API REST

COLLABORATORS			
	TITLE : API REST		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	http://dev.efixo.net	11 oct. 2011	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
20111011	11 oct. 2011		H

Contents

List of Tables

1 ChangeLog

1.1 Firmware 3.1

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

1.2 Firmware 3.0.14

- Nouvelle méthode app.getInfo Section 3.1.1
- Nouvelle méthode backup3g.getPinCode Section 3.3.1
- Nouvelle méthode backup3g.setPinCode Section 3.3.4

1.3 Firmware 3.0.7

- Nouvelle méthode voip.getCallhistoryList Section 3.10.1
- Nouvelle méthode lan.getHostsList Section 3.7.1
- Nouvelle méthode system.reboot Section 3.9.3
- Méthode lan.getInfo Section 3.7.2:
  - ajout des attributs **dhcp_active**, **dhcp_start**, **dhcp_end** et **dhcp_lease**.
- Méthode voip.getInfo Section 3.10.2:
  - ajout des attributs **hook_status** et **callhistory_active**.
- Méthode wlan.getInfo Section 3.12.4:
  - ajout de l'attribut **mac_filtering**.

1.4 Firmware 3.0.6

- Méthode wan.getInfo Section 3.11.1:
  - la méthode **wan.getInfo** est dorénavent public.

1.5 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 3.3.
- Méthode wan.getInfo Section 3.11.1:
  - ajout de l'attribut **infra**.
- Méthode voip.getInfo Section 3.10.2:
  - ajout de l'attribut **infra**.

- Méthode lan.getInfo Section 3.7.2:
  - 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 3.9.1:
  - ajout de l'attribut **version_dsldriver**.
  - ajout de l'attribut **net_infra**.
- Correction d'erreurs diverses:
  - La méthode ppp.setCredentials Section 3.8.3 est corrigée.

1.6 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.
- 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. L'authentification se fait grâce au module auth.

Exemple d'appel d'une méthode avec curl

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

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 Code d'erreur

Il existe 2 types de code d'erreur:

- 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	Need authentication	Authentification nécessaire
120	The box being upgrade	La neufbox est en cours de mise à jour

Table 1: Code d'erreur générique

3 Sections

3.1 app

3.1.1 app.getInfo

Note
Existe depuis le firmware 3.0.14

- Méthode HTTP : GET
- Accès : public
- Description : informations sur les applications tournant sur la box.
- Retour :
  - balise **p910nd** > attribut **status** = `(up|down)`. Status du partage de l'imprimante.
  - balise **p910nd** > attribut **bidir** = `(on|off)`. Activation du mode bidirectionnel.
  - balise **smbd** > attribut **active** = `(on|off)`. Activation du partage de fichiers.
  - balise **smbd** > attribut **state** = `(up|down)`. Status du partage de fichiers.

– balise **smbd** > attribut **codeno** = `(starting|error_internal)`. Information complémentaire sur le status du partage de fichiers.

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <p910nd status="down" bidir="on" />
   <smbd active="off" state="down" codeno="" />
</rsp>
```

3.2 auth

Depuis le firmware 2.1.2, l'authentification sur l'API REST se base sur celle employée sur l'interface web de configuration de la neufbox.

Vous avez donc le choix entre:

- désactiver l'authentification,
- le login/mot de passe,
- le bouton de service,
- activer toutes les méthodes d'authentification.

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 Section 4.1

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

3.2.1 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.

3.2.2 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>
```

3.2.3 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 : vérifier si le token a été validé
- 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>
```

3.3 backup3g

3.3.1 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>
```

3.3.2 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)*

3.3.3 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

3.3.4 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})`

3.4 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.getHostsList

Note
Existe depuis le firmware 3.0.7

- Méthode HTTP : GET
- Accès : public
- Description : liste des clients du réseau local.
- Retour :
  - balise **host** > attribut **name**. Nom d'hôte de l'ordinateur.
  - balise **host** > attribut **ip**. Adresse IP.
  - balise **host** > attribut **mac**. Adresse MAC.
  - balise **host** > attribut **iface** = `(lan1|lan2|lan3|lan3|wlan0)`. Nom du port sur lequel l'ordinateur est branché.

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
   <host name="bigbox" ip="192.168.1.98" mac="00:21:91:12:d4:7c" iface="lan4" />
   <host name="laptop" ip="192.168.1.99" mac="00:15:d5:42:d4:7f" iface="lan3" />
</rsp>
```

3.7.2 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 **ppp** > attribut **ip_addr**. Adresse IP de la box.
  - balise **ppp** > attribut **netmask**. Masque réseau de la box.
  - balise **ppp** > attribut **dhcp_active**. Activation du service DHCP. *(firmware >= 3.0.7)*
  - balise **ppp** > attribut **dhcp_start**. Adresse IP du début de la plage des IP attribuée par DHCP. *(firmware >= 3.0.7)*
  - balise **ppp** > attribut **dhcp_end**. Adresse IP de fin de la plage des IP attribuée par DHCP. *(firmware >= 3.0.7)*
  - balise **ppp** > 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 ppp

3.8.1 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>
```

3.8.2 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>
```

3.8.3 ppp.setCredentials

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

3.9 system

3.9.1 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)*

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" />
</rsp>
```

3.9.2 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>
```

3.9.3 system.reboot

Note
Existe depuis le firmware 3.0.7

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

3.9.4 system.setNetMode

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

3.10 voip

3.10.1 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>
```

3.10.2 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>
```

3.10.3 voip.restart

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

3.10.4 voip.start

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

3.10.5 voip.stop

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

3.11 wan

3.11.1 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>
```

3.12 wlan

Note
A partir des versions 2.1 du firmware, la répresentation du mode wifi a changé pour plus de clareté 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

3.12.1 wlan.enable

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

3.12.2 wlan.disable

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

3.12.3 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>
```

3.12.4 wlan.getInfo

Warning
Les modes wifi ont changé à partir de la 2.1: table de correspondance [?note].

- 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 **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>
```

3.12.5 wlan.setChannel

- Méthode HTTP : POST
- Accès : privé
- Description : définir le canal WiFi.
- Paramètre requête :
  - **channel**. 1 ⇐ channel ⇐ 13.

3.12.6 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)`

3.12.7 wlan.setWl0Keytype

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

3.12.8 wlan.setWl0Ssid

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

3.12.9 wlan.setWl0Wepkey

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

3.12.10 wlan.setWl0Wpakey

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

3.12.11 wlan.setWlanMode

Warning
Les modes wifi ont changé à partir de la 2.1: table de correspondance [?note].

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

3.12.12 wlan.restart

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

3.12.13 wlan.start

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

3.12.14 wlan.stop

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

4 Annexe

4.1 Code de hashage en C

hash.c

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <tropicssl/sha2.h>

int crypto_hmac_sha256_hash(char *key, char *msg, char **result)
{
        sha2_context hmac;
        unsigned char digest[32];
        int i;

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

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

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

        sha2_hmac_starts(&hmac, key, strlen(key), 0);
        sha2_hmac_update(&hmac, (unsigned char*) msg, strlen(msg));
        sha2_hmac_finish(&hmac, 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)
{
        sha2_context sha256;
        unsigned char digest[32];
        int i;

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

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

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

        sha2_starts(&sha256, 0);
        sha2_update(&sha256, (unsigned char*) msg, strlen(msg));
        sha2_finish(&sha256, 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>
```