via CMD UART

Template:	set httpc post <data><\r><\n>
Example:	set httpc post TEST<\r><\n>

In order to get the response to this post from the module, the following command is used:

Get HTTP(S) Response via CMD UART

```
get httpc get<\r><\n>
```

Considering the previous configuration and setup including the resource on the server (`Post.php`), the data received from the server looks like this:

```
<\n>
PAN9X20 POST Test<\n>
Content-Type: text/plain<\n>
Host: 192.168.2.100<\n>
Transfer-Encoding: chunked<\n>
User-Agent: HTTP-Client<\n>
Data: TEST<\n>
```

9.4.2 POST Request via BIN UART

To perform a POST request using the BIN UART the following command is used:

Initiate HTTP(S) POST Request via CMD UART

```
Template:    send httpc post <datasize><\r><\n>
Example:     send httpc post 451<\r><\n>
```

As with the BIN UART GET request, the response does not have to be prompted using a special command. The PAN9x20 informs the host via the CMD UART about received data on the BIN UART.

HTTP(S) received response via CMD UART

```
info httpc header 200 0 573<\r><\n>
info httpc status 1 573<\r><\n>
```

Considering the previous configuration and setup including the resource on the server (`Post.php`), the data received from the server on the BIN UART looks like this:

```
<\n>
PAN9X20 POST Test<\n>
Content-Type: text/plain<\n>
Host: 192.168.2.100<\n>
Transfer-Encoding: chunked<\n>
User-Agent: HTTP-Client<\n>
Data: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.<\n>
```

10 Appendix

10.1 Contact Details

10.1.1 Contact Us

Please contact your local Panasonic Sales office for details on additional product options and services:

For Panasonic Sales assistance in the **EU**, visit

https://eu.industrial.panasonic.com/about-us/contact-us

Email: wireless@eu.panasonic.com

For Panasonic Sales assistance in **North America**, visit the Panasonic Sales & Support Tool to find assistance near you at

https://na.industrial.panasonic.com/distributors

Please visit the **Panasonic Wireless Technical Forum** to submit a question at

https://forum.na.industrial.panasonic.com

10.1.2 Product Information

Please refer to the Panasonic Wireless Connectivity website for further information on our products and related documents:

For complete Panasonic product details in the **EU**, visit

http://pideu.panasonic.de/products/wireless-modules.html

For complete Panasonic product details in **North America**, visit

http://www.panasonic.com/rfmodules