2N Helios IP HTTP API

Konfigurační manuál

Verze: 2.17

www.2n.cz

Společnost 2N TELEKOMUNIKACE a.s. je českým výrobcem a dodavatelem telekomunikační techniky.

K produktovým řadám, které společnost vyvíjí, patří GSM brány, pobočkové ústředny, dveřní a výtahové komunikátory. 2N TELEKOMUNIKACE a.s. se již několik let řadí mezi 100 nejlepších firem České republiky a již dvě desítky let symbolizuje stabilitu a prosperitu na trhu telekomunikačních technologií. V dnešní době společnost vyváží do více než 120 zemí světa a má exkluzivní distributory na všech kontinentech.

Pro rychlé nalezení informací a zodpovězení dotazů týkajících se 2N produktů a služeb 2N TELEKOMUNIKACE spravuje databázi FAQ nejčastějších dotazů. Na www.faq.2n.cz naleznete informace týkající se nastavení produktů, návody na optimální použití a postupy „Co dělat, když...".

Společnost 2N TELEKOMUNIKACE a.s. tímto prohlašuje, že zařízení 2N® je ve shodě se základními požadavky a dalšími příslušnými ustanoveními směrnice 1999/5/ES. Plné znění prohlášení o shodě naleznete CD-ROM (pokud je přiloženo) nebo na www.2n.cz.

Společnost 2N TELEKOMUNIKACE a.s. je vlastníkem certifikátu ISO 9001:2009. Všechny vývojové, výrobní a distribuční procesy společnosti jsou řízeny v souladu s touto normou a zaručují vysokou kvalitu, technickou úroveň a profesionalitu všech našich výrobků.

Obsah:

1. Úvod

2N® Helios IP HTTP API je aplikační rozhraní pro ovládání vybraných funkcí interkomu pomocí HTTP protokolu. Toto rozhraní umožňuje jednoduše integrovat interkomy 2N Helios IP s produkty třetích stran, např. systémy domácí automatizace, zabezpečovací a monitorovací systémy budov apod.

2N® Helios IP HTTP API je podle funkce rozděleno do následujících služeb:

- System API – umožňuje změny konfigurace, získání stavu a upgrade interkomu.
- Switch API – umožňuje řízení a sledování stavu spínačů, např. otevírání dveřních zámků apod.
- I/O API – umožňuje řízení a sledování logických vstupů a výstupů interkomu.
- Audio API – umožňuje řízení přehrávání zvuků a monitorování mikrofonu zařízení.
- Camera API – umožňuje řízení a sledování obrazu z kamery.
- Display API – umožňuje řízení displeje a zobrazování uživatelských informací na displeji.
- E-mail API – umožňuje ze zařízení odesílat uživatelské e-maily.
- Phone/Call API – umožňuje řízení a sledování příchozích a odchozích hovorů.
- Logging API – u možňuje vyčítat zaznamenané události zařízení

Pro každou službu lze nastavit transportní protokol (HTTP nebo HTTPS) a způsob autentizace (žádná, Basic nebo Digest). V konfiguraci HTTP API lze vytvořit až pět uživatelských účtů (s vlastním jménem a heslem) s možností detailního řízení přístupu k jednotlivým službám a funkcím.

2N® Helios IP HTTP API se konfiguruje pomocí konfiguračního webového rozhraní interkomu v záložce Služby / HTTP API. Zde lze povolovat a konfigurovat jednotlivé služby a nastavovat parametry uživatelských účtů. Pro demonstraci a odzkoušení funkce 2N® Helios IP HTTP API slouží speciální nástroj integrovaný v HTTP serveru interkomu dostupný na adrese http(s)://ip_adresa_interkomu/apitest.html.

1.1 Revize dokumentu

Verze	Popis změn
2.15	• Doplněny nové události **TamperSwitchActivated**, **UnauthorizedDoorOpen**, **DoorOpenTooLong**, **LoginBlocked**. • Události rozšířeny o parametr **tzShift** udávající rozdíl mezi místním a UTC časem. • Funkce **email/send** rozšířena o možnost nastavení rozlišení odesílaných obrázků.
2.14	• Doplněny funkce **api/pcap**, **api/pcap/restart**, **api/pcap/stop** pro stahování a řízení záznamu síťového provozu. • Doplněna funkce **audio/test** pro spuštění automatického audio testu. • Doplněna funkce **email/send** pro odesílání e-mailu. • Doplněn nový parametr **response** do funkcí **/api/io/ctrl** a **/api/switch/ctrl**. • Funkce **/call/hangup** rozšířena o nový parametr **reason** umožňující zadat důvod ukončení hovoru. • Doplněny nové události **MotionDetected, NoiseDetected** a **SwitchStateChanged**. • Událost **CallStateChanged** rozšířena o parametr **reason** specifikující důvod ukončení hovoru.
2.13	• První verze dokumentu.

2. Popis protokolu HTTP API

Všechny příkazy **HTTP API** jsou odesílány pomocí **HTTP** nebo **HTTPS** protokolu na adresu interkomu s absolutní cestou doplněným prefixem **/api**. Volba protokolu závisí na aktuálním nastavení interkomu v sekci **Služby / HTTP API**. Funkce **HTTP API** jsou rozděleny do služeb a u každé služby je možné nastavit požadovanou úroveň zabezpečení včetně požadavku na **TLS** spojení (tj. **HTTPS** protokol)

Příklad: *Sepnutí spínače 1* **http://10.0.23.193/api/switch/ctrl?switch=1&action=on**

Absolutní cesta obsahuje název skupiny funkcí (systém, firmware, config, switch apod.) a název funkce samotné (caps, status, ctrl apod.). Minimalistická varianta požadavku akceptovaná interkomem musí obsahovat řádek požadavku s metodou a absolutní cestou následovaný hlavičkou Host:

Příklad:

```
GET /api/system/info HTTP/1.1
Host: 10.0.23.193
HTTP Server interkomu odpoví zprávou:
HTTP/1.1 200 OK
Server: HIP2.10.0.19.2
Content-Type: application/json
Content-Length: 253
{
  "success" : true,
  "result" : {
    "variant" : "2N Helios IP Vario",
    "serialNumber" : "08-1860-0035",
    "hwVersion" : "535v1",
    "swVersion" : "2.10.0.19.2",
    "buildType" : "beta",
    "deviceName" : "2N Helios IP Vario"
  }
}
```

V této kapitole dále naleznete:

- 2.3 Odpovědi na požadavky

2.1 Metody HTTP protokolu

2N Helios IP využívá následující čtyři metody HTTP protokolu:

- GET – požadavky stahující obsah z interkomu nebo provádějící obecné příkazy
- POST – požadavky stahující obsah z interkomu nebo provádějící obecné příkazy
- PUT – požadavky pro nahrávání obsahu do interkomu
- DELETE – požadavky pro odstranění obsahu z interkomu

Metody GET a POST jsou z hlediska 2N® Helios IP HTTP API ekvivalentní a liší se pouze způsobem předávání parametrů (viz následující kapitola). Metody PUT a DELETE se používají k manipulaci s velkými objekty jako je konfigurace, firmware, obrázky nebo zvukové soubory.

2.2 Parametry požadavků

Prakticky všechny funkce HTTP API mohou mít parametry. Parametry jsou pojmenované (switch, action, width, height, blob-image apod.) a jsou uvedeny v popisu příslušné funkce HTTP API. Parametry je možné v požadavku předávat třemi způsoby, které lze navzájem kombinovat:

1. v cestě požadavku (uri query, metody GET, POST, PUT a DELETE)
2. v obsahu zprávy (application/x-www-form-urlencoded, metody POST a PUT)
3. v obsahu zprávy (multipart/form-data, metody POST a PUT) – RFC-1867

V případě, že jednotlivé metody předávání parametrů se navzájem kombinují, může nastat situace, kdy je parametr v požadavku uveden vícekrát. V tomto případě se dává přednost poslednímu výskytu parametru.

Parametry funkcí HTTP API jsou dvou typů:

1. Parametr s jednoduchou hodnotou (switch, action apod.) může být předán pomocí všech třech výše uvedených metod. Tyto parametry nemají v názvu prefix blob-.
2. Parametr obsahující velká data (např. konfiguraci, firmware, obrázky apod.). Tyto parametry začínají vždy prefixem blob- a lze je předávat pouze pomocí poslední metody (multipart/form-data).

2.3 Odpovědi na požadavky

Odpovědi na požadavky jsou převážně ve formátu JSON. Výjimku tvoří pouze požadavky na stažení binárních dat (uživatelské zvuky, obrázky apod.) nebo konfiguraci intercomu v XML. Formát odpovědi lze rozlišit dle hlavičky Content-Type. Pro JSON jsou definovány tři základní typy odpovědí:

Kladná odpověď bez parametrů

Tato odpověď je odesílána v případě úspěšného provedení požadavku u funkcí nevracejících žádné parametry. Tato odpověď je vždy kombinovaná se stavovým kódem HTTP protokolu 200 OK.

```
{
  "success" : true,
}
```

Kladná odpověď s parametry

Tato odpověď je odesílána v případě úspěšného provedení požadavku u funkcí vracejících doplňkové parametry. Položka result obsahuje další parametry odpovědi poplatné dané funkci. Tato odpověď je vždy kombinovaná se stavovým kódem HTTP protokolu 200 OK.

```
{
  "success" : true,
  "result" : {
    ...
  }
}
```

Záporná odpověď při chybě zpracování požadavku

Tento typ odpovědi je odesílán v případě, že při zpracování požadavku dojde k chybě. Odpověď nese kód chyby (parametr code), její textový popis (parametr description) a případně upřesnění chyby (parametr param). Tato odpověď může být kombinovaná se stavovým kódem HTTP protokolu 200 OK nebo 401 Authorization Required.

```
{
  "success" : false,
  "error" : {
    "code" : 12,
    "param" : "port",
    "description" : "invalid parameter value"
  }
}
```

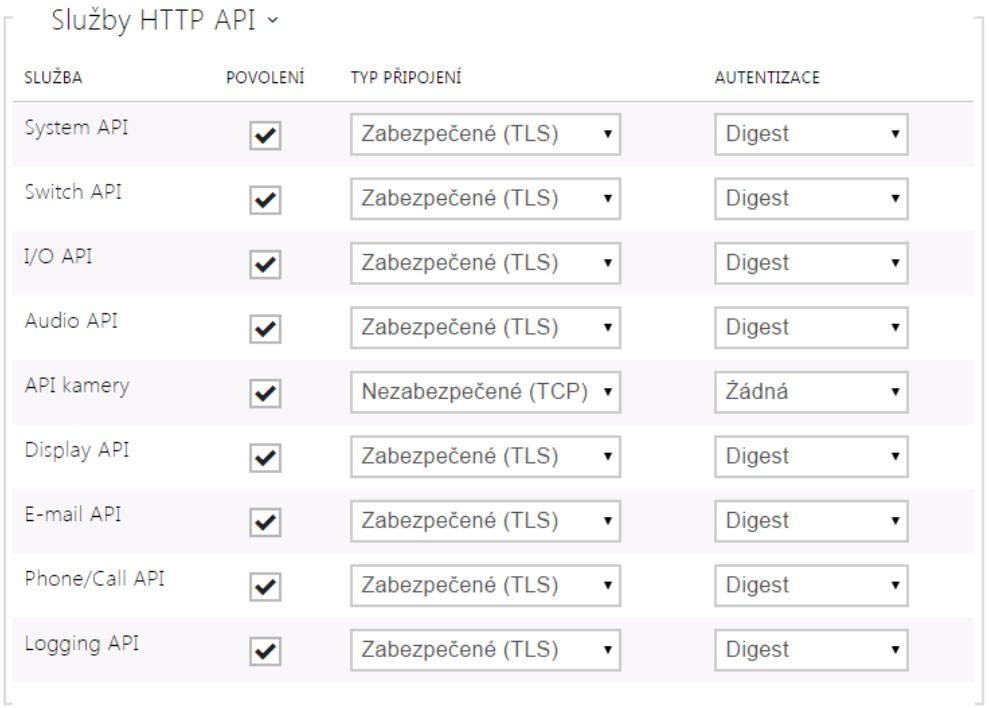
V následující tabulce jsou vyjmenovány možné kódy chyb:

Kód	Popis	
1	function is not supported	Požadovaná funkce není na tomto modelu dostupná.
2	invalid request path	Absolutní cesta specifikovaná v HTTP požadavku neodpovídá žádné z funkcí HTTP API.
3	invalid request method	Použitá metoda HTTP protokolu není pro danou funkci platná.
4	function is disabled	Funkce (příslušná služba) není povolena. Službu je potřeba povolit na stránce konfiguračního rozhraní Služby / HTTP API.
5	function is licensed	Funkce (příslušná služba) je licencovaná a je dostupná pouze po vložení licenčního klíče.
7	invalid connection type	Je vyžadováno HTTPS připojení.
8	invalid authentication method	Použitá autentizační metoda není pro danou službu povolena. Tato chyba může nastat v případě, kdy pro službu je povolena pouze autentizační metoda Digest a klient se pokouší autentizovat pomocí metody Basic.
9	authorization required	Pro přístup ke službě je vyžadovaná autorizace uživatele. Tato chyba je odesílána společně stavovým kódem HTTP protokolu 401 Authorization Required.

Kód	Popis	
10	insufficient user privileges	Autentizovaný uživatel nemá dostatečná privilegia pro provedení funkce.
11	missing mandatory parameter	V požadavku není uveden povinný parametr. Název chybějícího parametru je uveden v položce **param**.
12	invalid parameter value	Hodnota jednoho z parametrů požadavku není platná. Název neplatného parametru je uveden v položce **param**.
13	parameter data too big	Data parametru překračují maximální povolenou velikost. Název chybného parametru je uveden v položce **param**.
14	unspecified processing error	Nastala nespecifikovaná chyba při zpracování požadavku.
15	no data available	Server nemá k dispozici požadovaná data.

3. Zabezpečení služeb HTTP API

V konfiguračním webovém rozhraní **2N Helios IP** na stránce **Služby / HTTP API** lze nastavovat úroveň zabezpečení jednotlivých služeb **HTTP API**. Služby lze vypnout, zapnout nebo nastavit požadovaný komunikační protokol a způsob autentizace uživatelů.



U každé služby lze nezávisle nastavit požadovaný transportní protokol:

- **HTTP** – požadavky mohou být odesílány pomocí **HTTP** nebo **HTTPS** protokolu. Oba protokoly jsou povoleny a úroveň zabezpečení definuje klient použitým protokolem.
- **HTTPS** – požadavky musí být odesílány pomocí **HTTPS** protokolu a požadavky odesílané pomocí nezabezpečeného **HTTP** protokolu jsou interkomem odmítány. **HTTPS** protokol zajišťuje, že případný útočník nemůže číst obsah zpráv odesílaných a přijímaných zpráv.

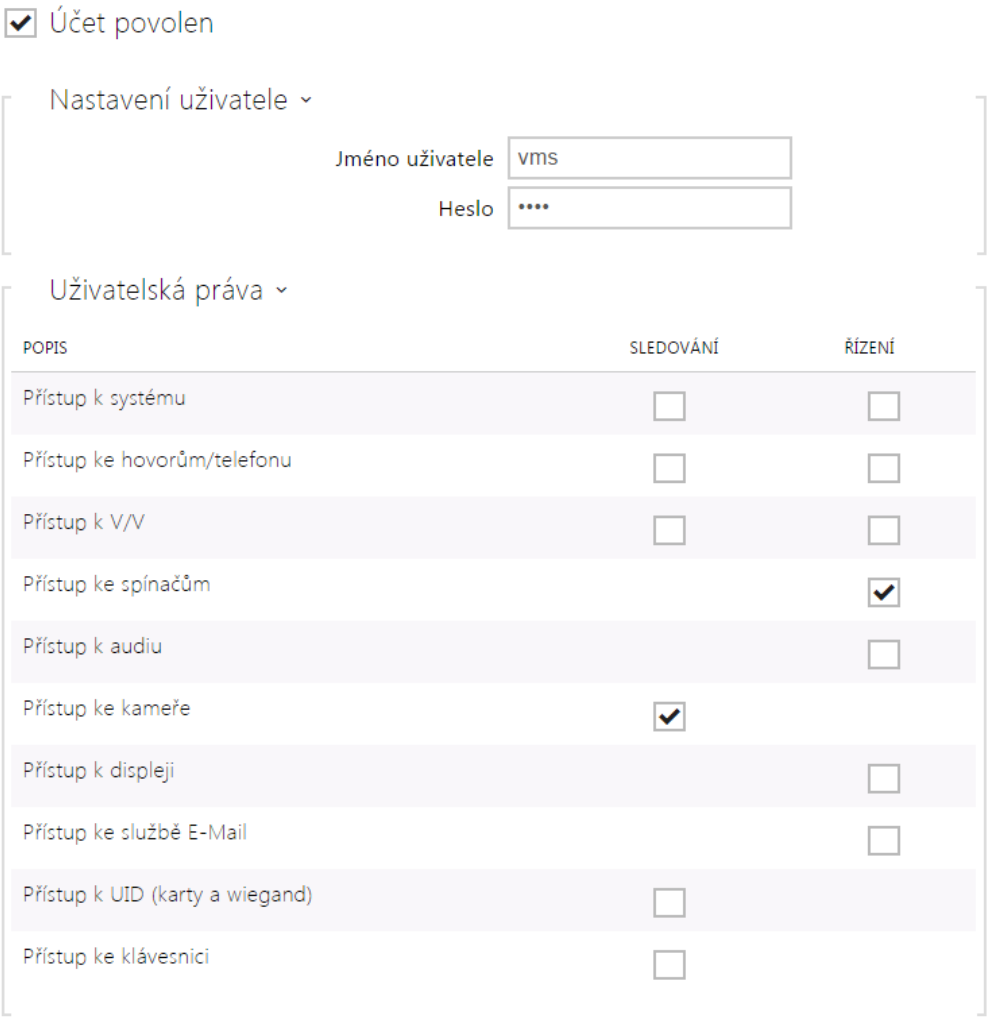
U každé služby lze nastavit vyžadovaný způsob autentizace požadavků odesílaných na interkom. Pokud autentizace není provedena, požadavek je odmítnut. Požadavky jsou autentizovány pomocí standardního autentizačního protokolu popsaného v RFC-2617. Je možné volit tyto tři metody autentizace:

- Žádná – služba nevyžaduje žádnou autentizaci. Služba je v tomto případě v lokální síti zcela nechráněná.
- Basic – služba vyžaduje autentizaci Basic podle RFC-2617. Služba v tomto případě vyžaduje heslo, to je však odesíláno v otevřeném formátu. Doporučujeme tuto volbu kombinovat s HTTPS protokolem, pokud je to možné.
- Digest – služba vyžaduje autentizaci Digest podle RFC-2617. Tato varianta je výchozí a z výše uvedených metod nejbezpečnější.

Pro maximální bezpečnost a odolnost proti zneužití doporučujeme u všech služeb využívat kombinaci HTTPS + Digest. V případě, že druhá strana komunikující s interkomem tuto kombinaci nepodporuje, lze konkrétní službě udělit výjimku a úroveň zabezpečení snížit.

4. Uživatelské účty

2N Helios IP umožňuje spravovat až pět uživatelských účtů určených pro přístup ke službám HTTP API. Součástí uživatelského účtu je jméno a heslo uživatele a tabulka přístupových práv uživatele k jednotlivým službám HTTP API.



Pomocí tabulky přístupových práv lze řídit privilegia uživatelského účtu k jednotlivým službám.

5. Přehled funkcí HTTP API

V tabulce níže je souhrn všech dostupných funkci HTTP API. Tabulka obsahuje následující informace:

- absolutní cesta HTTP požadavku
- podporované HTTP metody
- služba, ve které se funkce nachází
- vyžadovaná privilegia uživatele (v případě, že se využívá autentizace)
- zda je funkce licencovaná (tzn. je dostupná až po vložení licenčního klíče zahrnujícího licenci Enhanced Integration)

Absolutní cesta	Metoda	Služba	Potřebná privilegia	Licence
/api/system/info	GET/POST	System	System Control	Ne
/api/system/status	GET/POST	System	System Control	Ano
/api/system/restart	GET/POST	System	System Control	Ano
/api/firmware	PUT	System	System Control	Ano
/api/firmware/apply	GET/POST	System	System Control	Ano
/api/config	GET/POST/PUT	System	System Control	Ano
/api/config/factoryreset	GET/POST	System	System Control	Ano
/api/switch/caps	GET/POST	Switch	Switch Monitoring	Ano
/api/switch/status	GET/POST	Switch	Switch Monitoring	Ano
/api/switch/ctrl	GET/POST	Switch	Switch Control	Ano

Absolutní cesta	Metoda	Služba	Potřebná privilegia	Licence
/api/io/caps	GET/POST	I/O	I/O Monitoring	Ano
/api/io/status	GET/POST	I/O	I/O Monitoring	Ano
/api/io/ctrl	GET/POST	I/O	I/O Control	Ano
/api/phone/status	GET/POST	Phone/Call	Call Monitoring	Ano
/api/call/status	GET/POST	Phone/Call	Call Monitoring	Ano
/api/call/dial	GET/POST	Phone/Call	Call Control	Ano
/api/call/answer	GET/POST	Phone/Call	Call Control	Ano
/api/call/hangup	GET/POST	Phone/Call	Call Control	Ano
/api/camera/caps	GET/POST	Camera	Camera Monitoring	Ne
/api/camera/snapshot	GET/POST	Camera	Camera Monitoring	Ne
/api/display/caps	GET/POST	Display	Display Control	Ano
/api/display/image	PUT/DELETE	Display	Display Control	Ano
/api/log/caps	GET/POST	Logging	*	Ne
/api/log/subscribe	GET/POST	Logging	*	Ne
/api/log/unsubscribe	GET/POST	Logging	*	Ne
/api/log/pull	GET/POST	Logging	*	Ne
/api/audio/test	GET/POST	Audio	Audio Control	Ano
/api/email/send	GET/POST	E-mail	E-mail Control	Ano
/pcap	GET/POST	System	System Control	Ano

Absolutní cesta	Metoda	Služba	Potřebná privilegia	Licence
/pcap/restart	GET/POST	System	System Control	Ano
/pcap/stop	GET/POST	System	System Control	Ano

V této kapitole dále naleznete:

- 5.31 api pcap stop

5.1 api system info

Funkce **/api/system/info** slouží k získání základních informací o zařízení, jako je typ, výrobní číslo, verze firmware apod. Funkce je dostupná na všech typech zařízení bez ohledu na nastavená přístupová práva.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a obsahuje souhrn informací o zařízení:

Parametr	Popis
variant	Název modelu (varianty) zařízení
serialNumber	Sériové (výrobní) číslo zařízení
hwVersion	Verze hardware
swVersion	Verze firmware
buildType	Typ sestavení firmware (alpha, beta, případně prázdná hodnota pro oficiální verze)
deviceName	Název zařízení nastavení v konfiguračním rozhraní v sekci **Služby / Web Server**

Příklad:

```
GET /api/system/info
{
  "success" : true,
  "result" : {
    "variant" : "2N Helios IP Vario",
    "serialNumber" : "08-1860-0035",
    "hwVersion" : "535v1",
    "swVersion" : "2.10.0.19.2",
    "buildType" : "beta",
    "deviceName" : "2N Helios IP Vario"
  }
}
```

5.2 api system status

Funkce **/api/system/status** vrací aktuální stav interkomu.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Systém Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a obsahuje aktuální stav zařízení:

Parametr	Popis
systemTime	Reálný čas v zařízení v sekundách od 00:00 1.1.1970 (unix time)
upTime	Doba chodu zařízení od posledního restartu v sekundách.

Příklad:

```
GET /api/system/status
{
  "success" : true,
  "result" : {
    "systemTime" : 1418225091,
    "upTime" : 190524
  }
}
```

5.3 api system restart

Funkce **/api/system/restart** provede restart interkomu.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby měl uživatel přiřazené privilegium **Systém Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/system/restart
{
  "success" : true
}
```

5.4 api firmware

Funkce **/api/firmware** slouží k uploadu nového firmware do zařízení. Po uploadu firmware je nutné nahraný firmware potvrdit pomocí funkce **/api/firmware/apply**. Až tato operace vyvolá restart zařízení a změnu firmware.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Systém Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít pouze metodu **PUT**.

Parametry požadavku:

Parametr	Popis
blob-fw	Povinný parametr obsahující firmware zařízení

Odpověď je ve formátu **application/json** a obsahuje informaci o uploadovaném firmware.

Parametr	Popis
version	Verze uploadovaného firmware
downgrade	Příznak nastavený, pokud je uploadovaný firmware starší než aktuální

Příklad:

```
PUT /api/firmware
{
  "success" : true,
  "result" : {
    "version" : "2.10.0.19.2",
    "downgrade" : false
  }
}
```

Pokud je nahraný soubor s firmware zařízení poškozen, nebo není určen pro toto zařízení, interkom vrací odpověď s chybovým kódem 12 – invalid parameter value.

5.5 api firmware apply

Funkce **/api/firmware/apply** slouží k potvrzení dříve nahraného firmware (funkcí **PUT /api/firmware**) a následnému restartu zařízení.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Systém Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/firmware/apply
{
  "success" : true
}
```

5.6 api config

Funkce **/api/config** slouží k uploadu nebo downloadu konfigurace zařízení.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Systém Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST** pro download konfigurace nebo metodu **PUT** pro upload konfigurace.

Parametry požadavku pro metodu **PUT**:

Parametr	Popis
blob-cfg	Povinný parametr obsahující konfiguraci zařízení (ve formátu XML).

Pro metody GET/POST nejsou definovány žádné parametry.

V případě downloadu konfigurace je odpověď ve formátu **application/xml** a obsahuje kompletní konfigurační soubor zařízení.

Funkce **/api/config** s použitím metody **PUT** provádí upload konfigurace se zpožděním cca 15 s, během tohoto intervalu se zařízení nesmí resetovat ani vypínat.

Příklad:

```
 GET /api/config
<?xml version="1.0" encoding="UTF-8"?>
<!--
        Product name: 2N Helios IP Vario
       Serial number: 08-1860-0035
    Software version: 2.10.0.19.2
    Hardware version: 535v1
  Bootloader version: 2.10.0.19.1
             Display: No
         Card reader: No
-->
<DeviceDatabase Version="4">
<Network>
    <DhcpEnabled>1</DhcpEnabled>
    ...
    ...
```

V případě uploadu konfigurace je odpověď ve formátu **application/json** a neobsahuje žádné další parametry.

Příklad:

```
PUT /api/config
{
  "success" : true
}
```

5.7 api config factoryreset

Funce **/api/config/factoryreset** nastaví všechny parametry interkomu do výchozího stavu. Tato funkce je shodná se stejnojmennou funkcí webového konfiguračního rozhraní Systém / Údržba – Výchozí nastavení.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Systém Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Funkce **/api/config/factoryreset** nastavuje interkom do výchozího stavu se zpožděním cca 15 s, během tohoto intervalu se interkom nesmí resetovat ani vypínat.

Příklad:

```
GET /api/config/factoryreset
{
  "success" : true
}
```

5.8 api switch caps

Funkce **/api/switch/caps** vrací aktuální nastavení a možnosti řízení spínačů. Funkce má volitelný parametr **switch**, který určuje spínač, jehož vlastnosti a nastavení se mají vrátit. Pokud parametr **switch** není uveden, funkce vrací stav všech spínačů.

Funkce je součástí služby **Switch** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Switch Monitoring**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
switch	Volitelný parametr definující číslo spínače (obvykle 1 až 4).

Odpověď je ve formátu **application/json** a obsahuje seznam spínačů (pole **switches**) a jejich aktuální nastavení. V případě použití parametru **switch** obsahuje pole **switches** právě jednu položku.

Parametr	Popis
switch	ID spínače (1 až 4)
enabled	Řízení spínače je povoleno v konfiguračním webovém rozhraní.
mode	Nastavený režim spínače (**monostable**, **bistable**)
switchOnDuration	Doba sepnutí spínače v sekundách (jen pro monostabilní režim)
type	Typ spínače (**normal**, **security**)

Příklad:

```
GET /api/switch/caps
{
  "success" : true,
  "result" : {
    "switches" : [
      {
        "switch" : 1,
        "enabled" : true,
        "mode" : "monostable",
        "switchOnDuration" : 5,
        "type" : "normal"
      },
      {
        "switch" : 2,
        "enabled" : true,
        "mode" : "monostable",
        "switchOnDuration" : 5,
        "type" : "normal"
      },
      {
        "switch" : 3,
        "enabled" : false
      },
      {
        "switch" : 4,
        "enabled" : false
      }
    ]
  }
}
```

5.9 api switch status

Funkce **/api/switch/status** vrací aktuální stav spínačů. Funkce má volitelný parametr **switch**, který určuje spínač, jehož stav se má vrátit. Pokud parametr **switch** není uveden, funkce vrací stav všech spínačů.

Funkce je součástí služby **Switch** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Switch Monitoring**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
Switch	Volitelný parametr definující číslo spínače (obvykle 1 až 4). Přesný počet spínačů lze také získat pomocí funkce **/api/switch/caps**.

Odpověď je ve formátu **application/json** a obsahuje seznam spínačů (pole **switches**) a jejich aktuální stav (položka **active**). V případě použití parametru **switch** obsahuje pole **switches** právě jednu položku.

Příklad:

```
GET /api/switch/status
{
  "success" : true,
  "result" : {
    "switches" : [
      {
        "switch" : 1,
        "active" : false
      },
      {
        "switch" : 2,
        "active" : false
      },
      {
        "switch" : 3,
        "active" : false
      },
      {
        "switch" : 4,
        "active" : false
      }
    ]
  }
}
```

5.10 api switch ctrl

Funkce **/api/switch/ctrl** řídí stav spínačů. Funkce má povinný parametr **switch**, který určuje řízený spínač a povinný parametr **action** definující akci provedenou nad spínačem (sepnutí, vypnutí, překlopení).

Funkce je součástí služby **Switch** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Switch Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
switch	Povinný parametr definující číslo spínače (obvykle 1 až 4), přesný počet spínačů lze také získat pomocí funkce **/api/switch/caps**.
action	Povinný parametr definující akci (**on** – sepnutí spínače, **off** – vypnutí spínače, **trigger** – překlopení stavu spínače).
response	Nepovinný parametr umožňující modifikovat odpověď interkomu, tak aby neobsahovala JSON zprávu, ale obyčejný text odpovídající tomuto parametru.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/switch/ctrl?switch=1&action=trigger
{
  "success" : true
}
```

V případě použití parametru **response** odpověď interkomu neobsahuje zprávy **json**, server vrací odpověď typu text/plain se zadaným textem (zadaný text může být prázdný).

Příklad:

```
GET /api/switch/ctrl?switch=1&action=on&response=text
text
```

```
GET /api/switch/ctrl?switch=1&action=on&response=
```

5.11 api io caps

Funkce **/api/io/caps** vrací seznam dostupných hardwarových vstupů a výstupů zařízení (portů). Funkce má volitelný parametr **port**, který určuje vstup/výstup, jehož vlastnosti se mají vrátit. Pokud parametr **port** není uveden, funkce vrací seznam všech vstupů a vstupů.

Funkce je součástí služby **I/O** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **I/O Monitoring**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
Port	Volitelný parametr definující identifikátor vstupu nebo výstupu.

Odpověď je ve formátu **application/json** a obsahuje seznam portů (pole **ports**) a jejich aktuální nastavení. V případě použití parametru **port** obsahuje pole **ports** právě jednu položku.

Parametr	Popis
port	Identifikátor vstupu nebo výstupu
type	Typ (**input** – pro digitální vstupy, **output** – pro digitální výstupy)

Příklad:

```
GET /api/io/caps
{
  "success" : true,
  "result" : {
    "ports" : [
      {
        "port" : "relay1",
        "type" : "output"
      },
      {
        "port" : "relay2",
        "type" : "output"
      }
   ]
  }
}
```

5.12 api io status

Funkce **/api/io/status** vrací aktuální stav logických vstupů a výstupů zařízení (portů). Funkce má volitelný parametr **port**, který určuje vstup/výstup, jehož stav se má vrátit. Pokud parametr **port** není uveden, funkce vrací stav všech vstupů a výstupů.

Funkce je součástí služby **I/O** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **I/O Monitoring**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
port	Volitelný parametr definující identifikátor vstupu nebo výstupu. Identifikátory dostupných vstupů a výstupů lze získat pomocí funkce **/api/io/caps**.

Odpověď je ve formátu **application/json** a obsahuje seznam portů (pole **ports**) a jejich aktuální stav (položka **state**). V případě použití parametru **port** obsahuje pole **ports** právě jednu položku.

Příklad:

```
 GET /api/io/status
{
  "success" : true,
  "result" : {
    "ports" : [
      {
        "port" : "relay1",
        "state" : 0
      },
      {
        "port" : "relay2",
        "state" : 0
      }
    ]
  }
}
```

5.13 api io ctrl

Funkce **/api/io/ctrl** řídí stav logických výstupu zařízení. Funkce má povinný parametr **port**, který určuje řízený výstup a povinný parametr **action** definující akci provedenou nad spínačem (sepnutí, vypnutí).

Funkce je součástí služby **I/O** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **I/O Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
port	Povinný parametr definující identifikátor vstupu nebo výstupu. Identifikátory dostupných vstupů a výstupů lze získat pomocí funkce **/api/io/caps**.
action	Povinný parametr definující akci (**on** – sepnutí výstupu, log.1, **off** – vypnutí výstupu, log.0).
response	Nepovinný parametr umožňující modifikovat odpověď interkomu tak, aby neobsahovala JSON zprávu, ale obyčejný text odpovídající tomuto parametru.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/io/ctrl?port=relay1&action=on
{
  "success" : true
}
```

V případě použití parametru **response** odpověď interkomu neobsahuje zprávy **json**, server vrací odpověď typu text/plain se zadaným textem (zadaný text může být prázdný).

Příklad:

```
GET /api/io/ctrl?port=relay1&action=on&response=text
text
```

```
GET /api/io/ctrl?port=relay1&action=on&response=
```

5.14 api phone status

Funkce **/api/phone/status** slouží k získání aktuálního stavu SIP účtů zařízení.

Funkce je součástí služby **Phone/Call** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Phone/Call Monitoring**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
account	Volitelný parametr definující identifikátor SIP účtu (1 nebo 2). Pokud parametr není uveden, funkce vrací stav všech SIP účtů.

Odpověď je ve formátu **application/json** a obsahuje seznam SIP účtů zařízení (pole **accounts**) a jejich aktuální stav. V případě použití parametru **account** obsahuje pole **accounts** právě jednu položku.

Parametr	Popis
account	Jednoznačný identifikátor SIP účtu (1 nebo 2).
sipNumber	Telefonní číslo SIP účtu.
registered	Signalizuje, zda je účet úspěšně zaregistrován u SIP registraru.
registerTime	Čas poslední úspěšné registrace v sekundách od 00:00 1.1.1970 (unix time).

Příklad:

```
GET /api/phone/status
{
  "success" : true,
  "result" : {
    "accounts" : [
      {
        "account" : 1,
        "sipNumber" : "5046",
        "registered" : true,
        "registerTime" : 1418034578
      },
      {
        "account" : 2,
        "sipNumber" : "",
        "registered" : false
      }
    ]
  }
}
```

5.15 api call status

Funkce **/api/call/status** slouží k získání aktuální stavu probíhajících telefonního hovorů. Funkce vrací seznam probíhajících hovorů a jejich parametry.

Funkce je součástí služby **Phone/Call** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Phone/Call Monitoring**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
session	Volitelný parametr obsahující identifikátor hovoru, jehož stav se má vrátit. Pokud parametr není uveden, funkce vrací stav všech probíhajících hovorů.

Odpověď je ve formátu **application/json** a obsahuje seznam probíhajících hovorů (pole **sessions**) a jejich aktuální stav. V případě použití parametru **session** obsahuje pole **sessions** právě jednu položku. Pokud aktuálně neprobíhá žádný hovor, pole **sessions** je prázdné.

Parametr	Popis
session	Identifikátor hovoru.
direction	Směr hovoru (**incoming** – příchozí, **outgoing** – odchozí)
state	Stav hovoru (**connecting**, **ringing**, **connected**)

Příklad:

```
GET /api/call/status
{
  "success" : true,
  "result" : {
    "sessions" : [
      {
        "session" : 1,
        "direction" : "outgoing",
        "state" : "ringing"
      }
    ]
  }
```

5.16 api call dial

Funkce **/api/call/dial** umožňuje iniciovat nový odchozí hovor na zvolené telefonní číslo nebo sip uri.

Funkce je součástí služby **Phone/Call** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Phone/Call Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
number	Povinný parametr specifikující cílové telefonní číslo nebo sip uri.

Odpověď je ve formátu **application/json** a obsahuje informace o vytvořeném odchozím hovoru.

Parametr	Popis
session	Identifikátor hovoru, který lze použít např. pro sledování hovoru pomocí funkce **/api/call/status**, příp. pro ukončení hovoru funkcí **/api/call/hangup**.

Příklad:

```
GET /api/call/dial?number=sip:1234@10.0.23.194
{
  "success" : true,
  "result" : {
    "session" : 2
  }
}
```

5.17 api call answer

Funkce **/api/call/answer** umožňuje vyzvednout probíhající příchozí hovor (ve stavu **ringing**).

Funkce je součástí služby **Phone/Call** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Phone/Call Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
session	Identifikátor probíhajícího příchozího hovoru.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/call/answer?session=3
{
  "success" : true
}
```

5.18 api call hangup

Funkce **/api/call/hangup** umožňuje ukončit probíhající příchozí nebo odchozí hovor.

Funkce je součástí služby **Phone/Call** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Phone/Call Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
session	Identifikátor probíhajícího příchozího nebo odchozího hovoru.
reason	Důvod ukončení hovoru: **"normal"** – běžné ukončení hovoru (výchozí hodnota) **"rejected"** – signalizace odmítnutí hovoru **"busy"** – signalizace obsazení stanice

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/call/hangup?session=4
{
  "success" : true
}
```

5.19 api camera caps

Funkce **/api/camera/caps** vrací seznam možných zdrojů videa a variant rozlišení JPEG snímků, které lze stahovat pomocí funkce **/api/camera/snapshot**.

Funkce je součástí služby **Camera** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Camera Monitoring**.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a obsahuje seznam podporovaných rozlišení JPEG snímků (pole **jpegResolution**) a seznam dostupných zdrojů obrazu (pole **sources**), které lze použít v parametrech funkce **/api/camera/snapshot**.

Parametr	Popis
width, height	Rozlišení snímku v pixelech
source	Identifikátor zdroje obrazu

Příklad:

```
GET /api/camera/caps
{
  "success" : true,
  "result" : {
    "jpegResolution" : [
      {
        "width" : 160,
        "height" : 120
      },
      {
        "width" : 176,
        "height" : 144
      },
      {
        "width" : 320,
        "height" : 240
      },
      {
        "width" : 352,
        "height" : 272
      },
      {
        "width" : 352,
        "height" : 288
      },
      {
        "width" : 640,
        "height" : 480
      }
    ],
    "sources" : [
      {
        "source" : "internal"
      },
      {
        "source" : "external"
      }
    ]
  }
}
```

5.20 api camera snapshot

Funkce **/api/camera/snapshot** umožňuje stažení obrázku z interní nebo externí IP kamery připojené k interkomu. Pomocí parametrů lze specifikovat zdroj obrázku, rozlišení apod.

Funkce je součástí služby **Camera** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Camera Monitoring**.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
width	Povinný parametr specifikující horizontální rozlišení JPEG snímku v pixelech.
height	Povinný parametr specifikující vertikální rozlišení JPEG snímku v pixelech. Výška a šířka snímku musí odpovídat jedné z podporovaných variant (viz funkce **api/camera /caps**).
source	Volitelný parametr definující zdroj videa (**internal** – interní kamera, **external** – externí IP kamera). Pokud parametr není uveden, je zvolen výchozí zdroj videa uvedený v konfiguračním webovém rozhraní v sekci Hardware / Kamera / Společné nastavení.
fps	Volitelný parametr definující snímkovou frekvenci. Pokud je parametr nastaven na hodnotu >= 1, interkom odesílá s nastavenou snímkovou frekvencí obrázky metodou **http server push.**

Odpověď je ve formátu **image/jpeg** příp. **multipart/x-mixed-replace** (pro fps >= 1). V případě chybných parametrů požadavku, funkce vrací informaci ve formátu **application /json**.

Příklad:

```
GET /api/camera/snapshot?width=640&height=480&source=internal
```

5.21 api display caps

Funkce **/api/display/caps** vrací seznam displejů v zařízení a jejich vlastnosti. Funkci lze použít pro detekci displeje a získání jeho rozlišení.

Funkce je součástí služby **Display** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Display Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a obsahuje seznam dostupný displejů (pole **displays**).

Parametr	Popis
display	Identifikátor displeje
resolution	Rozlišení displeje v pixelech

Příklad:

```
GET /api/display/caps
{
  "success" : true,
  "result" : {
    "displays" : [
      {
        "display" : "internal",
        "resolution" : {
          "width" : 320,
          "height" : 240
        }
      }
    ]
  }
}
```

5.22 api display image

Funkce **/api/display/image** umožňuje modifikovat obsah zobrazovaný na displeji zařízení. Umožňuje nahrát libovolný obrázek ve formátu GIF, příp. nahraný obrázek z displeje odstranit.

Poznámka

- Funkce je dostupná pouze, pokud je vypnuta standardní funkce displeje v konfiguračním rozhraní v sekci Hardware / Display.

Funkce je součástí služby **Display** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Display Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **PUT** nebo **DELETE**. Metoda **PUT** slouží k uploadu obrázku na displej. Metoda **DELETE** slouží k odstranění dříve uploadovaného obrázku z displeje.

Parametry požadavku:

Parametr	Popis
display	Povinný identifikátor displeje (**internal**)
blob-image	Povinný parametr obsahující obrázek ve formátu GIF s rozlišením daného displeje (viz funkce **/api/display/caps**). Parametr se uplatní pouze v případě metody **PUT**.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
DELETE /api/display/image?display=internal
{
  "success" : true
}
```

5.23 api log caps

Funkce **/api/log/caps** vrací seznam typů podporovaných událostí, které se na daném zařízení zaznamenávají. Vrácený seznam je podmnožinou kompletního seznamu typů událostí uvedeného v následující tabulce:

Typ události	Popis	Poznámka
DeviceState	Systémová událost generovaná při změnách stavu zařízení.	
AudioLoopTest	Signalizuje provedení automatického audio loop testu a jeho výsledek.	pouze po zadání platného licenčního klíče **Enhanced Audio**
MotionDetected	Signalizuje detekci pohybu pomocí kamery.	pouze modely vybavené kamerou
NoiseDetected	Signalizuje detekci zvýšené hladiny hluku	pouze modely vybavené mikrofonem nebo mikrofonním vstupem
KeyPressed	Signalizuje stisk tlačítka rychlé volby nebo klávesy numerické klávesnice.	
KeyReleased	Signalizuje uvolnění tlačítka rychlé volby nebo klávesy numerické klávesnice.	
CodeEntered	Signalizuje zadání kódu uživatelem pomocí numerické klávesnice.	pouze modely vybavené numerickou klávesnicí
CardEntered	Signalizuje přiložení RFID karty ke čtečce.	pouze modely vybavené čtečkou RFID karet
InputChanged	Signalizuje změnu stavu logického vstupu.	
OutputChanged		

Typ události	Popis	Poznámka
	Signalizuje změnu stavu logického výstupu.	
SwitchStateChanged	Signalizuje změnu stavu spínače 1–4	
CallStateChanged	Signalizuje vytvoření, ukončení, příp. jinou změnu stavu probíhajícího hovoru.	
RegistrationStateChanged	Signalizuje změnu stavu registrace zařízení k SIP serveru.	
TamperSwitchActivated	Signalizuje aktivaci ochranného spínače.	pouze modely vybavené ochranným spínačem
UnauthorizedDoorOpen	Signalizuje neautorizované otevření dveří.	pouze modely vybavené digitálními vstupy
DoorOpenTooLong	Signalizuje dlouhé otevření dveří, resp. nezavření dveří do nastavené doby.	pouze modely vybavené digitálními vstupy
LoginBlocked	Signalizuje dočasné zablokování přístupu k webovému rozhraní.	

Funkce je součástí služby **Logging** a pro provedení funkce nejsou potřeba žádná zvláštní privilegia uživatele.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json**:

Parametr	Typ	Popis
events	array	Pole řetězců obsahující seznam podporovaných typů událostí.

Příklad:

```
GET /api/log/caps
{
  "success" : true,
  "result" : {
    "events" : [
      "KeyPressed",
      "KeyReleased",
      "InputChanged",
      "OutputChanged",
      "CardEntered",
      "CallStateChanged",
      "AudioLoopTest",
      "CodeEntered",
      "DeviceState",
      "RegistrationStateChanged"
    ]
  }
}
```

5.24 api log subscribe

Funkce **/api/log/subscribe** vytvoří kanál pro odběr událostí (subscription) a vrací unikátní identikátor, který se použije při následném volání funkcí **/api/log/pull**, příp. **/api/log/unsubscribe**.

Každý kanál pro odběr událostí obsahuje vlastní frontu událostí. Do fronty kanálu jsou ukládány všechny nové události, které odpovídají filtru kanálu (parametr **filter**). Události z fronty kanálu lze číst pomocí funkce **/api/log/pull**.

Současně zařízení udržují v interní paměti frontu historie událostí (posledních 500 události). Po restartu zařízení je tato fronta historie vždy prázdná.

Pomocí parametru **include** lze specifikovat, zda fronta kanálu bude na počátku prázdná (tj. budou do ní zapsány pouze nové události, které vzniknou po vytvoření kanálu), příp. zda má být jednorázově naplněna událostmi z části nebo celé zaznamenané historie událostí.

Pomocí parametru **duration** lze specifikovat životnost kanálu v případě, že se k němu nepřistupuje pomocí funkce **/api/log/pull**. Po nastavené době bude nepoužívaný kanál automaticky uzavřen, jako by byla použita funkce **/api/log/unsubscribe**.

Funkce je součástí služby **Logging** a v případě použití autentizace je nutné pro některé události nastavit privilegia uživatele podle tabulky níže. Do fronty kanálu nebudou zařazovány události, pro které autentizovaný uživatel nemá požadovaná privilegia.

Typ události	Vyžadovaná privilegia uživatele
DeviceState	žádná
AudioLoopTest	žádná
MotionDetected	žádná
NoiseDetected	žádná
KeyPressed	monitorování klávesnice
KeyReleased	monitorování klávesnice
CodeEntered	monitorování klávesnice
CardEntered	monitorování UID (karty/wiegand)

Typ události	Vyžadovaná privilegia uživatele
InputChanged	monitorování V/V
OutputChanged	monitorování V/V
SwitchStateChanged	monitorování V/V
CallStateChanged	monitorování hovorů/telefonu
RegistrationStateChanged	monitorování hovorů/telefonu
TamperSwitchActivated	žádná
UnauthorizedDoorOpen	žádná
DoorOpenTooLong	žádná
LoginBlocked	žádná

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Typ	Povinný	Výchozí hodnota	Popis
include	string	Ne	new	Určuje, zda má být fronta událostí kanálu naplněna položkami z historie: **new** – pouze nové události, které vzniknou až po vytvoření kanálu **all** – všechny dosud zaznamenané událost včetně těch, které vzniknou až po vytvoření kanálu **-t** – všechny dosud zaznamenané události za posledních **t** sekund včetně těch, které vzniknou až po vytvoření kanálu (např. -10)
filter	list	Ne	bez filtru	

Parametr	Typ	Povinný	Výchozí hodnota	Popis
				Seznam typů požadovaných událostí (názvy typů událostí oddělené čárkou). Parametr je nepovinný a pokud není uveden, pak se v rámci kanálu předávají všechny dostupné typy událostí.
duration	uint32	Ne	90	Definuje dobu v sekundách, po které bude kanál automaticky uzavřen, pokud na něm nebudou probíhat žádné operace čtení pomocí **/api/log/pull**. Každým čtením z kanálu je automaticky životnost kanálu prodloužena o zde nastavenou hodnotu. Maximální možná hodnota je 3600 s.

Odpověď je ve formátu **application/json** a obsahuje pouze identifikátor vytvořeného subscription.

Parametr	Typ	Popis
id	uint32	Unikátní identifikátor vytvořeného subscription.

Příklad:

```
GET /api/log/subscribe?filter=KeyPressed,InputChanged
{
  "success" : true,
  "result" : {
    "id" : 2121013117
  }
}
```

5.25 api log unsubscribe

Funkce **/api/log/unsubscribe** uzavře kanál odběru událostí (subscription) s daným identifikátorem. Po provedení funkce nebude možné daný identifikátor použít, tj. následná volání funkce **/api/log/pull**, příp. **/api/log/unsubscribe** se stejným identifikátorem skončí chybou.

Funkce je součástí služby **Logging** a pro provedení funkce nejsou potřeba žádná zvláštní privilegia uživatele.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Typ	Povinný	Výchozí hodnota	Popis
id	uint32	Ano	–	Identifikátor existujícího kanálu získaný při předchozím volání funkce **/api/log/subscribe**.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/log/unsubscribe?id=21458715
{
  "success" : true,
}
```

5.26 api log pull

Funkce **/api/log/pull** provádí čtení položek z fronty kanálu (subscription) a vrací seznam dosud nevyčtených událostí, příp. prázdný seznam, pokud žádná nová událost není k dispozici.

Pomocí parametru **timeout** lze specifikovat maximální dobu, za jakou musí interkom vygenerovat odpověď. V případě, že ve frontě kanálu je alespoň jedna položka, odpověď je vygenerována okamžitě. V případě, že je fronta kanálu prázdná, interkom odloží odeslání odpovědi do doby, než vznikne nová událost, příp. vyprší nastavený timeout.

Funkce je součástí služby **Logging** a pro provedení funkce nejsou potřeba žádná zvláštní privilegia uživatele.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Typ	Povinný	Výchozí hodnota	Popis
id	uint32	Ano	–	Identifikátor existujícího kanálu vytvořeného předchozím voláním funkce **/api/log/subscribe**.
timeout	uint32	Ne	0	Specifikuje zpoždění odpovědi (v sekundách) v případě, že fronta kanálu je prázdná. Výchozí hodnota 0 znamená, že interkom odpovídá vždy okamžitě bez jakéhokoliv zpoždění.

Odpověď je ve formátu **application/json** a obsahuje seznam událostí.

Parametr	Typ	Popis
events	array	Pole objektu události. V případě, že během nastaveného timeoutu nenastala žádná událost, je pole prázdné.

Příklad:

```
GET /api/log/pull
{

  "success" : true,
  "result" : {
    "events" : [
      {
        "id" : 1,
        "tzShift" : 0,
        "utcTime" : 1437987102,
        "upTime" : 8,
        "event" : "DeviceState",
        "params" : {
          "state" : "startup"
        }
      },
      {
        "id" : 3,
        "tzShift" : 0,
        "utcTime" : 1437987105,
        "upTime" : 11,
        "event" : "RegistrationStateChanged",
        "params" : {
          "sipAccount" : 1,
          "state" : "registered"
        }
      }
    ]
  }
}
```

Události

Každá událost v poli **events** obsahuje následující informace, které jsou společné pro všechny typy událostí:

Parametr	Typ	Popis
id	uint32	Interní ID záznamu o události (32bit číslo, 1 po restartu interkomu, inkrementované s každou novou událostí)
utcTime	uint32	Absolutní čas vzniku události (Unix Time, UTC – koordinovaný světový čas).
upTime	uint32	Relativní čas vzniku události (počet sekund od restartu interkomu).
tzShift	int32	Rozdíl mezi místním časem a časem UTC v minutách. Přičtením této hodnoty k utcTime se získá místní čas vzniku události dle nastavení časové zóny v zařízení: localTime = utcTime + tzShift * 60
event	string	Typ události ("KeyPressed", "InputChanged", ...)
params	object	Specifické parametry události.

Událost DeviceState

Signalizuje změny stavu zařízení.

Parametry události:

Parametr	Typ	Popis
state	string	Signalizovaný stav zařízení: "startup" – generováno jednorázově po startu zařízení (vždy úplně první událost)

Příklad:

```
{
  "id" : 1,
  "tzShift" : 0,
  "utcTime" : 1437987102,
  "upTime" : 8,
  "event" : "DeviceState",
  "params" : {
    "state" : "startup"
  }
}
```

Událost AudioLoopTest

Signalizuje provedení automatického audio loop testu a jeho výsledek. Událost **AudioLoopTest** je dostupná pouze na vybraných modelech se zadanou platnou licencí **Enhanced Audio**. Událost je signalizovaná vždy po provedení automatického testu (naplánovaného příp. manuálně spuštěného).

Parametr	Typ	Popis
result	string	Výsledek provedeného testu. **"passed"** – test byl úspěšně proveden a nebyl zjištěn žádný problém **"failed"** – test byl proveden, ale byl detekován problém s reproduktorem nebo mikrofonem v zařízení

Příklad:

```
{
  "id" : 26,
  "tzShift" : 0,
  "utcTime" : 1438073190,
  "upTime" : 9724,
  "event" : "AudioLoopTest",
  "params" : {
    "result" : "passed"
  }
}
```

Událost MotionDetected

Signalizuje detekci pohybu pomocí kamery. Událost je dostupná pouze na modelech vybavených kamerou. Událost se generuje pouze v případě, že v konfiguraci kamery interkomu je tato funkce povolena.

Parametry události:

Parametr	Typ	Popis
state	string	Stav detektoru pohybu: "in" – signalizuje začátek intervalu, ve kterém byl detekován pohyb "out" – signalizuje konec intervalu, ve kterém byl detekován pohyb

Příklad:

```
{
  "id" : 2,
  "tzShift" : 0,
  "utcTime" : 1441357589,
  "upTime" : 1,
  "event" : "MotionDetected",
  "params" : {
    "state" : "in"
  }
}
```

Událost NoiseDetected

Signalizuje zvýšenou hladinu zvuku detekovanou pomocí zabudovaného nebo externího mikrofonu. Událost se generuje pouze v případě, že v konfiguraci interkomu je tato funkce povolena.

Parametry události:

Parametr	Typ	Popis
state	string	Stav detektoru hluku: "in" – signalizuje začátek intervalu, ve kterém byl detekován hluk "out" – signalizuje konec intervalu, ve kterém byl detekován hluk

Příklad:

```
{
  "id" : 2,
  "tzShift" : 0,
  "utcTime" : 1441357589,
  "upTime" : 1,
  "event" : "NoiseDetected",
  "params" : {
    "state" : "in"
  }
}
```

Události KeyPressed a KeyReleased

Signalizuje stisk **(KeyPressed)** nebo uvolnění **(KeyReleased)** tlačítka nebo klávesy numerické klávesnice.

Parametry události:

Parametr	Typ	Popis
key	string	Kód stisknutého nebo uvolněného tlačítka: "0" až "9" – tlačítka numerické klávesnice "%1" – "%150" - tlačítka rychlé volby "*" – tlačítko se symbolem * příp. telefonu "#" – tlačítko se symbolem # příp. klíčku

Příklad:

```
{
  "id" : 4,
  "tzShift" : 0,
  "utcTime" : 1437987888,
  "upTime" : 794,
  "event" : "KeyPressed",
  "params" : {
    "key" : "5"
  }
}
```

Událost CodeEntered

Signalizuje zadání kódu uživatelem na numerické klávesnici interkomu. Událost se generuje pouze na zařízeních vybavených numerickou klávesnicí.

Parametry události:

Parametr	Typ	Popis
code	string	Uživatelem zadaný kód, např. "1234". Kódy mají minimálně dvě číslice a kód "00" nelze použít.
valid	boolean	Platnost zadaného kódu (tj. zda je kód zaveden v konfiguraci interkomu jako platný kód uživatele nebo jako platný univerzální kód spínače). **false** – kód není platný **true** – kód je platný

Příklad:

```
{
  "id" : 23,
  "tzShift" : 0,
  "utcTime" : 1438072978,
  "upTime" : 9512,
  "event" : "CodeEntered",
  "params" : {
    "code" : "5864",
    "valid" : false
  }
}
```

Událost CardEntered

Signalizuje přiložení RFID karty ke čtečce. Událost je dostupná pouze na modelech vybavených čtečkou RFID karet.

Parametry události:

Parametr	Typ	Popis
direction	string	Směr průchodu RFID čtečky: "in" – příchod "out" – odchod "any" – průchod *Pozn.: Směr průchodu čtečky se nastavuje pomocí konfiguračního rozhraní interkomu.*
reader	string	Jméno RFID čtečky příp. Wiegand modulu, příp. jedna z následujících hodnot pro nemodulární modely interkomu: "internal" – interní čtečka (modely 2N Helios) "external" – externí čtečka připojená pomocí Wiegand interface *Pozn.: Jméno čtečky se nastavuje pomocí konfiguračního rozhraní interkomu.*
uid	string	Jednoznačný indentifikátor přiložené karty (číslo v hexadecimálním formátu, 6–16 znaků, podle typu karty).
valid	boolean	Platnost přiložené RFID karty (tj. zda je uid karty přiřazeno jednomu z uživatelů v adresáři interkomu). false – karta není platná true – karta je platná

Příklad:


```
{
  "id" : 26,
  "tzShift" : 0,
  "utcTime" : 1438072979,
  "upTime" : 9513,
  "event" : "CardEntered",
  "params" : {
    "direction" : "any",
    "reader" : "ext7",
    "uid" : "01045E31BB",
    "valid" : false
  }
}
```

Události InputChanged a OutputChanged

Signalizuje změnu stavu logického vstupu **(InputChanged)** nebo výstupu **(OutputChanged).** Aktuální seznam dostupných vstupů a výstupů lze zjistit pomocí funkce **/api/io/caps**.

Parametry události:

Parametr	Typ	Popis
port	string	Název I/O portu.
state	boolean	Aktuální logický stav I/O portu: **false** – neaktivní, log.0 **true** – aktivní, log.1

Příklad:

```
{
  "id" : 2,
  "tzShift" : 0,
  "utcTime" : 1437987103,
  "upTime" : 9,
  "event" : "OutputChanged",
    "params" : {
    "port" : "led_secured",
    "state" : false
  }
}
```

Událost SwitchStateChanged

Signalizuje změnu stavu spínače (viz konfigurace interkomu Hardware | Spínače).

Parametry události:

Parametr	Typ	Popis
switch	uint32	Číslo spínače 1..4
state	boolean	Aktuální logický stav spínače: **false** – neaktivní, log.0 **true** – aktivní, log.1

Příklad:

```
{
  "id" : 2,
  "tzShift" : 0,
  "utcTime" : 1437987103,
  "upTime" : 9,
  "event" : "SwitchStateChanged",
    "params" : {
        "switch" : 1,
        "state" : true
  }
}
```

Událost CallStateChanged

Signalizuje vytvoření, ukončení příp. jinou změnu stavu probíhajícího hovoru.

Parametry události:

Parametr	Typ	Popis
direction	string	Specifikuje, zda se jedná o příchozí nebo odchozí hovor: "incoming" – příchozí hovor "outgoing" – odchozí hovor
state	string	Aktuální stav automatu hovoru: "connecting" – probíhá sestavování hovoru (pouze odchozí hovory) "ringing" – vyzvánění "connected" – hovor spojen "terminated" – hovor ukončen
peer	string	SIP URI volajícího (příchozí hovory) nebo volaného (odchozí hovory) účastníka.
reason	string	Důvod ukončení hovoru. Parametr je přítomný pouze, pokud je signalizován stav pro ukončení hovoru. "normal" – běžné ukončení hovoru "busy" – volaná stanice je obsazena "rejected" – hovor byl odmítnut "noanswer" – volaný účastník neodpovídá "noresponse" – volaná stanice neodpovídá (nereaguje na SIP zprávy) "completed_elsewhere" – hovor byl vyzvednut jinou stanicí (skupinové hovory) "failure" – chyba při sestavování hovoru
session	uint32	Jednoznačný identifikátor hovoru. Identifikátor hovoru lze dále použít ve funkcích /api/call/answer, /api/call/hangup a /api/call/status.
call	uint32	TBD

Příklad:


```
{
  "id" : 5,
  "tzShift" : 0,
  "utcTime" : 1438064126,
  "upTime" : 660,
  "event" : "CallStateChanged",
  "params" : {
    "direction" : "incoming",
    "state" : "ringing",
    "peer" : "sip:2229@10.0.97.150:5062;user=phone",
    "session" : 1,
    "call" : 1
  }
}
```

Událost RegistrationStateChanged

Signalizuje změnu stavu registrace všech SIP účtů k SIP serveru.

Parametry události:

Parametr	Typ	Popis
sipAccount	uint32	Číslo SIP účtu, na kterém proběhla změna stavu: **1** – první SIP účet **2** – druhý SIP účet
state	string	Udává nový stav automatu registrace SIP účtu: **"registered"** – účet byl úspěšně zaregistrován **"unregistered"** – účet byl odregistrován **"registering"** – probíhá registrace **"unregistering"** – probíhá odregistrování účtu

Příklad:

```
{
  "id" : 3,
  "tzShift" : 0,
  "utcTime" : 1437987105,
  "upTime" : 11,
  "event" : "RegistrationStateChanged",
  "params" : {
    "sipAccount" : 1,
    "state" : "registered"
  }
}
```

Událost TamperSwitchActivated

Signalizuje aktivaci ochranného spínače – otevření krytu zařízení. Funkce ochranného spínače musí být nakonfigurována v menu Digitální vstupy / Ochranný spínač.

Parametry události:

Parametr	Typ	Popis
state	string	Stav ochranného spínače: "in" – signalizuje aktivaci ochranného spínače (tj. otevření krytu zařízení) "out" – signalizuje deaktivaci ochranného spínače (tj. zavření krytu zařízení)

Příklad:

```
{
  "id" : 54,
  "tzShift" : 0,
  "utcTime" : 1441357589,
  "upTime" : 158,
  "event" : "TamperSwitchActivated",
  "params" : {
    "state" : "in"
  }
}
```

Událost UnauthorizedDoorOpen

Signalizuje neautorizované otevření dveří. Vyžaduje připojení snímače otevřených dveří na jeden z digitálních vstupů a příslušné nastavení v menu Digitální vstupy | Stav dveří.

Parametry události:

Parametr	Typ	Popis
state	string	Stav neautorizovaného otevření dveří: **"in"** – signalizuje zahájení stavu neautorizovaného otevření **"out"** – signalizuje ukončení stavu neautorizovaného otevření dveří

Příklad:

```
{
  "id" : 80,
  "tzShift" : 0,
  "utcTime" : 1441367842,
  "upTime" : 231,
  "event" : "UnauthorizedDoorOpen",
  "params" : {
    "state" : "in"
  }
}
```

Událost DoorOpenTooLong

Signalizuje dlouhé otevření dveří, resp. nezavření dveří do nastavené doby. Vyžaduje připojení snímače otevřených dveří na jeden z digitálních vstupů a příslušné nastavení v menu Digitální vstupy | Stav dveří.

Parametry události:

Parametr	Typ	Popis
state	string	Stav dlouhého otevření dveří: "in" – signalizuje zahájení stavu dlouhého otevření dveří. "out" – signalizuje ukončení stavu dlouhého otevření dveří.

Příklad:

```
{
  "id" : 96,
  "tzShift" : 0,
  "utcTime" : 1441369745,
  "upTime" : 275,
  "event" : "DoorOpenTooLong",
  "params" : {
    "state" : "out"
  }
}
```

Událost LoginBlocked

Signalizuje dočasné zablokování přístupu k webovému rozhraní z důvodu opakovaného zadání neplatného přihlašovacího jména nebo hesla.

Parametry události:

Parametr	Typ	Popis
address	string	IP adresa, ze které bylo provedeno opakované přihlášení s neplatnými údaji.

Příklad:

```
{
  "id" : 5,
  "tzShift" : 0,
  "utcTime" : 1441369745,
  "upTime" : 275,
  "event" : "LoginBlocked",
  "params" : {
    "address" : "10.0.23.32"
  }
}
```

5.27 api audio test

Funkce **/api/audio/test** spustí automatický test zabudovaného mikrofonu a reproduktoru interkomu. Po provedení testu je výsledek zapsán do logu v zařízení jako událost **AudioLoopTest**.

Funkce je součástí služby **Audio** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Audio Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration a Enhanced Audio.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/audio/test
{
  "success" : true
}
```

5.28 api email send

Funkce **/api/email/send** odešle ze zařízení e-mail na uvedenou adresu. Pro správnou funkci je potřeba nakonfigurovat službu SMTP v zařízení (tj. nastavit adresu SMTP serveru, správné přihlašovací údaje apod.)

Funkce je součástí služby **Email** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **Email Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Parametry požadavku:

Parametr	Popis
to	Povinný parametr obsahující e-mailovou adresu pro doručení.
subject	Povinný parametr specifikující předmět zprávy.
body	Nepovinný parametr specifikující obsah zprávy (může obsahovat html značky). Pokud parametr není uveden, zpráva bude doručena bez obsahu.
pictureCount	Nepovinný parametr specifikující počet přiložených obrázků z kamery. V případě, že není uveden, obrázky nejsou přiloženy. Parametr může nabývat hodnot 0 a 1.
width	Nepovinné parametry nastavující rozlišení přiložených obrázků z kamery. Výška a šířka snímku musí odpovídat jedné z podporovaných variant (viz funkce **api /camera/caps**).
height	

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/email/send?to=somebody@email.com&subject=Hello&body=Hello
{
  "success" : true
}
```

5.29 api pcap

Funkce **/api/pcap** slouží ke stažení zaznamenaného provozu na síťovém rozhraní zařízení (pcap soubor). Záznam síťového provozu lze také řídit pomocí funkcí **/api /pcap/restart** a **/api/pcap/stop**.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **System Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST.**

Funkce nemá žádné parametry.

Odpověď ve formátu **application/pcap** a stažený soubor lze přímo otevřít např. v programu Wireshark.

Příklad:

```
GET /api/pcap
```

5.30 api pcap restart

Funkce **/api/pcap/restart** provede odstranění všech zaznamenaných paketů a znovu spustí záznam provozu na síťovém rozhraní zařízení.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **System Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/pcap/restart
{
  "success" : true
}
```

5.31 api pcap stop

Funkce **/api/pcap/stop** zastaví zaznamenávání provozu na síťovém rozhraní zařízení.

Funkce je součástí služby **System** a v případě použití autentizace je nutné, aby uživatel měl přiřazené privilegium **System Control**. Funkce je dostupná pouze po zadání licenčního klíče zahrnujícího licenci Enhanced Integration.

Pro tuto funkci lze použít metody **GET** nebo **POST**.

Funkce nemá žádné parametry.

Odpověď je ve formátu **application/json** a neobsahuje žádné parametry.

Příklad:

```
GET /api/pcap/restart
{
  "success" : true
}
```

2N TELEKOMUNIKACE a.s.

Modřanská 621, 143 01 Prague 4, Czech Republic

Phone: +420 261 301 500, Fax: +420 261 301 599

E-mail: sales@2n.cz

Web: www.2n.cz

v2.17