zApp API

Application Programming Interface
for Software Version 01.08.18

10.ZEPAPI-E.1612 / Version 1.0

by Schneider Electric

1 Introduction

1.1 How zApp works

On each zApp device which is connected with a zeptrion actuator to control the lights or blinds, a Web-Server is running. On top of this Web-Server we provide a simple web service API as interface. Use it as your own tools! We hope this will help you to truly use zeptrion actuator as you want by making new apps, websites and digital installations. Let’s integrate zeptrionAIR into something else or just playing around!

1.2 Acronyms

zApp	zeptrionAir embedded device software
zrap	zeptrionAir remote access protocol
DNS	Domain Name System
mDNS	multicast Domain Name System
URL	Uniform Resource Locator
NTP	Network Time Protocol
Percent-encoding	also known as URL encoding like”%3A” for ‘:’ or a ’+’ for ‘’
HTML character entities	also known as ampersand encoding like “&gt” for ‘>’
XML	Extensible Markup Language
JSON	JavaScript Object Notation
HTML	Hypertext Markup Language

1.3 Supported Device

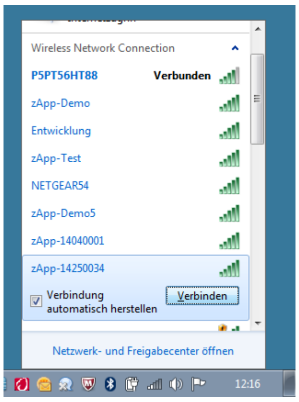
Product Name	Device ID	Projekt Name	
WLAN-Nebenstelle 4K **WLAN-Nebenstelle 4K V2**	**3340-4-A** **3340-4-B**	**zApp-Gateway** **zApp-Gateway**	
WLAN-Zwischenmodul 2K **WLAN-Zwischenmodul 2K V2**	**3340-2-A** **3340-2-B**	**zApp-Booster** **zApp-Booster**	

2 Getting started

First make sure that your zeptrionAir device is running as access-point with an SSID like “zApp-14250034” (factory default mode). To force factory defaults you can press the reset button for 5 seconds and wait until the green LED is blinks once per second. THIS WILL DELETE ALL SETTINGS IN THIS DEVICE! The fastest way to learn how to control a zeptrionAir device is to use a simple REST client to send GET and POST requests to it. In the following examples we use the “Advanced REST client for Google Chrome browser”:
https://code.google.com/p/chrome-rest-client/
You can also use Firefox with an Add-on like “HttpRequester”:
https://addons.mozilla.org/de/firefox/addon/httprequester/

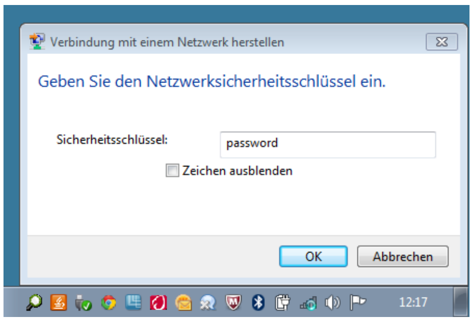
2.1 Step 1

Connect your PC directly to the zeptrionAir device by using the Wireless Network Connection window.


Wireless Network Connection
P5PT56HT88 Verbunden
zApp-Demo
Entwicklung
zApp-Test
NETGEAR54
zApp-Demo5
zApp-14040001
zApp-14250034
Verbindung automatisch herstellen
Verbinden
Netzwerk- und Freigabecenter öffnen
12:16

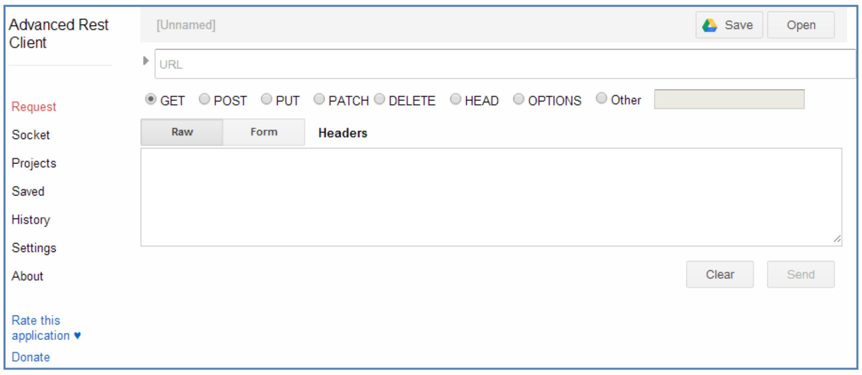
2.2 Step 2

If you create a connection the first time you have to enter the default WPA2-Password ‘*password*’.


Verbindung mit einem Netzwerk herstellen
Geben Sie den Netzwerksicherheitsschlüssel ein.
Sicherheitsschlüssel: password
Zeichen ausblenden
OK Abbrechen
12:17

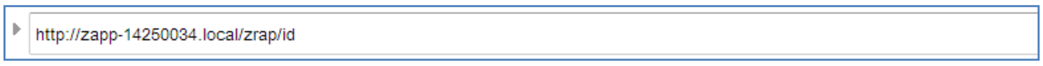
2.3 Step 3

Now start up the Advanced Rest Client.



2.4 Step 4

Write the following URL (see example below), select the `GET` method and press Send.

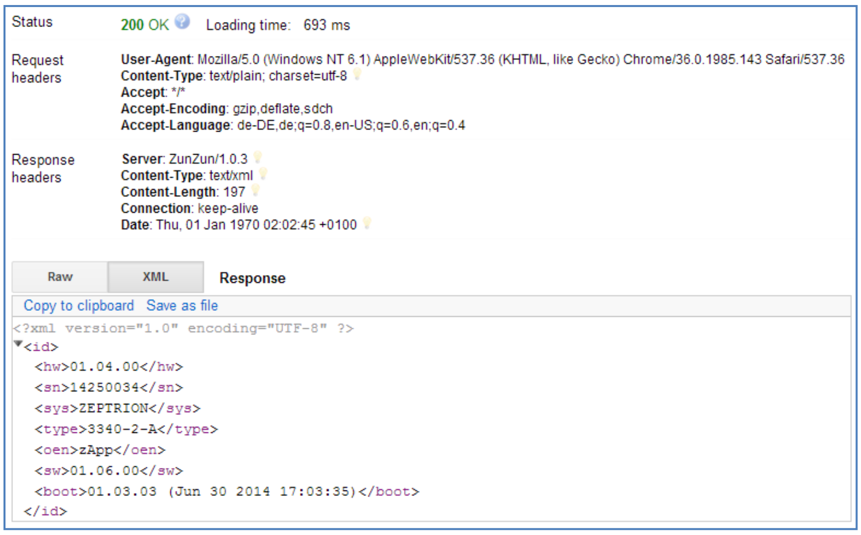


```
http://zapp-14250034.local/zrap/id
```

The host name is unique for each device. It is identical to the SSID (`zApp-14250034` in this example), except that SSIDs are case-sensitive and host names are not. This name is also printed onto the product label. The DNS top-level domain is always `.local`. You can also use the IP address which is always "192.168.0.1" if the device is in access-point mode. An URL path of a zeptrion service always starts with `/zrap/` or `/zapi/` followed by the service or resource name. The `zrap` stands for zeptrion remote access protocol.

2.5 Step 5

You get a response like this:



```
Status            200 OK    Loading time:  693 ms

Request headers   User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36
                  Content-Type: text/plain; charset=utf-8
                  Accept: */*
                  Accept-Encoding: gzip,deflate,sdch
                  Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4

Response headers  Server: ZunZun/1.0.3
                  Content-Type: text/xml
                  Content-Length: 197
                  Connection: keep-alive
                  Date: Thu, 01 Jan 1970 02:02:45 +0100
```

```
<?xml version="1.0" encoding="UTF-8" ?>
▼<id>
   <hw>01.04.00</hw>
   <sn>14250034</sn>
   <sys>ZEPTRION</sys>
   <type>3340-2-A</type>
   <oen>zApp</oen>
   <sw>01.06.00</sw>
   <boot>01.03.03 (Jun 30 2014 17:03:35)</boot>
 </id>
```

To play with other services check out chapter 3 Webservice API.

Hint: If you type just the IP address or DNS name into your browser URL field, you will see a simple web interface. These web pages are implemented in JavaScript and use some of the /zrap services!

3 Webservice API

3.1 Core concept

zApp API is built around the idea that everything has a unique URL served by the zeptrion WLAN device. Interacting with these URLs lets you modify them or find out their current state as explained above.

3.1.1 zApp web addresses

A zApp resource web address will always start with the following.

`http://<IP address or DNS name>/zrap`

This is the Root-URL for your app or controller to talk with the zeptrion WLAN device interface.

3.1.2 zApp resources

There are different kinds of resources to interact with where we've grouped those together with similar attributes. For example the `*/id` resource which contains all device information attributes or the `*/net` resource which contains all network settings.
We will add more attributes as we add features to the system.
You can query resources available in your zeptrion WLAN device by doing a `GET` on its local URL. For example the following returns all network attributes from your device

Address	`http://<ip address or DNS name>/zrap/net`
Methode	`GET`

After this `GET` request you'll get a response with the HTML status code 200 and a body in XML format. If you make a request on an invalid resource you will get a response with a HTML status code bigger or equal 400.

200 status Response Body	`<?xml version="1.0" encoding="US-ASCII"?>` `<ip>192.168.0.1</ip>`

3.1.3 Changing a resource attribute

The principle for changing an attribute of a resource is to send a POST request. The desired new value is attached to the request in the 'message body' with Percent-encoding (also known as URL encoding) format. For example to change the name of a channel `/chdes/ch1` we address the instance of this resource `/chdes` and send the new attribute value with the request in the message body.

Address	`http://<ip address or DNS name>/zrap/chdes/ch1`
Body	`name=My New Channel`
Methode	`POST`

If you are doing something that isn't allowed (invalid value, unknown resource name, etc.) then you will get a HTML status code bigger or equal 400 as response.

3.2 Webservice Encoding

3.2.1 URL encoding (Percent-encoding)

A byte must be URL encoded if the byte-value of a symbol is bigger as 126 or if it a reserved character.
In the table below we have the reserved characters, that must be URL encoded.

!	#	$	&	‘	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

Read more here: http://en.wikipedia.org/wiki/Percent-encoding

3.2.1.1 POST Example with URL Encoded Characters

Set in channel descriptor the name “Küche <Deckenspots & Wandlampe>”	
Resource URL	http://<IP Address or DNS Name>/zrap/chdes/ch1
Body Data	`name=K%C3%BCche+<Deckenspots+%26+Wandlampe>`

3.2.2 Ampersand encoding

In all GET services you will receive an XML string. The XML specification defines five "predefined entities" representing special characters, and requires that all XML processors honor them.
Read more here: http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references

3.2.2.1 GET Example with predefined ampersand entities

Get from channel descriptor the attribute value name	
Resource URL	http://<IP Address or DNS Name>/zrap/chdes/ch1/name
Response Body	

```
<?xml version="1.0" encoding="UTF-8"?>
<name>
  Küche &lt;Deckenspots &amp; Wandlampe&gt;
</name>
```

3.3 System Commands -> zrap/sys

3.3.1 Description

With this service you can reboot a zApp device. After some configuration changes a reboot of the zApp device is necessary. This is required for example after network changes so that the zApp device starts up with the new network settings. You can also use this service to bring the device into factory default mode.

3.3.2 Resource Information

URL	/zrap/sys
HTTP Methods	**POST**
Response Formats	**None**
HTTP Response Status Code	**302**
API Version	**V1.0**

3.3.3 POST Body

Method Name	Method Typ	Description	Free to publish
cmd	**reboot**	Reboots the device. This is necessary after network changes.	✓
cmd	**factory-default**	Go back into Access Point mode with factory-default SSID and password. **This will overwrite all configurations with default settings!**	✓
cmd	**network-default**	Go back in Access Point mode with factory-default SSID and password. All configurations are maintained.	✓

3.3.4 POST Example

Reboot the zApp device.	
Resource URL	http://<IP Address or DNS Name>/zrap/sys
Body Data	cmd=reboot

3.4 Channel Scan -> zrap/chscan

3.4.1 Description

This web service returns the actual state of each channel. The return value is between hundred and zero. We support at the moment only the state for the light. If the value is 100 the light is on and if the value is zero - you can guess three times - the light is off. If you have a blind actuator connected, the value will be always -1 this means the state is unknown.

3.4.2 Resource Information

URL	/zrap/chscan
HTTP Methods	**GET**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.4.3 GET Response Body

Cluster Name	Cluster Instance	Attribute Name	Attribute Value	Free to publish
chscan	**ch(n)**	**val**	the value can be between **0** and **100** or **-1** if the state is unknown	✓

3.4.4 GET Example

Read all channel states.	
Resource URL	http://<IP Address or DNS Name>/zrap/chscan

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<chscan>`
	`  <ch1>`
	`    <val>0</val>`
	`  </ch1>`
	`  <ch2>`
	`    <val>100</val>`
	`  </ch2>`
	`</chscan>`

Read one channel states.	
Resource URL	http://<IP Address or DNS Name>/zrap/chscan/ch2

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<ch2>`
	`  <val>100</val>`
	`</ch2>`

3.5 Channel Notification -> zrap/chnotify

3.5.1 Description

This web service is very special because it will not return immediately! It gives a device (a HTTP server) a chance to notify the requesting client as soon as something has happened in the device. The device will keep this request open until one of the channels has changed its state. Then the device will send the response. The response looks similar to that of a /zrap/chscan service. To be nice to naive clients each /zrap/chnotify request will also get a response after 30 seconds, even if no channel has changed its state! After each response the client must restart a new /zrap/chnotify to keep track of every state change.

3.5.2 Resource Information

URL	/zrap/chnotify
HTTP Methods	**GET**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.5.3 GET Response Body

Cluster Name	Cluster Instance	Attribute Name	Attribute Value	Free to publish
chnotify	**ch(n)**	**val**	the value can be between **0** and **100** or **-1** if the state is unknown	✓

3.5.4 GET Example

Channel state change notification.	
Resource URL	http://<IP Address or DNS Name>/zrap/chnotify

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<chnotify>`
	`  <ch1>`
	`    <val>0</val>`
	`  </ch1>`
	`</hnotify>`

3.6 Channel Control -> zrap/chctrl/ch(n)

3.6.1 Description

Switch on and off the light! This is possible with this service.

3.6.2 Resource Information

URL	/zrap/chctrl
HTTP Methods	**POST**
Response Formats	**None**
HTTP Response Status Code	**302**
API Version	**V1.0**

3.6.3 POST Body

Method Name	Method Typ	Description	Free to publish
cmd	**stop**	stop dimming lights or moving blinds	✓
cmd	**on**	switch light on (100%)	✓
cmd	**off**	switch light off (0%)	✓
cmd	**toggle**	toggle light command on to off or off to on	✓
cmd	**dim_up**	dim light up (use stop to hold)	✓
cmd	**dim_down**	dim light down(use stop to hold)	✓
cmd	**close**	close shutters	✓
cmd	**open**	open shutters	✓
cmd	**move_close**	move shutters to close (use stop to hold)	✓
cmd	**move_open**	move shutters to open (use stop to hold)	✓
cmd	**recall_s(n)**	recall scene (n 1-4)	✓
cmd	**store_s(n)**	store scene (n 1-4)	✓
cmd	**delete_s(n)**	delete scene (n 1-4)	✓
cmd	**dim_up_(t)**	dim light up and stop after time (t) (t = 100–32000 milliseconds)	✓
cmd	**dim_down_(t)**	dim light down and stop after time (t) (t = 100–32000 milliseconds)	✓
cmd	**move_open_(t)**	move blind and stop after time (t) (t = 100–32000 milliseconds)	✓
cmd	**move_close_(t)**	move blind and stop after time (t) (t = 100–32000 milliseconds)	✓
cmd	**dim_(t)**	dim light and stop after time (t) (t = 100–32000 milliseconds)	✓

3.6.4 POST Example

Switch channel one ‘on’ and channel three ‘off’.	
Resource URL	http://<IP Address or DNS Name>/zrap/chctrl/ch1
Body Data	cmd=on

3.6.5 Multicast POST Example

To speed up the channel control you can send also multicast POST methods. For that simply use the URL http://<IP Address or DNS Name>/zrap/chctrl and write the channel numbers behind the method-name “cmd” like this:

Switch channel 1 ‘on’ and channel 2, 3 and 4 ‘off’.	
Resource URL	http://<IP Address or DNS Name>/zrap/chctrl
Body Data	cmd1=on&cmd2=off&cmd3=off&cmd4=off

3.7 Channel Descriptor -> zrap/chdes

3.7.1 Description

With this service you can set a descriptive name for each channel. This resource is just for client Apps to store some user interface information in the device - use it for whatever you want! The only limitation is the length of the strings.

3.7.2 Resource Information

URL	/zrap/chdes
HTTP Methods	**GET/POST**
Response Formats	**XML**
HTTP Response Status Code	**200/302**
API Version	**V1.0**

3.7.3 GET Response Body

Cluster Name	Cluster Instance	Attribute Name	Attribute Value	Free to publish
chdes	ch(n)	name	Channel name string with maximum 32 bytes (1)	✓
chdes	ch(n)	group	Channel group string with maximum 32 bytes (1)	✓
chdes	ch(n)	icon	Channel icon string with maximum 24 bytes (1)	✓
chdes	ch(n)	type	Channel type string with maximum 4 bytes (1)	✓
chdes	ch(n)	cat	Channel category string with maximum 4 bytes (1)	✓

(1) Characters should be UTF8 encoded. Be aware that a UTF8 character can be bigger than 1 byte!

3.7.4 GET Example

Read all channel descriptors	
Resource URL	http://<IP Address or DNS Name>/zrap/chdes
Response Body	

```
<?xml version="1.0" encoding="UTF-8"?>
<chdes>
  <ch1>
    <name>Ceiling Lamp</name>
    <group>Living Room</group>
    <icon> picture34.png</icon>
    <type>3452</type>
    <cat>3209</cat>
  </ch1>
  <ch2>
    <name>Spots</name>
    <group>Kitchen</group>
    <icon>picture01.jpg</icon>
    <type>1234</type>
    <cat>34</cat>
  </ch2>
<chdes>
```

Read one channel descriptors	
Resource URL	http://<IP Address or DNS Name>/zrap/chdes/ch2

Response Body	`<?xml version="1.0" encoding="UTF-8"?>`
	`<chdes>`
	`  <ch2>`
	`    <name>Spots</name>`
	`    <group>Kitchen</group>`
	`    <icon>picture01.jpg</icon>`
	`    <type>1234</type>`
	`    <cat>34</cat>`
	`  </ch2>`
	`<chdes>`

Read one channel descriptors value	
Resource URL	http://<IP Address or DNS Name>/zrap/chdes/ch2/name

Response Body	`<?xml version="1.0" encoding="UTF-8"?>`
	`<name>Spots</name>`

3.7.5 POST Body

Attribute Name	Attribute Arguments	Description	Free to publish
name	**32 byte str**	freely definable UTF8 string for the client application (1)	✓
group	**32 byte str**	freely definable UTF8 string for the client application (1)	✓
icon	**24 byte str**	freely definable UTF8 string for the client application (1)	✓
type	**4 byte str**	freely definable UTF8 string for the client application (1)	✓
cat	**4 byte str**	freely definable UTF8 string for the client application (1)	✓

(1) Characters should be UTF8 encoded. Be aware that a UTF8 character can be bigger than 1 Byte!

3.7.6 POST Example

Set some arguments for channel 1	
Resource URL	http://<IP Address or DNS Name>/zrap/chdes/ch1
Body Data	name=Kitchen&icon=pan.jpg&type=0815

3.8 Network Scan -> zrap/netscan

3.8.1 Description

This service is helpful if you want to integrate a zApp device into a network. It will return a list of available networks including their RSSI. If an RSSI is lower than -75dBm you should inform the user that the router is too far away and a connection would be unreliable.

3.8.2 Resource Information

URL	/zrap/netscan
HTTP Methods	**GET**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.8.3 GET Response Body

Cluster Name	Cluster Instance	Attribute Name	Attribute Value	Free to publish
chdes	**net**	**ssid**	Network name (Service Set Identifier)	✓
chdes	**net**	**ch**	Network channel	✓
chdes	**net**	**enc**	Network encryption (OPEN, WEP, WPA, WPA2)	✓
chdes	**net**	**rssi**	Received Signal Strength Indication	✓
chdes	**net**	**bssid**	MAC address of Access Point	✓

3.8.4 GET Example

Scan all Networks	
Resource URL	http://<IP Address or DNS Name>/zrap/netscan
Response Body	

```
<?xml version="1.0" encoding="US-ASCII"?>
<netscan>
  <net>
    <ssid>guest</ssid>
    <ch>01</ch>
    <enc>NONE</enc>
    <rssi>-85</rssi>
  </net>
  <net>
    <ssid>hge-96486</ssid>
    <ch>06</ch>
    <enc>WPA2-PERSONAL</enc>
    <rssi>-54</rssi>
  </net>
  <net>
    <ssid>home-sweet-home</ssid>
    <ch>12</ch>
    <enc>NONE</enc>
    <rssi-33</rssi>
  </net>
 </netscan>
```

3.8.5 GET Example

If you have more than one Access Point with the same SSID and you need more information to distinguish each Access Point you can add the network name as query string. So you get back a list of all networks with that name including their MAC-addresses.

Scan all Networks	
Resource URL	http://<IP Address or DNS Name>/zrap/netscan?ssid= home-sweet-home
Response Body	

```
<?xml version="1.0" encoding="US-ASCII"?>
<netscan>
  <net>
    <ssid>home-sweet-home</ssid>
    <bssid>5c:50:15:37:ed:99</bssid>
    <ch>01</ch>
    <enc>NONE</enc>
    <rssi>-85</rssi>
  </net>
  <net>
    <ssid>home-sweet-home</ssid>
    <bssid>34:62:88:f2:f6:29</bssid>
    <ch>06</ch>
    <enc>WPA2-PERSONAL</enc>
    <rssi>-54</rssi>
  </net>
 </netscan>
```

3.9 Network Configuration -> zrap/net

3.9.1 Description

Read or write the network settings. Use this service to integrate the zApp device into a network.

3.9.2 Resource Information

URL	/zrap/net
HTTP Methods	**GET/POST**
Response Formats	**XML**
HTTP Response Status Code	**200/302**
API Version	**V1.0**

3.9.3 GET Response Body

Cluster Name	Attribute Name	Attribute Value	Free to publish
net	**ssid**	Network name (Service Set Identifier)	✓
net	**pw**	Password (only readable in AccessPointMode)	✓
net	**mac**	MAC address	✓
net	**mode**	Network Mode ('0'(AccessPoint), '1'(Associate))	✓
net	**enc**	Network encryption types (OPEN, WEP, WPA, WPA2) You can also use numbers to select the encryption. ('1'(OPEN), '2'(WEP), '4'(WPA), '8'(WPA2))	✓
net	**ip**	IP address	✓
net	**mask**	Network mask	✓
net	**gw**	Gateway address	✓
net	**bssid**	MAC address of Access Point	✓

3.9.4 GET Example

Read all Network Settings	
Resource URL	http://<IP Address or DNS Name>/zrap/net
Response Body	

```
<?xml version="1.0" encoding="US-ASCII"?>
 <net>
  <ssid>zApp-12345555</ssid>
  <pw>password</pw>
  <mac>20:f8:5e:a1:ba:fe</mac>
  <mode>0</mode>
  <enc>WPA2</enc>
  <ip>192.168.0.1</ip>
  <mask>255.255.255.0</mask>
  <gw>192.168.0.1</gw>
 </net>
```

Read one Network Attribute	
Resource URL	http://<IP Address or DNS Name>/zrap/net/mac
Response Body	

```
<?xml version="1.0" encoding="US-ASCII"?>
<mac>20:f8:5e:a1:ba:fe</mac>
```

3.9.5 POST Body

Attribute Name	Attribute Arguments	Description	Free to publish

ssid	**1-32 byte string**	Network name (Service Set Identifier)	✓
bssid	**mac address**	Use this attribute instead of SSID to bind a zeptrion WLAN device strictly to one Access Point. e.g. 20:f8:5e:a1:ba:fe ⚠ The SSID is still required for backward compatibility. Always set the SSID before you set the BSSID!	✓
pw	**8-55 byte string**	Password The following characters are allowed: A-Z a-z 0-9 Space ! # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~ "	✓
enc	**'OPEN' 'WEP' 'WPA' 'WPA2'**	Network encryption (OPEN, WEP, WPA, WPA2) If you set the encryption as number ('1'(OPEN), '2'(WEP), '4'(WPA), '8'(WPA2)) you get also a response with a number. This will be no longer supported in the next version!	✓

3.9.6 POST Example

Change Network setting	
Resource URL	http://<IP Address or DNS Name>/zrap/net
Body Data	ssid=HomeNet&pw=q9g34T34xdsdsd&enc=WPA2

To apply new network settings a reboot is necessary. Use this service to reboot the device:	
Resource URL	http://<IP Address or DNS Name>/zrap/sys
Body Data	cmd=reboot

3.10 Received Signal Strength Indication -> zrap/rssi

3.10.1 Description

This service returns the current RSSI (Received Signal Strength Indication) of a device. Checking the RSSI may help to fix connection problems! If the RSSI is below about -75 dBm then the connection may become unreliable and whenever it drops for too long the device will reboot to find a better connection.

3.10.2 Resource Information

URL	/zrap/rssi
HTTP Methods	**GET**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.10.3 GET Response Body

Attribute Name	Attribute Value	Free to publish
dbm	Received Signal Strength Indication (in dBm)	✓

3.10.4 GET Example

Get rssi	
Resource URL	http://<IP Address or DNS Name>/zrap/rssi

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<rssi>`
	`    <dbm>-35</dbm>`
	`</rssi>`

3.11 Device Identification -> zrap/id

3.11.1 Description

With this service you get all device identification attributes.
The version number ("hw", "sw" and "boot") have the format "<major>.<minor>.<bugfix><other_text>" where major, minor and bugfix all are 2-digit-numbers and higher means newer.

3.11.2 Resource Information

URL	/zrap/id
HTTP Methods	**GET**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.11.3 GET Response Body

Cluster Name	Attribute Name	Attribute Value	Free to publish
id	**hw**	Hardware Version	✓
id	**sn**	Unique serial number	✓
id	**sys**	System Name	✓
id	**type**	Device Type	✓
id	**oen**	owner environment name	✓
id	**sw**	Software Version	✓
id	**boot**	Bootloader Version	✓

3.11.4 GET Example

Read all id Settings	
Resource URL	http://<IP Address or DNS Name>/zrap/id

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<id>`
	`   <hw>01.04.00</hw>`
	`   <sn>12345555</sn>`
	`   <sys>ZEPTRION</sys>`
	`   <type>3340-2-A</type>`
	`   <oen>zApp</oen>`
	`   <sw>01.06.00</sw>`
	`   <boot>01.03.03 (Jun 30 2014 17:03:35)</boot>`
	`</id>`

Read one id attribute	
Resource URL	http://<IP Address or DNS Name>/zrap/id/type

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<type>3340-2-A</type>`

3.12 Location -> zrap/loc

3.12.1 Description

The location descriptor is like the channel descriptor, it is just used by a client app. You can store a useful string for example the site name.

3.12.2 Resource Information

URL	/zrap/loc
HTTP Methods	**GET/POST**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.12.3 GET Response Body

Attribute Name	Attribute Value	Free to publish
name	Location Descriptor Name	✓

3.12.4 GET Example

Get rssi	
Resource URL	http://<IP Address or DNS Name>/zrap/loc

Response Body	`<?xml version="1.0" encoding="UTF-8"?>`
	`<loc>`
	`  <name>Holiday Home</name>`
	`</loc>`

3.12.5 POST Body

Method Name	Method Arguments	Description	Free to publish
name	**32 byte String**	freely definable string by the client application (1)	✓

(1) Characters should be UTF8 encoded. Be aware that a UTF8 character can be bigger than 1 byte!

3.12.6 POST Example

Set location attribute	
Resource URL	http://<IP Address or DNS Name>/zrap/loc
Body Data	name='Holiday Home'

3.13 Date Time -> zrap/date

3.13.1 Description

Service for date time settings.

3.13.2 Resource Information

URL	/zrap/date
HTTP Methods	**GET/POST**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.13.3 GET Response Body

Cluster Name	Attribute Name	Attribute Value	Free to publish
date	**rfc1123**	RFC 1123 Date Time Stamp-String	✓
date	**tz**	Difference in HHMM between GMT and local time (time zone) Value in seconds	✓
date	**dst**	Offset for Daylight Saving Time Value in HHMM	✓

3.13.4 GET Example

Get rssi	
Resource URL	http://<IP Address or DNS Name>/zrap/date
Response Body	

```
<?xml version="1.0" encoding="US-ASCII"?>
<date>
  <rfc1123>Thu, 01 Jan 1970 02:17:57 +0300</rfc1123>
  <tz>+0200</tz>
  <dst>+0100</dst>
</date>
```

3.13.5 POST Body

Method Name	Method Arguments	Description	Free to publish
rfc1123	**rfc1123-string**	RFC 1123 Date Time Stamp-String Note: must be a GMT Time Stamp	✓
tz	**HHMM**	Difference in hours and minutes between GMT and local time. e.g. -0100, +0200, 0100	✓
dst	**HHMM**	Daylight Saving Time offset in hours and minutes e.g. +0100, 0100, 0000	✓

3.13.6 POST Example

Change Network setting	
Resource URL	http://<IP Address or DNS Name>/zrap/date
Body Data	rfc1123= Thu, 01 Jan 1970 02:17:57 GTM&tz=0000&dst=0100

3.14 Scheduler -> zrap/scheduler

3.14.1 Description

Let's run a scheduler on a zApp device.

3.14.2 Resource Information

URL	/zrap/scheduler
HTTP Methods	**GET/POST**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.14.3 GET Response Body

Cluster Name	Cluster Instance	Attribute Name	Attribute Value	Free to publish
scheduler	**job(n)**	**tm**	Scheduler time HHMM	✓

scheduler	**job(n)**	**day**	Day bit-field in HEX from Monday to Sunday
			`e.g. ('7F' -> all days active)`
			`('1F' -> Monday. – Friday  active)`
			`('01' -> Monday active)`
				✓

scheduler	**job(n)**	**act**	Action bytes for each channel
			`+---+---+---+---+`
			`|ch1|ch2|ch3|ch4|`
			`+---+---+---+---+`
			Supported values are '0' – '8'
			`0 ->'none'`
			`1 ->'off'`
			`2 ->'on'`
			`3 -> 'open'`
			`4 -> 'close'`
			`5 -> 'recall_s1'`
			`6 -> 'recall_s2'`
			`7 -> 'recall_s3'`
			`8 -> 'recall_s4'`
				✓

scheduler	**job(n)**	**on**	Switch on or off a "normal" scheduler job
			`1 -> 'on = true'  (hex value 0x0|0x1)`
			`0 -> 'on = false' (hex value 0x0|0x0)`
			Switch on or off a presence simulation scheduler job
			`3 -> 'on = true'  (hex value 0x2|0x1)`
			`2 -> 'on = false' (hex value 0x2|0x0)`
			
			Switch on or off a astro scheduler job
			`9 -> 'on = true'  (hex value 0x8|0x1)`
			`8 -> 'on = false' (hex value 0x8|0x0)`
				✓

scheduler	**job(n)**	**id**	Two-character identifier for a client application	✓

3.14.4 GET Example

Get rssi	
Resource URL	http://<IP Address or DNS Name>/zrap/scheduler

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<scheduler>`
	`  <job1>`
	`    <tm>1830</tm>`
	`    <day>7F</day>`
	`    <act>0102</act>`
	`    <on>1</on>`
	`    <id>85</id>`
	`  </job1>`
	`  <jobn>`
	`  …`
	`  <jobn>`
	`</scheduler>`

3.14.5 POST Body

Method Name	Method Arguments	Description	Free to publish
tm	**hhmm**	Scheduler time hhmm (hh -> 00 - 23) (mm-> 00 - 59)	✓

day	**00**	Bit field for each day in Hex from Monday to Sunday (1->day active) (0->day inactive)
		`e.g. ('7F' -> all days active)`
		`     ('1F' -> Monday. - Friday  active)`
		`     ('01' -> Monday active)`
			✓

act	**xxxx**	Action bytes for each channel
		`+---+---+---+---+`
		`|ch1|ch2|ch3|ch4|`
		`+---+---+---+---+`
		Supported values are '0' – '8'
		`0 ->'none'`
		`1 ->'off'`
		`2 ->'on'`
		`3 -> 'open'`
		`4 -> 'close'`
		`5 -> 'recall_s1'`
		`6 -> 'recall_s2'`
		`7 -> 'recall_s3'`
		`8 -> 'recall_s4'`
			✓

on	**x**	Switch on or off a "normal" scheduler job
		`1 -> 'on = true' (hex value 0x0|0x1)`
		`0 -> 'on = false'(hex value 0x0|0x0)`
		Switch on or off a presence simulation scheduler job
		`3 -> 'on = true' (hex value 0x2|0x1)`
		`2 -> 'on = false'(hex value 0x2|0x0)`
		
		Switch on or off a astro scheduler job
		`9 -> 'on = true' (hex value 0x8|0x1)`
		`8 -> 'on = false'(hex value 0x8|0x0)`
			✓

id	**xx**	Two-character identifier for a client application	✓

3.14.6 POST Example

Setup a Schedule for Saturday and Sunday at 18:35 for channel 1 and 2 with cmd 'on'	
Resource URL	http://<IP Address or DNS Name>/zrap/scheduler/job4
Body Data	tm=1835&day=60&cmd=on&ch=03

3.14.7 Multicast POST Body

To send more than one scheduler-job configuration you can use the multicast POST method. For that simply use the URL http://<IP Address or DNS Name>/zrap/scheduler and write the job number behind the method-name.

3.14.8 Multicast POST Example

Set for job 15 the command 'on', the time '08:00', the channel '1', the day Friday ('10') and the mode to active ('1').	
Resource URL	http://<IP Address or DNS Name>/zrap/scheduler
Body Data	cmd15=on&tm15=0800&ch15=1&day15=10&on15=1

3.15 Network Time Protocol -> zrap/ntp

3.15.1 Description

This service is used to synchronize the zApp device system-time with an NTP server.

3.15.2 Resource Information

URL	/zrap/ntp
HTTP Methods	**GET/POST**
Response Formats	**XML**
HTTP Response Status Code	**200**
API Version	**V1.0**

3.15.3 GET Response Body

Cluster Name	Attribute Name	Attribute Value	Free to publish
ntp	**url**	NTP server URL with maximum 32 characters	✓
ntp	**per**	NTP server polling period in hours.	✓

3.15.4 GET Example

Get rssi	
Resource URL	http://<IP Address or DNS Name>/zrap/ntp

Response Body	`<?xml version="1.0" encoding="US-ASCII"?>`
	`<ntp>`
	`  <url>ntp.metas.ch</url>`
	`  <per>12</per>`
	`</ntp>`

3.15.5 POST Body

Method Name	Method Arguments	Description	Free to publish
url	**url-string**	NTP-server URL or IP address (max. length 32 characters)	✓
per	**hours**	NTP server polling period in hours. If pre=0 the NTP service is disabled.	✓

3.15.6 POST Example

Change Network setting	
Resource URL	http://<IP Address or DNS Name>/zrap/ntp
Body Data	ip=ntp.metas.ch&per=12

4 zeptrion WLAN Device Discovery

A device discovery can locate all zeptrion WLAN devices in the same network. Each zeptrion WLAN device uses mDNS to register its unique host name. This host name is identical to the SSID of the device, except that SSIDs are case-sensitive and host names are not.

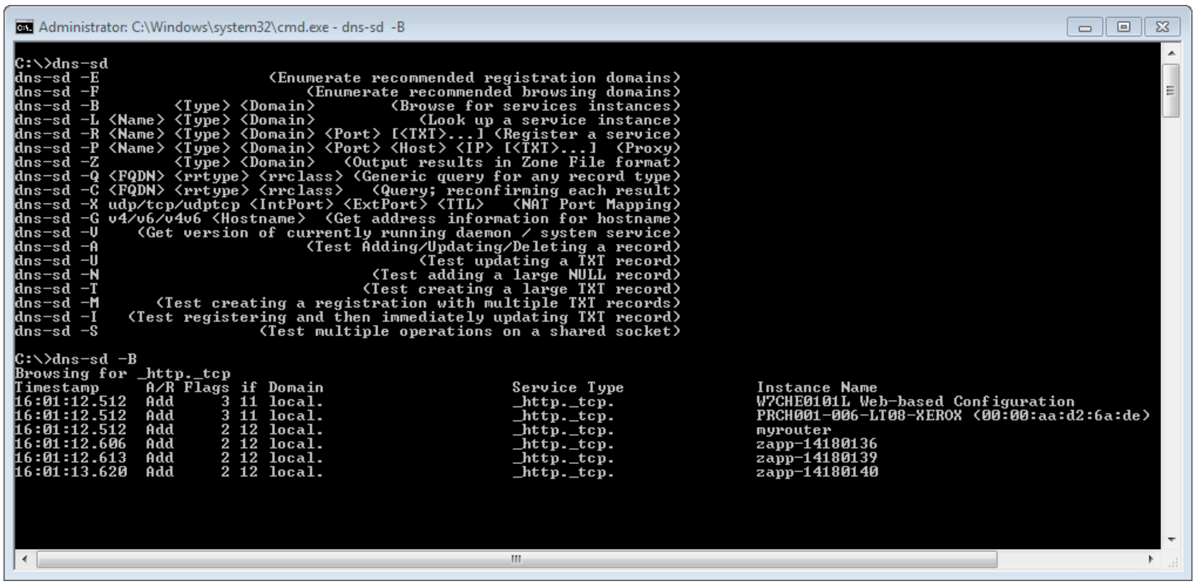
4.1 mDNS Host Name Format

```
Host-name head
  |
  |
  |      Host name number [YYWWNNNN]
  |      |    YY-> production year
  |      |    WW->production week
  |      |    NNNN -> continuous number per week
  |      |
  |      |
  |      |      Domain name
  |      |      |
  |      |      |

 __   ______   ___
/  \ /      \ /   \
zapp-yywwnnnn.local
```

4.2 mDNS Console Tools

To discover all zeptrion WLAN devices on your network you need an mDNS client like the command “dns-sd”. On Microsoft Windows you have to install Apple’s Bonjour Print Services for Windows first.

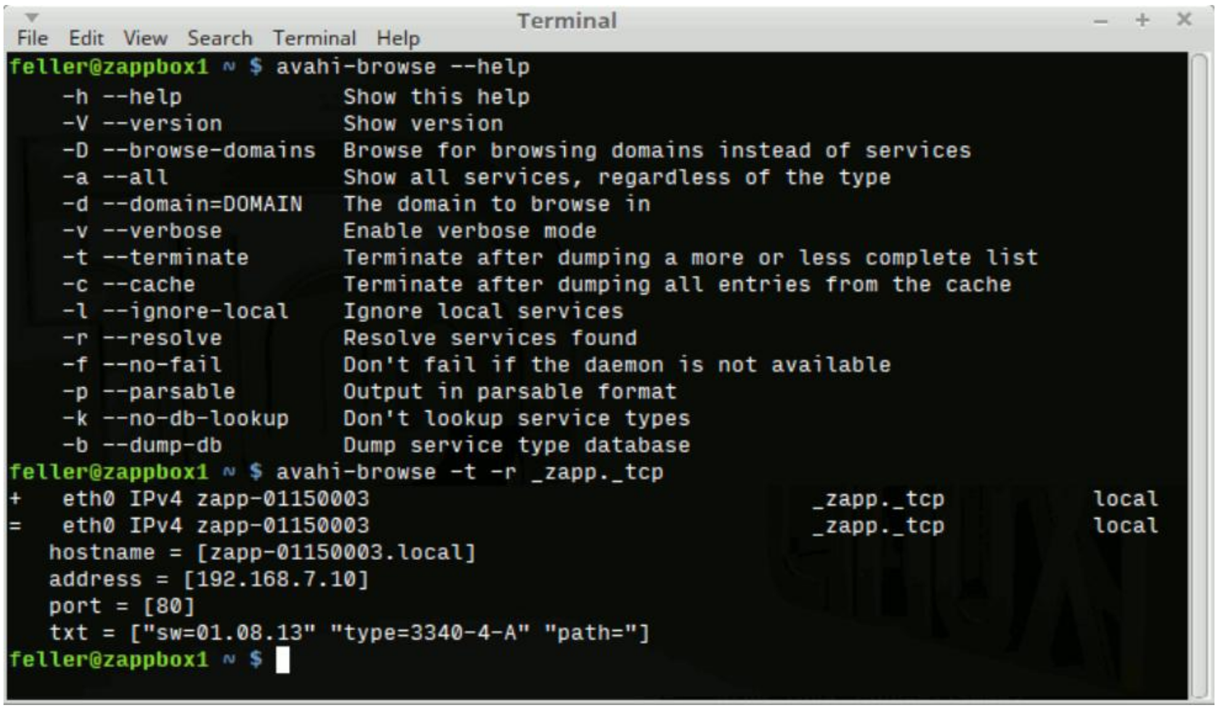


Administrator: C:\Windows\system32\cmd.exe - dns-sd -B

```
C:\>dns-sd
dns-sd -E                          <Enumerate recommended registration domains>
dns-sd -F                              <Enumerate recommended browsing domains>
dns-sd -B        <Type> <Domain>                 <Browse for services instances>
dns-sd -L <Name> <Type> <Domain>                    <Look up a service instance>
dns-sd -R <Name> <Type> <Domain> <Port> [<TXT>...] <Register a service>
dns-sd -P <Name> <Type> <Domain> <Port> <Host> <IP> [<TXT>...]  <Proxy>
dns-sd -Z        <Type> <Domain>   <Output results in Zone File format>
dns-sd -Q <FQDN> <rrtype> <rrclass> <Generic query for any record type>
dns-sd -C <FQDN> <rrtype> <rrclass>   <Query; reconfirming each result>
dns-sd -X udp/tcp/udptcp <IntPort> <ExtPort> <TTL>   <NAT Port Mapping>
dns-sd -G v4/v6/v4v6 <Hostname>  <Get address information for hostname>
dns-sd -V    <Get version of currently running daemon / system service>
dns-sd -A                          <Test Adding/Updating/Deleting a record>
dns-sd -U                                      <Test updating a TXT record>
dns-sd -N                                 <Test adding a large NULL record>
dns-sd -T                                <Test creating a large TXT record>
dns-sd -M      <Test creating a registration with multiple TXT records>
dns-sd -I    <Test registering and then immediately updating TXT record>
dns-sd -S                     <Test multiple operations on a shared socket>

C:\>dns-sd -B
Browsing for _http._tcp
Timestamp     A/R Flags if Domain                    Service Type              Instance Name
16:01:12.512  Add     3 11 local.                    _http._tcp.               W7CHE0101L Web-based Configuration
16:01:12.512  Add     3 11 local.                    _http._tcp.               PRCH001-006-LT08-XEROX (00:00:aa:d2:6a:de)
16:01:12.512  Add     2 12 local.                    _http._tcp.               myrouter
16:01:12.606  Add     2 12 local.                    _http._tcp.               zapp-14180136
16:01:12.613  Add     2 12 local.                    _http._tcp.               zapp-14180139
16:01:13.620  Add     2 12 local.                    _http._tcp.               zapp-14180140
```

On a Linux like Debian or Ubuntu you can install avahi-dnssd with “sudo apt-get install libavahi-compat-libdnssd1” and then use the commands avahi-*.



```
feller@zappbox1 ~ $ avahi-browse --help
    -h --help            Show this help
    -V --version         Show version
    -D --browse-domains  Browse for browsing domains instead of services
    -a --all             Show all services, regardless of the type
    -d --domain=DOMAIN   The domain to browse in
    -v --verbose         Enable verbose mode
    -t --terminate       Terminate after dumping a more or less complete list
    -c --cache           Terminate after dumping all entries from the cache
    -l --ignore-local    Ignore local services
    -r --resolve         Resolve services found
    -f --no-fail         Don't fail if the daemon is not available
    -p --parsable        Output in parsable format
    -k --no-db-lookup    Don't lookup service types
    -b --dump-db         Dump service type database
feller@zappbox1 ~ $ avahi-browse -t -r _zapp._tcp
+   eth0 IPv4 zapp-01150003                                 _zapp._tcp           local
=   eth0 IPv4 zapp-01150003                                 _zapp._tcp           local
   hostname = [zapp-01150003.local]
   address = [192.168.7.10]
   port = [80]
   txt = ["sw=01.08.13" "type=3340-4-A" "path="]
feller@zappbox1 ~ $ 
```

4.3 mDNS Python Script Examples

“pybonjour” provides a Python interface to Apple Bonjour and other compatible DNS-SD libraries such as Avahi. It allows Python scripts to take advantage of Zero Configuration Networking to register, discover and resolve services on both local and wide-area networks.

```
#! /usr/bin/env python

from pybonjour import DNSServiceBrowse, DNSServiceResolve, DNSServiceProcessResult

def resolve_callback(sdRef, flags, interfaceIndex, errorCode,
                     fullname, hosturl, port, txtRecord):
    services.append({
        'hosturl': str(hosturl),
        'port': str(port),
        'fullname': str(fullname),
    })

def browse_callback(sdRef, flags, interfaceIndex, errorCode,
                    serviceName, regtype, replyDomain):
    resolve_sdRef = DNSServiceResolve(
        0, interfaceIndex, serviceName, regtype, replyDomain, resolve_callback )
    DNSServiceProcessResult(resolve_sdRef)
    resolve_sdRef.close()


services = []
browse_sdRef = DNSServiceBrowse(regtype = '_http._tcp', callBack = browse_callback)
DNSServiceProcessResult(browse_sdRef)
browse_sdRef.close()

print('\n%d service(s) found:' % len(services))
for service in services:
    print('Host "%(hosturl)s:%(port)s" has service "%(fullname)s"' % (service))
```

Alternatively there is a pure-Python implementation named “zeroconf” which does not need Apple Bonjour! It is hosted on https://pypi.python.org/pypi/zeroconf and can be installed like other Python packages by executing “pip install zeroconf”.

```
#! /usr/bin/env python

from zeroconf import ServiceBrowser, Zeroconf
from time import sleep

class MyListener(object):

    def remove_service(self, zeroconf, type, name):
        print("Service %s removed" % (name,))

    def add_service(self, zeroconf, type, name):
        info = zeroconf.get_service_info(type, name)
        print("Service %s added, service info: %s" % (name, info))


zeroconf = Zeroconf()
listener = MyListener()
browser = ServiceBrowser(zeroconf, "_zapp._tcp.local.", listener)
sleep(5.0)
zeroconf.close()
```

Important note: zeptrion devices with software before 01.08.xx will register themself only as service type “_http._tcp”. Since 01.08.00 they will register as service type **“_zapp._tcp”** and for backward-compatibility also as “_http._tcp”. The old service type “_http._tcp” will include other devices like printers and should not be used anymore.

5 zeptrionAir Webservices

For the Smartfront functionalities additional Webservices have been added. All these new services are in the `JSON` format and are accessible under the path `zapi`.
In the near future all existing `zrap` services will be also supported under `zapi` and in the `JSON` format.

5.1 Smartfront

As soon as you have connected a Smartfront on the WLAN-Zwischenmodul-2k 3340-2-B, the following services are available in the `zapi/smartfront` path.

```
http://<IP Address or DNS Name>/zapi/smartfront/id
http://<IP Address or DNS Name>/zapi/smartfront/sensors
http://<IP Address or DNS Name>/zapi/smartfront/led
```

On the webpage `http://<IP address or hostname>/smf.html` of each device you will find an example that uses some of these services.

5.1.1 Get Smartfront Identification

URL	/zapi/smartfront/id
HTTP Methods	**GET**
API Version	**V1.0**
Free to publish	✓

5.1.1.1 Description

Gets the identification and functionality attributes of the connected Smartfront.

5.1.1.2 Response

Name	Type	Description
btfu	string	button function descriptor
sw	string	software version
hw	string	hardware version
sys	string	supported system name
sid	string	short identification
type	string	Feller product-type

5.1.1.3 Response example

```
GET /zapi/id
{
    "btfu":"0,0,1000,1000,0,0,8501,8602",
    "sw"  :"01.01.02",
    "hw"  :"01.00.00",
    "sys" :"ZEPTRION",
    "sid" :"1",
    "type":"920-3306.24.ZS"
}
```

5.1.2 Get Smartfront Sensor Values

URL	/zapi/smartfront/sensor
HTTP Methods	**GET**
API Version	**V1.0**
Free to publish	✓

5.1.2.1 Description

Gets a list of all sensor values.

Note:

- Not all sensors are available on each Smartfront.
- The value of the temperature can be influenced by the installation situation.

5.1.2.2 Response

Name	Type	Description
temp	string	temperature in Celsius
lux	string	brightness in Lux
hum	string	humidity in percents

5.1.2.3 Response example

```
GET /zapi/smartfront/sensor

{
    "temp" :"24.50C",
    "lux"  :"none",
    "hum"  :"none",
}
```

5.1.3 Get and set Smartfront LEDs

URL	/zapi/smartfront/id
HTTP Methods	**GET / POST**
API Version	**V1.0**
Free to publish	✓

5.1.3.1 Description

Get or set the attributes and states of each Smartfront LED.
Note:

- Most attributes are used by the zeptrionAir application itself for system status indications.
- Only the background color is never overwritten.

5.1.3.2 Response

Name	Type	Description

id	uint8	Identify number of each LED
		`+------------------+`
		`|1               2|`
		`+------------------+`
		`|3               4|`
		`+------------------+`
		`|5               6|`
		`+------------------+`
		`|7               8|`
		`+------------------+`

on	bool	On/Off state of the light. On=true, Off=false Do not overwrite this attribute in a zeptrionAIR installation. It's used from the zeptrionAir application!
rgb	string	Color as RGB string Do not overwrite this attribute in a zeptrionAIR installation. It's used from the zeptrionAir application!
effects	string	effect string Do not overwrite this attribute in a zeptrionAIR installation. It's used from the zeptrionAir application!
bg	string	background color as RGB string individual color RGB attribute

5.1.3.3 Get example

Get an attribute list of all eight LEDs.

```
GET /zapi/smartfront/led
```

```
[
    {
        "id"      : 1,
        "on"      : true,
        "rgb"     :"#004020",
        "effects" :"#ffffff",
        "bg"      :"#000000",
    },
    {
        "id"      : 2,
        "on"      : false,
        "rgb"     :"#000000",
        "effects" :"#000000",
        "bg"      :"#550000",
    },
    ...
]
```

5.1.3.4 POST example

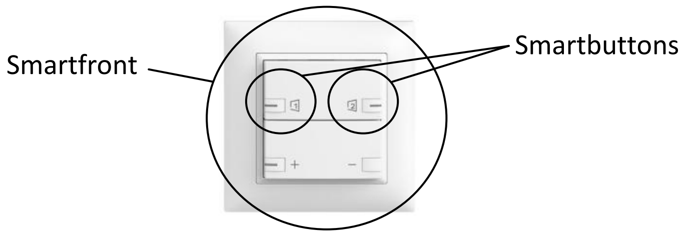
Set LED 2 to red and LED 4 to green.
You can set several LEDs with a JSON-array in one service.

```
POST /zapi/smartfront/led
```

```
[{"id":2,"bg":"#220000",},{"id":4,"bg":"#002200",}]
```

5.2 Smartfront Configuration

The main function of the Smartfront is to use it as a Webservice trigger. In this chapter we explain how you can configure your custom services on a Smartfront.
To use this functionality you need a WLAN-Zwischenmodul `3340-2-B` and a Front `920-330x.xx.ZS` or `920-330x.xx.ZU`.



To configure a custom HTTP-request trigger onto a Smartbutton you need the following services.

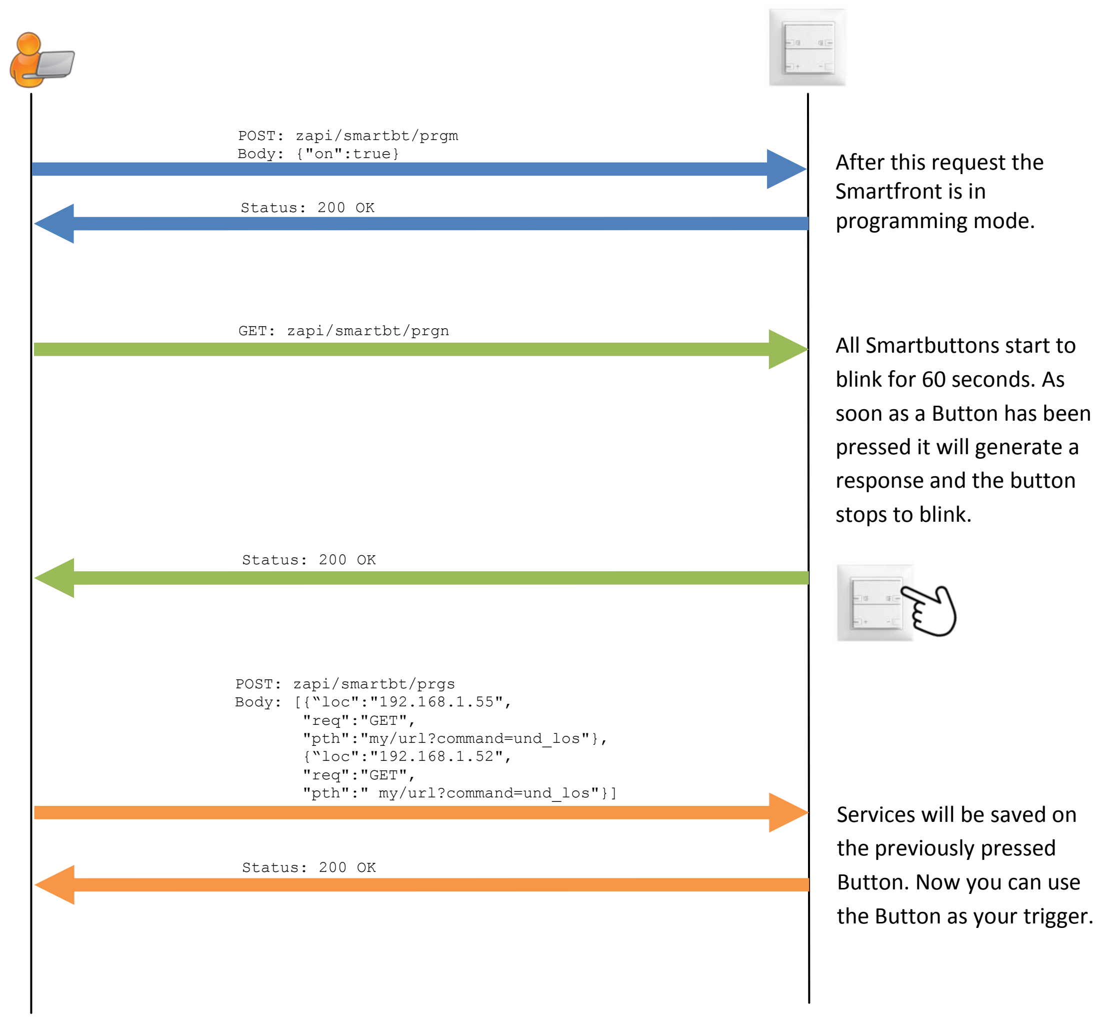
`http://<IP Address or DNS Name>/zapi/smartbt/prgm`
Set the Smartfront into programming mode.

`http://<IP Address or DNS Name>/zapi/smartbt/prgn`
Gets a response as soon as Smartbutton has been pressed.

`http://<IP Address or DNS Name>/zapi/smartbt/prgs`
Stores the HTTP-request on the previously pressed Smartbutton.

5.2.1 Smartfront configuration example

To configure a Smartbutton the following sequence is necessary.


POST: zapi/smartbt/prgm
Body: {"on":true}
Status: 200 OK
After this request the Smartfront is in programming mode.
GET: zapi/smartbt/prgn
All Smartbuttons start to blink for 60 seconds. As soon as a Button has been pressed it will generate a response and the button stops to blink.
Status: 200 OK
POST: zapi/smartbt/prgs
Body: [{"loc":"192.168.1.55",
"req":"GET",
"pth":"my/url?command=und_los"},
{"loc":"192.168.1.52",
"req":"GET",
"pth":" my/url?command=und_los"}]
Services will be saved on the previously pressed Button. Now you can use the Button as your trigger.
Status: 200 OK

5.2.2 Set Smartfront into programming mode

URL	/zapi/smartbt/prgm
HTTP Methods	**GET / POST**
API Version	**V1.0**
Free to publish	✓

5.2.2.1 Description

Get or set the attributes of the Smartfront programming mode.

5.2.2.2 Response

Name	Type	Description
on	bool	On/Off state of the programming mode. On=true, Off=false
ntm	uint16	Notification time in seconds after which `zapi/smartfront/prgn` will be aborted and the Smartbutton stops blinking even if it was not pressed. This attribute is optional (default is 60 seconds)

5.2.3 Get Smartbutton programming notification

URL	/zapi/smartbt/prgn
HTTP Methods	**GET**
API Version	**V1.0**
Free to publish	✓

5.2.3.1 Description

Get a 200 response as soon a smart button is pressed.

5.2.3.2 Response

Name	Type	Description
prg	bool	True: valid button has been pressed False: timeout has passed or invalid button has been pressed

5.2.4 Set Smartbutton Service

URL	/zapi/smartbt/prgs
HTTP Methods	**POST**
API Version	**V1.0**
Free to publish	✓

5.2.4.1 Description

Store a custom Webservice to be trigger by a SmartButton.
If you want to store several HTTP-requests on one Smartbutton you can use a JSON-array. But be careful: a total `prgs` service must not be longer than 730 bytes including your HTTP-header! If you reach this limitation, you can omit each attribute that is set to the same value as in the previous array element (see example below).

If there are double-quotes or back-slashes in your service-values you must escape them with a backslash. For example to set bdy to {"hue":12345} send bdy="{\"hue\":12345}"!

5.2.4.2 Response

Name	Type	Description
req	string	request method eg. `"POST", "GET", "PUT", "DELETE"`
loc	string	Location can be a IP or dns name eg. `"192.168.2.2", "zeptrion.feller.ch"`
pth	string	URL path eg. `"/zrap/chctrl"`

typ	string	content-type HTTP header field
		
		eg. `"application/x-www-form-urlencoded",`
		`"application/json",`
		`"text/xml"`

hdr	string	individual HTTP header field `"\r\n"` is after each header field recommended eg. `"SOAPACTION:http://test/foo#MyMessage\r\n"`
prt	sting	port eg. `"1400"`
bdy	string	body eg. `"cmd1=toggle"`

5.2.4.3 POST examples

Store a zeptrionAir scene on a Smartfront:

```
POST /zapi/smartbt/prgs
[
{"req":"POST",
 "loc":"192.168.1.164",
 "pth":"/zrap/chctrl",
 "bdy":"cmd1=recall_s1"
},
{"req":"POST",
 "loc":"192.168.1.185",
 "pth":"/zrap/chctrl",
 "bdy":"cmd2=on&cmd3=off"
}]
```

Equal but shorter: (without repeating attributes with constant values)

```
POST /zapi/smartbt/prgs
[
{"req":"POST",
 "loc":"192.168.1.164",
 "pth":"/zrap/chctrl",
 "bdy":"cmd1=recall_s1"
},
{"loc":"192.168.1.185",
 "bdy":"cmd2=on&cmd3=off"
}]
```

Store a Philips Hue command on a Smartfront:

```
POST /zapi/smartbt/prgs
{"typ":"application/json",
 "req":"PUT",
 "loc":"192.168.1.101",
 "pth":"/api/ppVQsNcCKHf0V4rtfhgxT4zVvpD1KhIovkk7b6RLLX/lights/1/state",
 "bdy":"{\"on\":true,\"hue\":46920,\"sat\":254}"
}
```

Store a Sonos command on a Smartfront:

```
POST /zapi/smartbt/prgs
{"typ":"text/xml",
 "req":"POST",
 "loc":"192.168.1.190", "prt":"1400",
 "pth":"/MediaRenderer/AVTransport/Control",
 "hdr":"SOAPACTION: urn:schemas-upnp-org:service:AVTransport:1#Play\r\n",
 "bdy":"<s:Envelope xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\"
s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\"><s:Body><u:P
lay xmlns:u=\"urn:schemas-upnp-
org:service:AVTransport:1\"><InstanceID>0</InstanceID><Speed>1</Speed></u:
Play></s:Body></s:Envelope>"
}
```

Example of a party scene including zeptrion, Hue and Sonos services:

```
POST /zapi/smartbt/prgs
[
{"typ":" application/x-www-form-urlencoded",
 "req":"POST",
 "loc":"192.168.1.164",
 "pth":"/zrap/chctrl/ch",
 "bdy":"cmd1=recall_s1"
},
{"typ":" application/x-www-form-urlencoded",
 "req":"POST",
 "loc":"192.168.1.185",
 "pth":"/zrap/chctrl",
 "bdy":"cmd2=on&cmd3=off"
},
{"typ":"application/json",
 "req":"PUT",
 "loc":"192.168.1.101",
 "pth":"/api/ppVQsNcCKHf0V4rtfhgxT4zVvpD1KhIovkk7b6RLLX/lights/1/state",
 "bdy":"{\"on\":true,\"hue\":46920,\"sat\":254}"
},
{"typ":"text/xml",
 "req":"POST",
 "loc":"192.168.1.190",
 "prt":"1400",
 "pth":"/MediaRenderer/AVTransport/Control",
 "hdr":"SOAPACTION: urn:schemas-upnp-org:service:AVTransport:1#Play\r\n",
 "bdy":"<s:Envelope xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\"
s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\"><s:Body><u:P
lay xmlns:u=\"urn:schemas-upnp-
org:service:AVTransport:1\"><InstanceID>0</InstanceID><Speed>1</Speed></u:
Play></s:Body></s:Envelope>"
}
]
```