API REST

COLLABORATORS

	TITLE : API REST	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	http://dev.efixo.net	08 juin 2012	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
20120608	08 juin 2012		H

Contents

List of Tables

1 ChangeLog

1.1 Firmware 3.2.1

- Méthode ddns.getInfo Section 3.3.4
  - ajout des attributs **lastfreeze**, **lastfreezetime**.
- Nouvelle méthode smb.getInfo Section 3.11.1
- Nouvelle méthode p910nd.getInfo Section 3.9.1
- Nouvelle méthode ont.getInfo Section 3.8.1

1.2 Firmware 3.2.0

- Nouvelle méthode system.setRefClient Section 3.12.5
- Nouvelle méthode lan.getDnsHostList Section 3.7.3
- Nouvelle méthode lan.addDnsHost Section 3.7.1
- Nouvelle méthode lan.deleteDnsHost Section 3.7.2
- Nouvelle méthode wlan.setWl0EncType Section 3.15.7
- Méthode wlan.getInfo Section 3.15.4:
  - ajout de l'attribut **enctype**.
- Méthode lan.getHostsList Section 3.7.4:
  - ajout des attributs **type**, **probe**, **alive**, **status**.
- Méthode system.getInfo Section 3.12.1:
  - ajout des attributs **current_datetime**, **refclient**.
- Nouvelle méthode ddns.getInfo Section 3.3.4
- Nouvelle méthode ddns.setService Section 3.3.5
- Nouvelle méthode ddns.enable Section 3.3.2
- Nouvelle méthode ddns.disable Section 3.3.1
- Nouvelle méthode ddns.forceUpdate Section 3.3.3

1.3 Firmware 3.1.0

- Mise à jour des valeurs des modes hotspot dans hotspot.getInfo Section 3.6.4

1.4 Firmware 3.0.14

- Nouvelle méthode backup3g.getPinCode Section 3.2.3
- Nouvelle méthode backup3g.setPinCode Section 3.2.4

1.5 Firmware 3.0.7

- Nouvelle méthode voip.getCallhistoryList Section 3.13.1
- Nouvelle méthode lan.getHostsList Section 3.7.4
- Nouvelle méthode system.reboot Section 3.12.3
- Méthode lan.getInfo Section 3.7.5:
  - ajout des attributs **dhcp_active**, **dhcp_start**, **dhcp_end** et **dhcp_lease**.
- Méthode voip.getInfo Section 3.13.2:
  - ajout des attributs **hook_status** et **callhistory_active**.
- Méthode wlan.getInfo Section 3.15.4:
  - ajout de l'attribut **mac_filtering**.

1.6 Firmware 3.0.6

- Méthode wan.getInfo Section 3.14.1:
  - la méthode **wan.getInfo** est dorénavent public.

1.7 Firmware 2.1.2

- Nouvelle authentification par login/mot de passe.
- La configuration de l'authentification de l'API REST se base sur celle de l'interface web de configuration (même méthode d'authentification, même login/mot de passe, ...).
- Nouveau module backup3g Section 3.2.
- Méthode wan.getInfo Section 3.14.1:
  - ajout de l'attribut **infra**.
- Méthode voip.getInfo Section 3.13.2:
  - ajout de l'attribut **infra**.
- Méthode lan.getInfo Section 3.7.5:
  - la méthode est dorénavent public.
- Méthode firewall.getInfo Section 3.5.3:
  - le tag "stmpdrop" a été renommé en "smtpdrop" (faute de frappe).
- Méthode dsl.getInfo Section 3.4.1:
  - ajout des attributs **linemode**, **uptime**, **counter**, **crc**.
- Méthode system.getInfo Section 3.12.1:
  - ajout de l'attribut **version_dsldriver**.
  - ajout de l'attribut **net_infra**.
- Correction d'erreurs diverses:
  - La méthode ppp.setCredentials Section 3.10.3 est corrigée.

1.8 Firmware 2.1.1

- Nouvelles valeurs de l'attribut "mode" pour les méthodes "wlan.getInfo" et "wlan.setWlanMode".

2 Introduction

2.1 Utilisation

- L'URL de l'interface REST est http://neufbox/api/1.0/ où 1.0 est le numéro de version de l'interface.
- L'interface peut être testée avec wget ou curl par exemple.

Exemple d'appel d'une méthode avec curl

```
$ curl http://neufbox/api/1.0/?method=auth.getToken
```

- L'interface doit être appelée avec une requête HTTP GET pour les méthodes qui ne font que consulter des informations, et une requête HTTP POST pour les méthodes qui modifient des informations.
- Certaines méthodes sont privées. Il est alors nécessaire d'être authentifié pour en avoir l'accès si l'authentification est activé. L'authentification se fait grâce au module auth. Une fois authentifié, vous devez utiliser le token fournit par le module auth pour accéder au méthode privée.

Exemple d'appel d'une méthode privée avec curl

```
$ curl http://neufbox/api/1.0/?method=hotspot.getClientList\&token=43 ↩
    f6168e635b9a90774cc4d3212d5703c11c9302
```

2.2 Message de retour

- Lorsque l'appel de la méthode a réussi, l'attribut stat de la balise rsp vaut ok

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
        [resultat]
</rsp>
```

- Si l'appel de la méthode a échoué, l'attribut stat de la balise rsp vaut fail. La balise rsp contient alors une balise err avec un attribut code contenant le code d'erreur et un attribut msg contenant un message d'explication de l'erreur en anglais.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="fail" version="1.0">
        <err code="[code-erreur]" msg="[message-erreur]" />
</rsp>
```

2.2.1 Codes d'erreurs

Il existe deux types de codes d'erreurs :

- les codes d'erreurs génériques qui peuvent être renvoyés suite à n'importe quel appel
- les codes d'erreurs propres à la méthode appelée

code	msg	explication
0	Unknown error	Une erreur inconnue s'est produite
112	Method not found	Impossible de trouver la méthode demandée
113	Need argument(s)	Besoin de plus d'arguments
114	Invalid argument(s)	Arguments soumis invalides
115	Authentication needed	Authentification nécessaire
120	The box is being upgraded	La neufbox est en cours de mise à jour

Table 1: Code d'erreur générique

3 Sections

3.1 auth

Ce module doit être utilisé pour s'authentifier et ainsi pouvoir accéder aux méthodes privées de l'API REST.

Techniquement, ce module fournit un token qui sera valide une fois authentifié et qu'il faudra utilisé en paramètre à chaque future requête.

Exemple de requête avec l'utilisation du token

```
$ http://neufbox/api/1.0/?method=hotspot.getClientList\&token=43 ↩
    f6168e635b9a90774cc4d3212d5703c11c9302
```

Depuis le firmware 2.1.2, l'API REST et l'interface web de la neufbox utilise la même configuration pour l'authentification.

Il y a 4 configurations possible:

- désactivé,
- login/mot de passe,
- bouton de service,
- login/mot de passe et le bouton de service.

Warning
Avant le firmware 2.1.2, il était impossible de désactivé l'authentification et seul l'authentification par bouton de service était disponible.

Pour s'authentifier avec un login et un mot de passe, il faut procéder comme avec l'authentification par bouton de service sauf qu'il faut utiliser en plus le paramétre hash lors de l'appel de la méthode auth.checkToken. Ce paramétre hash est la concaténation du hash du login et du hash du mot de passe.

Un hash d'une valeur est composé de 64 caractères (32 digest SHA256 en représentation hexadécimal) et se calcul ainsi (value étant la valeur a hasher et key le token):

```
fh = sha256_hash(value)
hash = hmac_sha256_hash(key, fh)
```

Exemple de code de hashage en C

3.1.1 Example d'authentification avec un login et un mot de passe

Note
Exemple avec login valant *admin* et mot de passe valant *admin*.

```
$ curl -s -G  http://neufbox/api/1.0/?method=auth.getToken
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <auth token="43f6168e635b9a90774cc4d3212d5703c11c9302" method="passwd" />
</rsp>
```

```
$ ./hash 43f6168e635b9a90774cc4d3212d5703c11c9302 admin
hash = 7aa3e8b3ed7dfd7796800b4c4c67a0c56c5e4a66502155c17a7bcef5ae945ffa
```

```
$ curl -s http://neufbox/api/1.0/?method=auth.checkToken\&token=43 ↩
    f6168e635b9a90774cc4d3212d5703c11c9302\&hash=7 ↩
    aa3e8b3ed7dfd7796800b4c4c67a0c56c5e4a66502155c17a7bcef5ae945ffa
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <auth token="43f6168e635b9a90774cc4d3212d5703c11c9302" />
</rsp>
```

Warning
Dans la pratique, il faut que toutes ses commandes soit executés en moins de 5 secondes à cause du timeout de validité du token lors de l'utilisation de la méthode d'authentification par login/mot de passe seul.

3.1.2 auth.getToken

- Méthode HTTP : GET
- Accès : public
- Description : retourne un nouveau token pour la procédure d'authentification, ou un code d'erreur
- Retour :
  - Si succès :
    - balise **tag** > attribut **token**. Valeur du nouveau token.
    - balise **tag** > attribut **method** = `(passwd|button|all)`. Méthodes possibles pour s'authentifier. La valeur `all` signifie que toutes les méthodes d'authentification sont possibles. *(firmware >= 2.1.2)*
  - Si erreur :
    - balise **err** > attribut **code** contient le code d'erreur :
      - `0` : Unknown error. Erreur interne lors de la génération du token
      - `201` : Max token reached. Nombre maximal de tokens atteint (la limite est de 64 demandes simultanées)
      - `205` : Authentification disabled. L'authentification est désactivée. *(firmware >= 2.1.2)*

Exemple :

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <auth token="fe5be7az1v9cb45zeogger8b4re145g3" method="passwd" />
</rsp>
```

3.1.3 auth.checkToken

Note
le paramètre hash est obtenue en concaténant le hash du login et le hash du mot de passe (la longueur de cette valeur est donc de 128 caractères).

Warning
Si la méthode d'authentification autorisée est uniquement par login/mot de passe, le timeout entre le getToken et checkToken est de 5 secondes. *(firmware >= 2.1.2)*

- Méthode HTTP : GET
- Accès : public
- Description : valider un token grace à une méthode d'authentification.
- Paramètres requête :
  - **token** : token à valider *(obligatoire)*
  - **hash** : hash du login/mot de passe *(optionnel: si essai d'authentification par login/mot de passe) (firmware >= 2.1.2) (voir la note ci dessus pour la méthode du fabrication du hash)*
- Retour :
  - Si succès :
    - balise **tag** > attribut **token**. Valeur du token validé.
  - Si erreur :
    - balise **err** > attribut **code** contient le code d'erreur :
      - 201 : Invalid session. La session n'existe pas ou est déjà authentifiée.
      - 202 : Pre-Session timeout. Vous disposez de 5 minutes entre le getToken et le checkToken pour valider le token.
      - 203 : Push button not pushed. Le bouton n'a pas été appuyé.
      - 204 : Invalid login and/or password. Le login et/ou le mot de passe est invalide. *(firmware >= 2.1.2)*
      - 205 : Authentification disabled. L'authentification est désactivée. *(firmware >= 2.1.2)*

Exemple d'un succès puis d'une erreur :

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <auth token="fe5be7az1v9cb45zeogger8b4re145g3" />
</rsp>
```

```
<?xml version="1.0" ?>
<rsp stat="fail" version="1.0">
   <err code="203" msg="Push button not pushed" />
</rsp>
```

3.2 backup3g

3.2.1 backup3g.forceDataLink

Note
Existe depuis le firmware 2.1.2

- Méthode HTTP : POST
- Accès : privé
- Description : Cette méthode définie la politique d'utilisation de la 3g pour la data.
- Paramètre requête :
  - **mode** = `(on|off|auto)`
    - `on` : on force l'utilisation de la 3g
    - `off` : on interdit l'utilisation de la 3g
    - `auto` : bascule en 3g uniquement si l'adsl et/ou le ppp adsl est down *(politique par défaut sur la neufbox)*

3.2.2 backup3g.forceVoipLink

Note
Existe depuis le firmware 2.1.2

- Méthode HTTP : POST
- Accès : privé
- Description : Cette méthode définie la politique d'utilisation de la 3g pour la voix.
- Paramètre :
  - **mode** = `(on|off)`
    - `on` : on force l'utilisation de la 3g
    - `off` : on interdit l'utilisation de la 3g

3.2.3 backup3g.getPinCode

Note
Existe depuis le firmware 3.0.14

- Méthode HTTP : GET
- Accès : privé
- Description : Retourne le code pin de la clé 3g.
- Retour :
  - balise **pin** > attribut **code**. Code pin.

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <pin code="0000" />
</rsp>
```

3.2.4 backup3g.setPinCode

Note
Existe depuis le firmware 3.0.14

- Méthode HTTP : POST
- Accès : privé
- Description : Cette méthode définie le code pin de la clé 3g.
- Paramètre :
  - **pincode** = `([0-9]{4,8})`

3.3 ddns

3.3.1 ddns.disable

- Méthode HTTP : POST
- Accès : privé
- Description : Désactive le service

3.3.2 ddns.enable

- Méthode HTTP : POST
- Accès : privé
- Description : Active le service

3.3.3 ddns.forceUpdate

- Méthode HTTP : POST
- Accès : privé
- Description : Force la mise à jour du service

3.3.4 ddns.getInfo

- Méthode HTTP : GET
- Accès : privé
- Description : Renvoie des informations sur le serice de dns dynamique.
- Retour :
  - balise **ddns** > attribut **active**. = `(on|off)`. Activation du service.
  - balise **ddns** > attribut **service**. Nom du service
  - balise **ddns** > attribut **username**. Identifiant du service
  - balise **ddns** > attribut **password**. Mot de passe du service
  - balise **ddns** > attribut **hostname**. Nom d'hôte du serivce

- balise **ddns** > attribut **status**. = `(down|starting|up|updated|waiting_wan|err_update|err_gprs|error_server|error_unknown|error_account|error_account_loginpass|error_account_hostname|error_account_abuse)`.
- balise **ddns** > attribut **lastupdate**. Timestamp de la dernière mise à jour du service.
- balise **ddns** > attribut **lastupdateip**. Dernière ip du service.
- balise **ddns** > attribut **lastfreeze**. Timestamp du dernier gel du service (suite à une erreur du serveur). *(firmware >= 3.2.1)*
- balise **ddns** > attribut **lastfreezetime**. Nombre de secondes du gel. *(firmware >= 3.2.1)*

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <ddns active="on"
         service="dyndns"
         username="toto"
         password="1234"
         domain="subdomain.dyndns.org"
         status="updated"
         lastupdate="167"
         lastupdateip="109.0.249.184"
         lastfreeze=""
         lastfreezetime="" />
</rsp>
```

3.3.5 ddns.setService

- Méthode HTTP : POST
- Accès : privé
- Description : Configurer le compte ddns
- Paramètres :
  - **service** = `(dyndns|no-ip|ovh|dyndnsit|changeip|sitelutions)`
  - **username**
  - **password**
  - **hostname**

3.4 dsl

3.4.1 dsl.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : Renvoie des informations sur le lien ADSL.
- Retour :
  - balise **dsl** > attribut **linemode**. Mode du lien. *(firmware >= 2.1.2)*
  - balise **dsl** > attribut **uptime**. Nombre de seconde depuis la montée du lien. *(firmware >= 2.1.2)*
  - balise **dsl** > attribut **counter**. Nombre de connexion ADSL effectué. *(firmware >= 2.1.2)*
  - balise **dsl** > attribut **crc**. Nombre d'erreur CRC. *(firmware >= 2.1.2)*

- balise **dsl** > attribut **status** = `(up|down)`. Status du lien.
- balise **dsl** > attribut **noise_down**. Marge de bruit flux descendant.
- balise **dsl** > attribut **noise_up**. Marge de bruit flux montant.
- balise **dsl** > attribut **attenuation_down**. Atténuation flux descendant.
- balise **dsl** > attribut **attenuation_up**. Atténuation flux montant.
- balise **dsl** > attribut **rate_down**. Débit flux descendant.
- balise **dsl** > attribut **rate_up**. Débit flux montant.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <dsl linemode="G.DMT" uptime="4857" counter="1" crc="0"
        status="up" noise_down="4.5" noise_up="4.2"
        attenuation_down="3.2" attenuation_up="5.2" rate_down="8000" rate_up="800" />
</rsp>
```

3.5 firewall

3.5.1 firewall.enableSmtpFilter

- Méthode HTTP : POST
- Accès : privé
- Description : activer le filtrage du SMTP

3.5.2 firewall.disableSmtpFilter

- Méthode HTTP : POST
- Accès : privé
- Description : désactiver le filtrage du SMTP

3.5.3 firewall.getInfo

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur l'activation des différents filtrages
- Retour :
  - balise **firewall** > attribut **mode** = `(simple|)`
  - balise **firewall** > balise **winsharedrop** > attribut **active** = `(on|off)`
  - balise **firewall** > balise **icmpdrop** > attribut **active** = `(on|off)`
  - balise **firewall** > balise **smtpdrop** > attribut **active** = `(on|off)`

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <firewall mode="simple">
      <winsharedrop active="on" />
      <icmpdrop active="off" />
      <smtpdrop active="on" />
   </firewall>
</rsp>
```

3.6 hotspot

3.6.1 hotspot.enable

- Méthode HTTP : POST
- Accès : privé
- Description : activer le hotspot.

3.6.2 hotspot.disable

- Méthode HTTP : POST
- Accès : privé
- Description : désactiver le hotspot.

3.6.3 hotspot.getClientList

- Méthode HTTP : GET
- Accès : privé
- Description : liste des clients hotspot.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <client mac_addr="00:00:00:00:00:00" ip_addr="192.168.2.1" />
   <client mac_addr="11:11:11:11:11:11" ip_addr="192.168.2.2" />
</rsp>
```

3.6.4 hotspot.getInfo

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur le service hotspot.
- Retour :
  - balise **hotspot** > attribut **status** = `(up|down)`
  - balise **hotspot** > attribut **enabled** = `(on|off)`

– balise **hotspot** > attribut **mode** = `(sfr|sfr_fon)` (anciennes valeurs avant firmware 3.1: `(twin_neuf|twin_neuf_fon)`)

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <hotspot status="up" enabled="on" mode="sfr" />
</rsp>
```

3.6.5 hotspot.restart

- Méthode HTTP : POST
- Accès : privé
- Description : redémarrer le service hotspot.

3.6.6 hotspot.setMode

- Méthode HTTP : POST
- Accès : privé
- Description : définir le mode hotspot.
- Paramètre requête :
  – **mode** = `(sfr|sfr_fon)`

3.6.7 hotspot.start

- Méthode HTTP : POST
- Accès : privé
- Description : démarrer le service hotspot (pour que le hotspot soit démarré, il faut qu'il soit activé).

3.6.8 hotspot.stop

- Méthode HTTP : POST
- Accès : privé
- Description : arrêter le service hotspot.

3.7 lan

3.7.1 lan.addDnsHost

- Méthode HTTP : POST
- Accès : privé
- Description : Ajoute une entrée DNS sur le réseau local.
- Paramètre requête :

  - **ip**
  - **name**
- Retour :
  - balise **lan** > attribut **ip**.
  - balise **lan** > attribut **name**.
- Retour :
  - Si erreur :
    - balise **err** > attribut **code** contient le code d'erreur :
      - 210 : DNS host already exist.
      - 211 : Hostname already used.

3.7.2 lan.deleteDnsHost

- Méthode HTTP : POST
- Accès : privé
- Description : Supprime une entrée DNS sur le réseau local.
- Paramètre requête :
  - **ip**
  - **name**
- Retour :
  - balise **lan** > attribut **ip**.
  - balise **lan** > attribut **name**.
- Retour :
  - Si erreur :
    - balise **err** > attribut **code** contient le code d'erreur :
      - 212 : Dns IP/Host not found

3.7.3 lan.getDnsHostList

- Méthode HTTP : GET
- Accès : public
- Description : Liste des entrées DNS sur le réseau local.
- Retour :
  - balise **lan** > attribut **ip**.
  - balise **lan** > attribut **name**.

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <dnshost ip="192.168.1.10" name="monpc.at.home" />
</rsp>
```

3.7.4 lan.getHostsList

Note
Existe depuis le firmware 3.0.7

- Méthode HTTP : GET
- Accès : public
- Description : liste des équipements du réseau local.
- Retour :
  - balise **host** > attribut **name**. Son nom.
  - balise **host** > attribut **ip**. Son adresse IP.
  - balise **host** > attribut **mac**. Son adresse MAC.
  - balise **host** > attribut **iface** = `(lan1|lan2|lan3|lan3|wlan0)`. Port sur lequel il est connecté.
  - balise **host** > attribut **probe** = Date de découverte (uptime neufbox) *(firmware >= 3.2.0)*
  - balise **host** > attribut **alive** = Date de dernière activité (uptime neufbox) *(firmware >= 3.2.0)*
  - balise **host** > attribut **type** = +(pc|stb|femto|plc|...) Type d'équipement *(firmware >= 3.2.0)*
  - balise **host** > attribut **status** = +(online|offline) Son état courant *(firmware >= 3.2.0)*

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <host type="pc" name="thinkpad" ip="192.168.1.98" mac="F0:DE:F1:62:20:FA" iface=" ↩
      lan1" probe="51" alive="51" status="online"/>
    <host type="pc" name="winbox" ip="192.168.1.64" mac="00:25:9c:90:e8:ac" iface=" ↩
       wlan0" probe="14014" alive="18295" status="online"/>
</rsp>
```

3.7.5 lan.getInfo

Note
Cette méthode était privée avant le firmware 2.1.2

- Méthode HTTP : GET
- Accès : public *(privé avant le firmware 2.1.2)*
- Description : informations sur le réseau local.
- Retour :
  - balise **lan** > attribut **ip_addr**. Adresse IP de la box.
  - balise **lan** > attribut **netmask**. Masque réseau de la box.
  - balise **lan** > attribut **dhcp_active** = `(on|off)`. Activation du service DHCP. *(firmware >= 3.0.7)*
  - balise **lan** > attribut **dhcp_start**. Adresse IP du début de la plage des IP attribuée par DHCP. *(firmware >= 3.0.7)*
  - balise **lan** > attribut **dhcp_end**. Adresse IP de fin de la plage des IP attribuée par DHCP. *(firmware >= 3.0.7)*
  - balise **lan** > attribut **dhcp_lease**. Nombre de seconde d'attribution de l'adresse IP par DHCP. *(firmware >= 3.0.7)*

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok">
   <lan ip_addr="192.168.1.1" netmask="255.255.255.0" dhcp_active="on" dhcp_start="
      192.168.1.20" dhcp_end="192.168.1.100" dhcp_lease="86400" />
</rsp>
```

3.8 ont

3.8.1 ont.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : informations sur le boitier ONT.
- Retour :
  - balise **ont** > attribut **type** = version matériel de l'ONT.
  - balise **ont** > attribut **version** = version du firmware de l'ONT.
  - balise **ont** > **info** > attribut **name**.
  - balise **ont** > **info** > attribut **value**.

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <ont type="I-010G-Q" version="3FE53583AOXA18">
          <info name="ranging" value="OPERATION (O5)" />
          <info name="uptime" value="7 day 6 hours 10 minutes 7 seconds..." />
          <info name="receive" value="-24.190000" />
          <info name="transmit" value="0.810000" />
          <info name="lanlos" value="INACTIVE" />
          <info name="lossframe" value="INACTIVE" />
          <info name="lossgem" value="INACTIVE" />
          <info name="losssignal" value="INACTIVE" />
          <info name="errmsg" value="" />
          <info name="omcc" value="INACTIVE" />
          <info name="deactivate" value="INACTIVE" />
          <info name="ranged" value="ACTIVE" />
          <info name="phyerror" value="INACTIVE" />
          <info name="rdi" value="INACTIVE" />
          <info name="failed" value="INACTIVE"  />
  </ont>
</rsp>
```

3.9 p910nd

3.9.1 p910nd.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : informations sur le service p910nd (partage réseau d'une imprimante). http://p910nd.sourceforge.net/.

- Retour :
  - balise **p910nd** > attribut **status** = `(up|down)`. Status du service.
  - balise **p910nd** > attribut **bidir** = `(on|off)`. Activation du mode bidirectionnel.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <p910nd status="up" bidir="on" />
</rsp>
```

3.10 ppp

3.10.1 ppp.getCredentials

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur le login et le mot de passe ppp.
- Retour :
  - balise **ppp** > attribut **login**. Login ppp.
  - balise **ppp** > attribut **password**. Mot de passe ppp.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <ppp login="0123456789@neufpnp" password="neufpnp" />
</rsp>
```

3.10.2 ppp.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : informations sur le lien ppp.
- Retour :
  - balise **ppp** > attribut **status** = `(up|down)`. Status du lien.
  - balise **ppp** > attribut **ip_addr**. Adresse IP du lien.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <ppp status="up" ip_addr="84.124.83.43" />
</rsp>
```

3.10.3 ppp.setCredentials

- Méthode HTTP : POST
- Accès : privé
- Description : définir le login et le mot de passe ppp.
- Paramètre requête :
  - **login**
  - **password**

3.11 smb

3.11.1 smb.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : informations sur le service SMB (partage de fichier windows)
- Retour :
  - balise **smb** > attribut **active** = `(on|off)`. Activation.
  - balise **smb** > attribut **status** = `(up|down|starting|installing|error_unknown)`. Status du service.
  - balise **smb** > attribut **name**. Nom du service.
  - balise **smb** > attribut **workgroup**. Workgroup.
  - balise **smb** > **share** > attribut **name**. Nom du partage.
  - balise **smb** > **share** > attribut **uuid**. UUID de la partition contenant le partage.
  - balise **smb** > **share** > attribut **dir**. Répertoire du partage.
  - balise **smb** > **share** > attribut **online** = `(true|false)`. Le partage est disponible (la partition est accessible, ie. la clé USB est branchée).

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
  <smb active="on" status="up" name="NeufBox" workgroup="Workgroup">
    <share name="photos" uuid="5C61-1D8D" dir="/photos" online="true"/>
  </smb>
</rsp>
```

3.12 system

3.12.1 system.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : informations système.
- Retour :

- balise **system** > attribut **product_id**. L'ID du produit: $(NB)-$(HARD)-$(HARD_VERSION).
- balise **system** > attribut **mac_addr**. L'adresse MAC de la neufbox.
- balise **system** > attribut **net_mode** = `(router|bridge)`.
- balise **system** > attribut **net_infra** = `(adsl|ftth|gprs)`. Connexion internet principale de la box.
- balise **system** > attribut **uptime**. Temps d'activité de la box en seconde.
- balise **system** > attribut **version_mainfirmware**. Version du firmware de la box: $(NB)-MAIN-R$(VERSION).
- balise **system** > attribut **version_rescuefirmware**.
- balise **system** > attribut **version_bootloader**.
- balise **system** > attribut **version_dsldriver**. *(indisponible sur NB5) (firmware >= 2.1.2)*
- balise **system** > attribut **current_datetime**. Date actuelle sous le format : "%Y%m%d%H%M". *(firmware >= 3.2.0)*
- balise **system** > attribut **refclient**. Référence client. *(firmware >= 3.2.0)*

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <system product_id="NB5-SER-r1"
           mac_addr="00:17:33:80:02:4a"
           net_mode="router"
           net_infra="adsl"
           uptime="76913"
           version_mainfirmware="NB5-MAIN-R2.1.2"
           version_rescuefirmware="NB5-RESCUE-R1.0.3"
           version_bootloader="NB5-BOOTLOADER-R05"
           version_dsldriver="NB4-A2pB024k2"
           current_datetime="201109210941"
           refclient="" />
</rsp>
```

3.12.2 system.getWpaKey

- Méthode HTTP : GET
- Accès : privé
- Description : clé WPA par défaut de la box.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <system wpa_key="thazcynshag4knahadza" />
</rsp>
```

3.12.3 system.reboot

Note
Existe depuis le firmware 3.0.7

- Méthode HTTP : POST
- Accès : privé
- Description : Redémarrer la box.

3.12.4 system.setNetMode

- Méthode HTTP : POST
- Accès : privé
- Description : définir le mode réseau de la box.
- Paramètre requête :
  - **mode** = `(router|bridge)`

3.12.5 system.setRefClient

- Méthode HTTP : POST
- Accès : privé
- Description : Définit la référence client.
- Paramètre requête :
  - **refclient**

3.13 voip

3.13.1 voip.getCallhistoryList

Note
Existe depuis le firmware 3.0.7

- Méthode HTTP : GET
- Accès : privé
- Description : historique des appels téléphonique.
- Retour :
  - balise **calls** > balise **call** > attribut **type** = `(pstn|voip|radio)`. Type de lien utilisé.
  - balise **calls** > balise **call** > attribut **direction** = `(incoming|outgoing)`. Sens de l'appel.
  - balise **calls** > balise **call** > attribut **number**. Numéro de téléphone.
  - balise **calls** > balise **call** > attribut **length**. Temps en seconde de l'appel.
  - balise **calls** > balise **call** > attribut **date**. Date en format UNIX de l'appel.

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
    <calls>
        <call type="voip" direction="incoming" number="065042 XXXX" length="125" date=" ↩
            1281111795" />
        <call type="voip" direction="incoming" number="044512 XXXX" length="31" date=" ↩
            1281111845" />
    </calls>
</rsp>
```

3.13.2 voip.getInfo

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur la voix sur IP.
- Retour :
  - balise **voip** > attribut **status** = `(up|down)`. Status du service VOIP.
  - balise **voip** > attribut **infra** = `(adsl|ftth|gprs)`. Lien utilisé pour la VOIP. *(firmware >= 2.1.2)*
  - balise **voip** > attribut **hook_status** = `(onhook|offhook|unknown)`. Status du combiné (onhook = raccroché). *(firmware >= 3.0.7)*
  - balise **voip** > attribut **callhistory_active** = `(on|off)`. Activation de l'historique des appels. *(firmware >= 3.0.7)*

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <voip status="up" infra="adsl" hook_status="onhook" callhistory_active="on" />
</rsp>
```

3.13.3 voip.restart

- Méthode HTTP : POST
- Accès : privé
- Description : redémarrer la voip.

3.13.4 voip.start

- Méthode HTTP : POST
- Accès : privé
- Description : démarrer la voip.

3.13.5 voip.stop

- Méthode HTTP : POST
- Accès : privé
- Description : arrêter la voip.

3.14 wan

3.14.1 wan.getInfo

Note
Cette méthode était privée avant le firmware 3.0.6

- Méthode HTTP : GET
- Accès : public *(privé avant le firmware 3.0.6)*
- Description : informations génériques sur la connexion internet.
- Retour :
  - balise **wan** > attribut **status** = `(up|down)`. Status de la connexion internet.
  - balise **wan** > attribut **uptime**. Temps de connexion internet.
  - balise **wan** > attribut **ip_addr**. Adresse IP internet.
  - balise **wan** > attribut **infra** = `(adsl|ftth|gprs)`. Lien utilisé pour la connexion internet. *(firmware >= 2.1.2)*

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
   <wan status="up" uptime="29873" ip_addr="10.23.40.20" infra="adsl" />
</rsp>
```

3.15 wlan

Il y a différents modes radio sur les neufbox. Chaque neufbox supporte une liste de modes radio:

- La NB6 et la NB5 supportent les modes 11n, 11b/g/n et 11b/g ;
- La NB4 et la CiBox supportent les modes 11b, 11b/g et auto.

Ci dessous, vous trouverez la correspondance entre les intitulés des modes radio et les valeurs utilisées en interne dans le système et dans l'API REST.

mode radio	**valeur en interne et dans l'API REST**
auto	auto
11b	11b
11b/g	11g
11b/g/n	11ng
11n	11n

Note
A partir des versions 2.1 du firmware, la représentation du mode wifi a changé pour plus de clarté et de simplicité. Voici la table de correspondance entre les anciennes et les nouvelles valeurs :

ancienne valeur	**nouvelle valeur**
0	11b
1	auto
2	11g
11n-only	11n
auto	11ng
11n-legacy	11ng
legacy	11g

3.15.1 wlan.enable

- Méthode HTTP : POST
- Accès : privé
- Description : activer le WiFi.

3.15.2 wlan.disable

- Méthode HTTP : POST
- Accès : privé
- Description : désactiver le WiFi.

3.15.3 wlan.getClientList

- Méthode HTTP : GET
- Accès : privé
- Description : liste des clients WiFi.

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
  <client mac_addr="01:02:03:04:05:06" ip_addr="192.168.1.23" />
  <client mac_addr="06:07:08:09:10:11" ip_addr="192.168.1.24" />
</rsp>
```

3.15.4 wlan.getInfo

Warning
Les modes wifi ont changé à partir de la 2.1: table de correspondance.

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur le WiFi.
- Retour :
  - balise **wlan** > attribut **active** = `(on|off)`. Activation.
  - balise **wlan** > attribut **channel**. Canal.
  - balise **wlan** > attribut **mode** = `(auto|11b|11g|11n|11ng)`. Mode radio.
  - balise **wlan** > attribut **mac_filtering** = `(whitelist|blacklist|off)`. Activation du filtrage mac. *(firmware >= 3.0.7)*
  - balise **wlan** > balise **wl0** > attribut **ssid**. Nom du réseau.
  - balise **wlan** > balise **wl0** > attribut **enc** = `(OPEN|WEP|WPA-PSK|WPA2-PSK|WPA-WPA2-PSK)`. Encryption. *(Nouveaux modes à partir du firmware 2.1)*

- balise **wlan** > balise **wl0** > attribut **enctype** = `(tkip|aes|tkipaes)`. *(firmware >= 3.2.0)*
- balise **wlan** > balise **wl0** > attribut **keytype** = `(ascii|hexa)`.
- balise **wlan** > balise **wl0** > attribut **wpakey**. Clé WPA.
- balise **wlan** > balise **wl0** > attribut **wepkey**. Clé WEP primaire.

Exemple

```
<?xml version="1.0" ?>
 <rsp stat="ok" version="1.0">
   <wlan active="on" channel="11" mode="11ng" mac_filtering="off">
     <wl0 ssid="NEUF_0060" enc="WPA-PSK" keytype="ascii" wpakey="thazcynshag4knahadza" ↩
          wepkey="" />
   </wlan>
 </rsp>
```

3.15.5 wlan.setChannel

- Méthode HTTP : POST
- Accès : privé
- Description : définir le canal WiFi.
- Paramètre requête :
  - **channel** (entre 1 et 13)

3.15.6 wlan.setWl0Enc

- Méthode HTTP : POST
- Accès : privé
- Description : définir la sécurité du réseau WiFi.
- Paramètre requête :
  - **enc** = `(OPEN|WEP|WPA-PSK|WPA2-PSK|WPA-WPA2-PSK)`

3.15.7 wlan.setWl0Enctype

Note
Existe depuis le firmware 3.2.0

- Méthode HTTP : POST
- Accès : privé
- Description : définir le type de clé WPA.
- Paramètre requête :
  - **enctype** = `(tkip|aes|tkipaes)`

3.15.8 wlan.setWl0Keytype

- Méthode HTTP : POST
- Accès : privé
- Description : définir le type de clé WEP.
- Paramètre requête :
  - **keytype** = `(ascii|hexa)`

3.15.9 wlan.setWl0Ssid

- Méthode HTTP : POST
- Accès : privé
- Description : définir le nom du réseau WiFi.
- Paramètre requête :
  - **ssid**

3.15.10 wlan.setWl0Wepkey

- Méthode HTTP : POST
- Accès : privé
- Description : définir la clé WEP.
- Paramètre requête :
  - **wepkey**

3.15.11 wlan.setWl0Wpakey

- Méthode HTTP : POST
- Accès : privé
- Description : définir la clé WPA.
- Paramètre requête :
  - **wpakey**

3.15.12 wlan.setWlanMode

Warning
Les modes wifi ont changé à partir de la 2.1: table de correspondance.

- Méthode HTTP : POST
- Accès : privé

- Description : définir le mode radio WiFi.
- Paramètre requête :
  - *(Pour NB5/NB6)* **mode** = `(11n|11ng|11g)`
  - *(Pour NB4/CIBOX)* **mode** = `(11b|11g|auto)`

3.15.13 wlan.start

- Méthode HTTP : POST
- Accès : privé
- Description : démarrer le WiFi (pour que le WiFi soit démarré, il faut qu'il soit activé).

3.15.14 wlan.stop

- Méthode HTTP : POST
- Accès : privé
- Description : arrêter le WiFi.

3.15.15 wlan.restart

- Méthode HTTP : POST
- Accès : privé
- Description : redémarrer le WiFi.

4 Annexe

4.1 Code de hashage en C

hash.c

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <etk/crypt.h>

int crypto_hmac_sha256_hash(char *key, char *msg, char **result)
{
        unsigned char digest[32];
        int i;

        if(msg == NULL)
        {
                return -1;
        }

        *result = calloc(1, 32 * 2 + 1);

        if (*result == NULL)
        {
                return -1;
```

```
        }

        etk_sha256_hmac((uint8_t*) msg, strlen(msg), (uint8_t*) key,
                        strlen(key), digest);

        for (i = 0; i < 32; i++)
        {
                snprintf((*result) + 2 * i, 3,
                         "%02x", (unsigned char) digest[i]);
        }

        return 0;
}

int crypto_sha256_hash(char *msg, char **result)
{
        unsigned char digest[32];
        int i;

        if (msg == NULL)
        {
                return -1;
        }

        *result = calloc(1, 32 * 2 + 1);

        if (*result == NULL)
        {
                return -1;
        }

        etk_sha256((uint8_t*) msg, strlen(msg), digest);

        for (i = 0; i < 32; i++)
        {
                snprintf((*result) + 2 * i, 3,
                         "%02x", (unsigned char) digest[i]);
        }

        return 0;
}

int main(int argc, char **argv)
{
        char* value_prehash = NULL, *value_hashed = NULL;
        int ret = 0;

        if(argc != 3)
        {
                fprintf(stderr,
                        "Usage: %s <token> <value to hash>\n", argv[0]);
                return 1;
        }

        if(crypto_sha256_hash(argv[2], &value_prehash) != 0)
        {
                fprintf(stderr, "crypto_sha256_hash failed !\n");
                ret = 1;
                goto clean;
        }

        if(crypto_hmac_sha256_hash(argv[1],
```

```
                                value_prehash,
                                &value_hashed) != 0)
        {
                fprintf(stderr, "crypto_hmac_sha256_hash failed !\n");
                ret = 1;
                goto clean;
        }

        printf("hash = %s\n", value_hashed);

clean:
        free(value_prehash);
        free(value_hashed);

        return ret;
}
```

Compilation

```
$ gcc -lm -lz -ltropicssl hash.c -o hash
$ ./hash
Usage: ./hash <token> <value to hash>
```

5 Crédits

5.1 Remerciements

Merci à la communauté pour toutes les suggestions de nouvelles fonctionnalités, pour les remontées de problèmes, etc. Merci en particulier à:

- VincentAlex
- Gandalf
- SGDA