

API REST

COLLABORATORS

	<i>TITLE :</i> API REST		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	http://dev.efixo.net	03 Aug 2016	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
20160803	03 Aug 2016		H

Contents

1	ChangeLog	1
1.1	Firmware 4.0.18	1
1.2	Firmware 4.0	1
1.3	Firmware 3.5	1
1.4	Firmware 3.4	2
1.5	Firmware 3.3.2	2
1.6	Firmware 3.2.3	2
1.7	Firmware 3.2.1	2
1.8	Firmware 3.2.0	3
1.9	Firmware 3.1.0	3
1.10	Firmware 3.0.14	3
1.11	Firmware 3.0.7	3
1.12	Firmware 3.0.6	4
1.13	Firmware 2.1.2	4
1.14	Firmware 2.1.1	4
2	Introduction	5
2.1	Utilisation	5
2.2	Message de retour	5
2.2.1	Codes d'erreurs	5
3	Sections	6
3.1	auth	6
3.1.1	Example d'authentification avec un login et un mot de passe	7
3.1.2	auth.getToken	7
3.1.3	auth.checkToken	8
3.2	backup3g	9
3.2.1	backup3g.forceDataLink	9
3.2.2	backup3g.forceVoipLink	9
3.2.3	backup3g.getPinCode	9
3.2.4	backup3g.setPinCode	10
3.3	ddns	10
3.3.1	ddns.disable	10
3.3.2	ddns.enable	10
3.3.3	ddns.forceUpdate	10
3.3.4	ddns.getInfo	10
3.3.5	ddns.setService	11

3.4	dsl	11
3.4.1	dsl.getInfo	11
3.5	ftth	12
3.5.1	ftth.getInfo	12
3.6	firewall	12
3.6.1	firewall.enableSmtpFilter	12
3.6.2	firewall.disableSmtpFilter	13
3.6.3	firewall.getInfo	13
3.7	hotspot	13
3.7.1	hotspot.enable	13
3.7.2	hotspot.disable	13
3.7.3	hotspot.getClientList	14
3.7.4	hotspot.getInfo	14
3.7.5	hotspot.setMode	14
3.7.6	hotspot.restart	14
3.7.7	hotspot.start	15
3.7.8	hotspot.stop	15
3.8	lan	15
3.8.1	lan.addDnsHost	15
3.8.2	lan.deleteDnsHost	15
3.8.3	lan.getDnsHostList	16
3.8.4	lan.getHostsList	16
3.8.5	lan.getInfo	17
3.9	ont	17
3.9.1	ont.getInfo	17
3.9.2	ont.sync	19
3.9.3	ont.push	19
3.9.4	ont.pull	20
3.10	p910nd	20
3.10.1	p910nd.getInfo	20
3.11	ppp	21
3.11.1	ppp.getCredentials	21
3.11.2	ppp.getInfo	21
3.11.3	ppp.setCredentials	21
3.12	smb	22
3.12.1	smb.getInfo	22
3.13	system	22
3.13.1	system.getInfo	22
3.13.2	system.getIfList	23

3.13.3	system.getWpaKey	24
3.13.4	system.reboot	24
3.13.5	system.setNetMode	25
3.13.6	system.setRefClient	25
3.14	tv	25
3.14.1	tv.getInfo	25
3.15	usb	26
3.15.1	usb.getInfo	26
3.16	voip	26
3.16.1	voip.getCallhistoryList	26
3.16.2	voip.getInfo	27
3.16.3	voip.restart	27
3.16.4	voip.start	27
3.16.5	voip.stop	27
3.17	wan	28
3.17.1	wan.getInfo	28
3.18	wlan	28
3.18.1	wlan.enable	29
3.18.2	wlan.disable	29
3.18.3	wlan.getClientList	29
3.18.4	wlan.getInfo	29
3.18.5	wlan.getScanList	30
3.18.6	wlan.setChannel	31
3.18.7	wlan.setW10Enc	31
3.18.8	wlan.setW10Encype	32
3.18.9	wlan.setW10Keytype	32
3.18.10	wlan.setW10Ssid	32
3.18.11	wlan.setW10Wepkey	32
3.18.12	wlan.setW10Wpakey	32
3.18.13	wlan.setWlanMode	33
3.18.14	wlan.start	33
3.18.15	wlan.stop	33
3.18.16	wlan.restart	33
3.19	wlan5	33
3.19.1	wlan5.getClientList	33
3.19.2	wlan5.getInfo	34
3.19.3	wlan5.setChannel	34
3.19.4	wlan5.setW10Enc	34
3.19.5	wlan5.setW10Encype	35

3.19.6	wlan5.setWl0Keytype	35
3.19.7	wlan5.setWl0Ssid	35
3.19.8	wlan5.setWl0Wepkey	35
3.19.9	wlan5.setWl0Wpakey	35
3.19.10	wlan5.setWlanMode	36
3.20	guest	36
3.20.1	guest.getInfo	36
3.20.2	guest.enable	36
3.20.3	guest.disable	37
3.20.4	guest.setSsid	37
3.20.5	guest.setWpakey	37
4	Annexe	38
4.1	Code de hashage en C	38
5	Crédits	39
5.1	Remerciements	39

List of Tables

1	Code d'erreur générique	6
---	-----------------------------------	---

1 ChangeLog

1.1 Firmware 4.0.18

- Méthode `wlan5.getInfo` Section 3.19.2
- Méthode `wlan5.getClientList` Section 3.19.1
- Méthode `wlan5.setChannel` Section 3.19.3
- Méthode `wlan5.setWlanMode` Section 3.19.10
- Méthode `wlan5.setWl0Ssid` Section 3.19.7
- Méthode `wlan5.setWl0Enc` Section 3.19.4
- Méthode `wlan5.setWl0EncType` Section 3.19.5
- Méthode `wlan5.setWl0KeyType` Section 3.19.6
- Méthode `wlan5.setWl0WpaKey` Section 3.19.9
- Méthode `wlan5.setWl0WepKey` Section 3.19.8
- Méthode `guest.getInfo` Section 3.20.1
- Méthode `guest.enable` Section 3.20.2
- Méthode `guest.disable` Section 3.20.3
- Méthode `guest.setSsid` Section 3.20.4
- Méthode `guest.setWpaKey` Section 3.20.5

1.2 Firmware 4.0

- Méthode `usb.getInfo` Section 3.15.1
- Méthode `wlan.getScanList` Section 3.18.5
- Méthode `system.getIfList` Section 3.13.2
- Méthode `system.getInfo` Section 3.13.1
 - ajout de l'attribut `serialnumber`

1.3 Firmware 3.5

- Méthode `tv.getInfo` Section 3.14.1
 - ajout de l'attribut `netchain`
- Méthode `system.getInfo` Section 3.13.1
 - ajout de l'attribut `alimvoltage`
 - ajout de l'attribut `temperature`
- Méthode `ftth.getInfo` Section 3.5.1
 - ajout de l'attribut `wanfibre`

1.4 Firmware 3.4

- Méthode `tv.getInfo` Section 3.14.1
- Méthode `system.getInfo` Section 3.13.1
 - ajout des attributs **idur**.
- Méthode `ont.sync` Section 3.9.2
 - la méthode **ont.sync** est dorénavent publique.
- Méthode `wan.getInfo` Section 3.17.1
 - ajout des attributs **infra6**, **status6**, **uptime6**, **ipv6_addr**
- Nouvelle méthode `tv.getInfo` Section 3.14.1
- Méthode `hotspot.getInfo` Section 3.7.4
 - l'attribut **mode** n'existe plus
- Suppression de la méthode `hotspot.setMode` Section 3.7.5

1.5 Firmware 3.3.2

- Méthode `wan.getInfo` Section 3.17.1
 - ajout de l'attribut **mode**.
- Nouvelle méthode `ftth.getInfo` Section 3.5.1
- Méthode `dsl.getInfo` Section 3.4.1
 - ajout des attributs **line_status**, **training**.
- Nouvelle méthode `ont.sync` Section 3.9.2
- Nouvelle méthode `ont.push` Section 3.9.3
- Nouvelle méthode `ont.pull` Section 3.9.4

1.6 Firmware 3.2.3

- Nouvelle méthode `ont.getInfo` Section 3.9.1

1.7 Firmware 3.2.1

- Méthode `ddns.getInfo` Section 3.3.4
 - ajout des attributs **lastfreeze**, **lastfreezetime**.
 - Nouvelle méthode `smb.getInfo` Section 3.12.1
 - Nouvelle méthode `p910nd.getInfo` Section 3.10.1
-

1.8 Firmware 3.2.0

- Nouvelle méthode `system.setRefClient` Section 3.13.6
- Nouvelle méthode `lan.getDnsHostList` Section 3.8.3
- Nouvelle méthode `lan.addDnsHost` Section 3.8.1
- Nouvelle méthode `lan.deleteDnsHost` Section 3.8.2
- Nouvelle méthode `wlan.setWl0EncType` Section 3.18.8
- Méthode `wlan.getInfo` Section 3.18.4:
 - ajout de l'attribut `enctype`.
- Méthode `lan.getHostsList` Section 3.8.4:
 - ajout des attributs `type`, `probe`, `alive`, `status`.
- Méthode `system.getInfo` Section 3.13.1:
 - 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

1.9 Firmware 3.1.0

- Mise à jour des valeurs des modes hotspot dans `hotspot.getInfo` Section 3.7.4

1.10 Firmware 3.0.14

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

1.11 Firmware 3.0.7

- Nouvelle méthode `voip.getCallhistoryList` Section 3.16.1
- Nouvelle méthode `lan.getHostsList` Section 3.8.4
- Nouvelle méthode `system.reboot` Section 3.13.4
- Méthode `lan.getInfo` Section 3.8.5:
 - ajout des attributs `dhcp_active`, `dhcp_start`, `dhcp_end` et `dhcp_lease`.
- Méthode `voip.getInfo` Section 3.16.2:
 - ajout des attributs `hook_status` et `callhistory_active`.
- Méthode `wlan.getInfo` Section 3.18.4:
 - ajout de l'attribut `mac_filtering`.

1.12 Firmware 3.0.6

- Méthode `wan.getInfo` Section 3.17.1:
 - la méthode `wan.getInfo` est dorénavent public.

1.13 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 3.17.1:
 - ajout de l'attribut `infra`.
- Méthode `voip.getInfo` Section 3.16.2:
 - ajout de l'attribut `infra`.
- Méthode `lan.getInfo` Section 3.8.5:
 - la méthode est dorénavent public.
- Méthode `firewall.getInfo` Section 3.6.3:
 - 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 3.13.1:
 - ajout de l'attribut `version_dslriver`.
 - ajout de l'attribut `net_infra`.
- Correction d'erreurs diverses:
 - La méthode `ppp.setCredentials` Section 3.11.3 est corrigée.

1.14 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 (GET)

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

Exemple d'appel d'une méthode avec curl (POST)

```
$ curl --data 'channel=36' http://192.168.1.1/api/1.0/?method=wlan5.setChannel
```

- 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
116	Invalid authentication	Authentification invalide
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 désactiver 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 paramètre hash lors de l'appel de la méthode auth.checkToken. Ce paramètre 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 hexadécimale) et se calcul ainsi (value étant la valeur à 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 Exemple 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 exécutés 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 : Authentication 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 définit 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 définit 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 définit 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 service 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 service
-

- 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.
- balise **dsl** > attribut **line_status** = (No Defect|Of Frame|Loss Of Signal|Loss Of Power|Loss Of Signal Quality|Unknown) . Etat détaillé du lien. (*firmware* >= 3.3.2)
- balise **dsl** > attribut **training**= (Idle|G.994 Training|G.992 Started|G.922 Channel Analysis|G.992 Message Exchange|G.993 Started|G.993 Channel Analysis|G.993 Message Exchange|Showtime|Unknown) . Etat de négociation avec le DSLAM. (*firmware* >= 3.3.2)

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 ftth

3.5.1 ftth.getInfo

- Méthode HTTP : GET
- Accès : public
- Description : informations sur le lien FTTH.
- Retour :
 - balise **ftth** > attribut **status** = (up|down) . Etat du lien. (*firmware* >= 3.3.2)
 - balise **ftth** > attribut **wanfibre** = (in|out) Etat du port fibre par rapport au bridge wan0. (*firmware* >= 3.5.0)

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
  <ftth status="up" wanfibre="in"/>
</rsp>
```

3.6 firewall

3.6.1 firewall.enableSmtptFilter

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

3.6.2 firewall.disableSntpFilter

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

3.6.3 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>
```

3.7 hotspot

3.7.1 hotspot.enable

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

3.7.2 hotspot.disable

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

3.7.3 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>
```

3.7.4 hotspot.getInfo

Note

L'attribut *mode* n'existe plus depuis le firmware 3.4

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

Exemple

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

3.7.5 hotspot.setMode

Note

N'existe plus depuis le firmware 3.4

3.7.6 hotspot.restart

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

3.7.7 hotspot.start

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

3.7.8 hotspot.stop

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

3.8 lan

3.8.1 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.

3.8.2 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

3.8.3 lan.getDnsHostList

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

Exemple

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

3.8.4 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|lan4|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** = +(pclstblfemt|plcl...) 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>
```

3.8.5 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>
```

3.9 ont

L'ONT ou Optical Network Terminal est un boîtier optique externe connecté à l'interface de la BOX.

3.9.1 ont.getInfo

Note

Existe depuis le firmware 3.2.3

- Méthode HTTP : GET
 - Accès : public
 - Description : informations sur le boîtier ONT. Veuillez utiliser la méthode **ont.sync** Section 3.9.2 pour mettre à jour les données.
-

- Retour :
 - balise **ont** > attribut **type** = version matériel de l'ONT.
 - balise **ont** > attribut **version** = version du firmware de l'ONT.
 - balise **ont** > attribut **status** = (up|down). Status du service. (*firmware* >= 3.3.2)
 - balise **ont** > attribut **sn** = numéro de série de l'ONT. (*firmware* >= 3.3.2)
 - balise **ont** > balise **info** > attribut **name** et **value** :
 - * Valeur de l'attribut name pour un ONT de type **I-010**
 - slid : Identifiant de connexion ONT
 - ranging : Rangement de l'ONT sur PON
 - uptime : Temps de service
 - receive : Intensité signal optique reçue (dBm)
 - transmit : Intensité signal optique transmise (dBm)
 - bias : BIAS (mA)
 - temp : Température
 - vcc : Tension Vcc
 - lanlos : Liaison Ethernet entre l'ONT et la BOX
 - lossframe : Perte de paquets sur la fibre
 - lossgem : Problème lors du traitement des paquets
 - lossignal : Perte du signal optique
 - errmsg : Message d'erreur sur la couche physique
 - omcc : Perte ou problème de communication sur la fibre
 - deactivate : Echec de désactivation de l'ONT
 - ranged : ONT rangé sur le PON
 - phyerror : ONT hors service
 - rdi : Erreurs dans le signal reçu par l'ONT
 - failed : Perte de signal, seuil d'erreur
 - slidattempts : Nombre de tentative de mis à jour du slid
 - updatecounter : Compteur de mis à jour
 - versionpassive : version du firmware de secours
 - ledseqpt : Statut du voyant équipement
 - ledspn : Statut du voyant PON
 - ledsmgmt : Statut du voyant gestion et administration
 - ledslan : Statut du voyant de la liaison Ethernet
 - dlfail : Status du téléchargement du firmware
 - dlidle : Status du téléchargement du firmware
 - dlip : Status du téléchargement du firmware
 - dlstart : Status du téléchargement du firmware
 - ledseqptstate : Statut de clignotement du voyant équipement (*firmware* >= 3.4)
 - ledspnstate : Statut de clignotement du voyant PON (*firmware* >= 3.4)
 - ledsmgmtstate : Statut de clignotement du voyant gestion et administration (*firmware* >= 3.4)
 - ledslanstate : Statut de clignotement du voyant de la liaison Ethernet (*firmware* >= 3.4)

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <ont type="I-010G-Q" status="up" sn="ALCLF5A03FA0" version="3FE53583AOXA18">
    <info name="slid" value="1234" />
    <info name="ranging" value="IS" />
    <info name="uptime" value="6 hours 59 minutes 28 seconds..." />
  </ont>
</rsp>
```

```

<info name="receive" value="-23.02" />
<info name="transmit" value="2.50" />
<info name="bias" value="17.94" />
<info name="temp" value="53.12" />
<info name="vcc" value="3.32" />
<info name="lanlos" value="INACTIVE" />
<info name="lossframe" value="INACTIVE" />
<info name="lossgem" value="INACTIVE" />
<info name="losssignal" value="INACTIVE" />
<info name="errmsg" value="INACTIVE" />
<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" />
<info name="slidattempts" value="0" />
<info name="defaultslid" value="0" />
<info name="updatecounter" value="4" />
<info name="versionpassive" value="3FE53583AOXA91" />
<info name="ledseqpt" value="GREEN" />
<info name="ledspon" value="GREEN" />
<info name="ledsmgmt" value="GREEN" />
<info name="ledslan" value="GREEN" />
<info name="dlfail" value="INACTIVE" />
<info name="dlidle" value="ACTIVE" />
<info name="dlip" value="INACTIVE" />
<info name="dlstart" value="INACTIVE" />
<info name="ledseqptstate" value="NONE" />
<info name="ledsponstate" value="SOLID" />
<info name="ledsmgmtstate" value="BLINKING" />
<info name="ledslanstate" value="NONE" />
</ont>
</rsp>

```

3.9.2 ont.sync

- Méthode HTTP : POST
- Accès : public depuis (*firmware* >= 3.4)
- Description : Synchronise les informations ONT avec la BOX. (*firmware* >= 3.3.2)

3.9.3 ont.push

Note

Cette méthode est asynchrone. Veuillez utiliser la méthode [ont.pull](#) Section 3.9.4 pour obtenir un code de retour.

- Méthode HTTP : POST
 - Accès : privé
 - Description : Envoi des informations de configuration à l'ONT (*firmware* >= 3.3.2)
 - Paramètre requête :
 - **name** = (*slid*) : Définit l'identifiant à modifier
 - **value** : Définit la valeur de remplacement
-

- **force** : Ce paramètre facultatif, permet de forcer le remplacement de la nouvelle valeur.
- * Valeurs possible de l'attribut **name** :
 - slid : Identifiant de connexion ONT.

3.9.4 ont.pull

- Méthode HTTP : GET
- Accès : privé
- Description : Retourne le statut d'un **ont.push** Section 3.9.3 (*firmware* >= 3.3.2)
- Retour :
 - balise **return** > attribut **pushname** = (slid)
 - balise **return** > attribut **code** : contient un code de retour.
 - * Les codes :
 - -1 : En cours d'exécution
 - 0 : Exécuté ou état par défaut
 - 1 : Erreur interne
 - 2 : ONT injoignable
 - 3 : Erreur lors de la communication avec l'ONT
 - >=10 : Retours spécifiques pour chaque push
 - * Pour le push du **slid** les codes de retour sont :
 - 10 : L'ONT est rangé.
 - 11 : Le SLID est fourni par le réseau
 - 12 : Le SLID est déjà présent sur la BOX
 - 13 : Le SLID présent sur la BOX a été remplacé
 - 14 : Le SLID a déjà été modifié plus de 9 fois

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <return pushname="slid" code="-1" />
</rsp>
```

3.10 p910nd

3.10.1 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>
```

3.11 ppp

3.11.1 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>
```

3.11.2 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>
```

3.11.3 ppp.setCredentials

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

3.12 smb

3.12.1 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>
```

3.13 system

3.13.1 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)
- balise **system** > attribut **idur**. Identifiant unique réseau. (*firmware* >= 3.4.0)
- balise **system** > attribut **alimvoltage**. Tension de l'alimentation exprimé en mV. (*firmware* >= 3.5.0)
- balise **system** > attribut **temperature**. Température de la BOX exprimé en m°C. (*firmware* >= 3.5.0)
- balise **system** > attribut **serialnumber**. Numéro de série de l'IAD. (*firmware* >= 4.0.0)

Exemple

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
  <system product_id="NB5-SER-r1"
    serial_number="XXXXXXXXXXXXXXXXXXXX"
    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="1-1234"
    idur="XXXXXXX"
    alimvoltage="YYYYY"
    temperature="YYYYY" />
</rsp>
```

3.13.2 system.getIfList

- Méthode HTTP : GET
- Accès : privé
- Description : clé WPA par défaut de la BOX.
- Retour :
 - balise **interface** > attribut **name**. Nom de l'interface.
 - balise **interface** > attribut **mac**. Adresse MAC de l'interface.
 - Si erreur :
 - * balise **err** > attribut **code** contient le code d'erreur :
 - 500 : Cannot retrieve interface list.

Exemple

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

  <interface name="lo" mac="00:00:00:00:00:00">

  <interface name="sit0" mac="00:00:00:00:00:08">

  <interface name="ip6tnl0" mac="00:00:00:00:00:00">

  <interface name="bcm5w" mac="DE:AD:BE:EF:FA:CE">

  <interface name="eth0" mac="DE:AD:BE:EF:FA:CE">
```

```
<interface name="eth1" mac="DE:AD:BE:EF:FA:CE">
<interface name="eth2" mac="DE:AD:BE:EF:FA:CE">
<interface name="eth3" mac="DE:AD:BE:EF:FA:CE">
<interface name="eth4" mac="DE:AD:BE:EF:FA:CE">
<interface name="wl0" mac="DE:AD:BE:EF:FA:CF">
<interface name="wl1" mac="DE:AD:BE:EF:FA:D0">
<interface name="hotspot0" mac="62:AD:BE:EF:FA:CC">
<interface name="lan0" mac="DE:AD:BE:EF:FA:CE">
<interface name="stb0" mac="DE:AD:BE:EF:FA:CE">
<interface name="guest0" mac="62:AD:BE:EF:FA:CD">
<interface name="wan0" mac="DE:AD:BE:EF:FA:CD">

</rsp>
```

3.13.3 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>
```

3.13.4 system.reboot

Note

Existe depuis le firmware 3.0.7

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

3.13.5 system.setNetMode

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

3.13.6 system.setRefClient

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

3.14 tv

3.14.1 tv.getInfo

Note

Existe depuis le firmware 3.4

- Méthode HTTP : GET
 - Accès : public
 - Description : Informations TV.
 - Retour :
 - balise **tv** > attribut **infra**. Lien utilisé pour le flux TV.
 - balise **tv** > attribut **lanchain**. Représentation au format Newick de l'installation entre la BOX et les différentes BOX TV.
 - balise **tv** > attribut **netchain**. Représentation de l'installation entre la box et le réseau du FAI. (*firmware* >= 3.5.0)
 - balise **igmpps** > balise **igmp**. Flux IGMP en cours d'utilisation.
 - balise **igmpps** > balise **igmp** > attribut **ipaddr**. Adresse IP enregistré au flux multicast.
 - balise **igmpps** > balise **igmp** > attribut **group**. Adresse du flux multicast.
 - balise **igmpps** > balise **igmp** > attribut **ether**. Adresse ethernet multicast.
-

3.15 usb

3.15.1 usb.getInfo

Note

Existe depuis le firmware 4.0

- Méthode HTTP : GET
- Accès : public
- Description : Informations sur les périphériques branchés en USB.
- Retour :
 - balise **device** > attribut **name**. Nom du périphérique.
 - balise **device** > attribut **product**. ProductID.
 - balise **device** > attribut **bus**. Bus USB utilisé.
 - Si pas de périphérique USB:
 - * pas de balise **device**.

Exemple

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

  <device name="Mass Storage Device" product="8564/1000/1100" bus="001-002"/>

</rsp>
```

3.16 voip

3.16.1 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>
```

3.16.2 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>
```

3.16.3 voip.restart

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

3.16.4 voip.start

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

3.16.5 voip.stop

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

3.17 wan

3.17.1 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 v4 ou v6 (suivant les cas)
 - balise **wan** > attribut **ip_addr**. Adresse IPv4 internet. (MAP-T inclus, peut être vide si Only sans conf MAP-T))
 - balise **wan** > attribut **infra** = (adsl|ftth|gprs) . Lien utilisé pour la connexion internet. (*firmware* >= 2.1.2)
 - balise **wan** > attribut **mode** = (ftth/routed|adsl/routed|adsl/ppp|gprs/ppp) . Mode de connexion internet. (*firmware* >= 3.3.2)
 - balise **wan** > attribut **infra6** = (Tunnel|Dual|Native|Only|unknown) . Lien utilisé pour la connexion internet IPv6. (*firmware* >= 3.4)
 - balise **wan** > attribut **status6** = (up|down) . Status de la connexion internet IPv6. (*firmware* >= 3.4)
 - balise **wan** > attribut **uptime6**. Temps de connexion internet IPv6. (*firmware* >= 3.4)
 - balise **wan** > attribut **ipv6_addr**. Adresse IPv6 globale de la box. (*firmware* >= 3.4)

Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
<wan status="up" uptime="9591" ip_addr="" infra="ftth" mode="ftth/routed" infra6=" ←
  Only" status6="up" uptime6="9571" ipv6_addr="2a02:8428:27d:8300::1/56" />
</rsp>
```

3.18 wlan

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

- La NB6/NB6VAC les modes 11n, 11b/g/n et 11b/g en 2.4Ghz et les modes 11n/11ac en 5Ghz;
- La NB6/NB6V 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

3.18.1 wlan.enable

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

3.18.2 wlan.disable

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

3.18.3 wlan.getClientList

- Méthode HTTP : GET
- Accès : privé
- Description : liste des clients WiFi 2.4Ghz .

Exemple

```
$ curl http://192.168.1.1/api/1.0/?method=wlan.getClientList
<?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>
```

3.18.4 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 2.4Ghz.
- 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

```
$ curl http://192.168.1.1/api/1.0/?method=wlan.getInfo
<?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>
```

3.18.5 wlan.getScanList

Note

Existe depuis le firmware 4.0

- Méthode HTTP :
- Accès : privé
- Description : liste des réseaux Wifi environnants.
- Retour :
 - balise **wifi** > attribut **freq**. Fréquence de l'interface WLAN.
 - balise **wifi** > balise **ap** > attribut **ssid**. SSID du réseau WLAN détecté.
 - balise **wifi** > balise **ap** > attribut **bssid**. Adresse MAC du point d'accès WLAN.
 - balise **wifi** > balise **ap** > attribut **rssi**. Gain du point d'accès en dBm.
 - balise **wifi** > balise **ap** > attribut **channel**. Channel utilisé par le réseau WLAN.

Exemple

```

$ curl http://192.168.1.1/api/1.0/?method=wlan.getScanList
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
  <wifi freq="5GHz">
    <ap ssid="TFM" bssid="98:4B:E1:26:5C:A0" rssi="-87 dBm" channel="108"/>
    <ap ssid="SFR" bssid="98:4B:E1:26:5C:A4" rssi="-88 dBm" channel="108"/>
    <ap ssid="SFR" bssid="98:4B:E1:26:39:E4" rssi="-91 dBm" channel="36"/>
    <ap ssid="TFM" bssid="98:4B:E1:26:39:E0" rssi="-91 dBm" channel="36"/>
    <ap ssid="TFM" bssid="98:4B:E1:26:7C:20" rssi="-61 dBm" channel="44"/>
    <ap ssid="SFR" bssid="98:4B:E1:26:7C:24" rssi="-61 dBm" channel="44"/>
    <ap ssid="SFR_FIBER_FACF_5GHZ" bssid="DE:AD:BE:EF:FA:CF" rssi="-89 dBm" channel="52"/>
    <ap ssid="SFR_FIBER_0039_5GHZ" bssid="44:CE:D7:50:00:39" rssi="-51 dBm" channel="60"/>
  </wifi>
  <wifi freq="2.4GHz">
    <ap ssid="SFR_FIBER_0039" bssid="44:CE:D7:50:00:3A" rssi="-50 dBm" channel="1"/>
    <ap ssid="Bcm2.4" bssid="02:20:20:B0:0B:02" rssi="-52 dBm" channel="1"/>
    <ap ssid="BcmHotspot2.4" bssid="72:20:20:B0:0B:03" rssi="-52 dBm" channel="1"/>
    <ap ssid="BcmGuest2.4" bssid="72:20:20:B0:0B:00" rssi="-52 dBm" channel="1"/>
    <ap ssid="BcmEAPSIM2.4" bssid="72:20:20:B0:0B:01" rssi="-52 dBm" channel="1"/>
    <ap ssid="" bssid="98:4B:E1:26:6C:32" rssi="-86 dBm" channel="1"/>
    <ap ssid="SFR_0008" bssid="30:7E:CB:A8:00:0C" rssi="-40 dBm" channel="1"/>
    <ap ssid="SFR_OED0" bssid="30:7E:CB:B0:0E:D4" rssi="-67 dBm" channel="1"/>
    <ap ssid="SFR WiFi Mobile" bssid="9A:7E:CB:B0:0F:37" rssi="-57 dBm" channel="6"/>
    <ap ssid="TFM" bssid="98:4B:E1:26:39:F0" rssi="-76 dBm" channel="6"/>
    <ap ssid="SFR" bssid="98:4B:E1:26:39:F4" rssi="-75 dBm" channel="6"/>
    <ap ssid="SFR WiFi FON" bssid="9A:CE:7D:4F:F2:35" rssi="-71 dBm" channel="6"/>
    <ap ssid="SFR WiFi Mobile" bssid="9A:CE:7D:4F:F2:37" rssi="-71 dBm" channel="6"/>
    <ap ssid="SFR_F1F0" bssid="44:CE:7D:4F:F1:F4" rssi="-38 dBm" channel="6"/>
    <ap ssid="private_0030" bssid="00:1D:16:05:03:35" rssi="-78 dBm" channel="1"/>
    <ap ssid="SFR" bssid="98:4B:E1:26:6C:33" rssi="-87 dBm" channel="1"/>
  </wifi>
</rsp>

```

3.18.6 wlan.setChannel

- Méthode HTTP : POST
- Accès : privé
- Description : définir le canal WiFi 2.4Ghz.
- Paramètre requête :
 - **channel** (entre 1 et 13)

3.18.7 wlan.setWl0Enc

- Méthode HTTP : POST
- Accès : privé
- Description : définir la sécurité du réseau WiFi 2.4Ghz.
- Paramètre requête :
 - **enc** = (OPEN | WEP | WPA-PSK | WPA2-PSK | WPA-WPA2-PSK)

3.18.8 wlan.setWl0Enctype

Note

Existe depuis le firmware 3.2.0

- Méthode HTTP : POST
- Accès : privé
- Description : définir le type de clé WPA 2.4Ghz.
- Paramètre requête :
 - **enctype** = (tkip|aes|tkipaes)

3.18.9 wlan.setWl0Keytype

- Méthode HTTP : POST
- Accès : privé
- Description : définir le type de clé WEP 2.4Ghz.
- Paramètre requête :
 - **keytype** = (ascii|hexa)

3.18.10 wlan.setWl0Ssid

- Méthode HTTP : POST
- Accès : privé
- Description : définir le nom du réseau WiFi 2.4Ghz.
- Paramètre requête :
 - **ssid**

3.18.11 wlan.setWl0Wepkey

- Méthode HTTP : POST
- Accès : privé
- Description : définir la clé WEP du réseau WiFi 2.4Ghz.
- Paramètre requête :
 - **wepkey**

3.18.12 wlan.setWl0Wpakey

- Méthode HTTP : POST
 - Accès : privé
 - Description : définir la clé WPA du réseau WiFi 2.4Ghz.
 - Paramètre requête :
 - **wpakey**
-

3.18.13 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 2.4Ghz.
- Paramètre requête :
 - (Pour NB5/NB6) **mode** = (11n|11ng|11g)
 - (Pour NB4/CIBOX) **mode** = (11b|11g|auto)

3.18.14 wlan.start

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

3.18.15 wlan.stop

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

3.18.16 wlan.restart

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

3.19 wlan5

3.19.1 wlan5.getClientList

- Méthode HTTP : GET
- Accès : privé
- Description : liste des clients WiFi connectés au réseau 5Ghz.

Exemple

```
$ curl http://192.168.1.1/api/1.0/?method=wlan5.getClientList
<?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>
```


3.19.2 wlan5.getInfo

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur le WiFi 5Ghz.
- Retour :
 - balise **wlan** > attribut **active** = (on|off). Activation.
 - balise **wlan** > attribut **channel**. Canal.
 - balise **wlan** > attribut **mode** = (auto|11n|11ac). Mode radio.
 - balise **wlan** > attribut **mac_filtering** = (whitelist|blacklist|off).
 - balise **wlan** > balise **wl0** > attribut **ssid**. Nom du réseau.
 - balise **wlan** > balise **wl0** > attribut **enc** = (OPEN|WPA-PSK|WPA2-PSK|WPA-WPA2-PSK). Encryption.
 - balise **wlan** > balise **wl0** > attribut **enctype** = (tkip|aes|tkipaes).
 - balise **wlan** > balise **wl0** > attribut **keytype** = (ascii|hexa).
 - balise **wlan** > balise **wl0** > attribut **wpakey**. Clé WPA.

Exemple

```
$ curl http://192.168.1.1/api/1.0/?method=wlan5.getInfo
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
  <wlan active="on" channel="36" mode="11n" mac_filtering="off">
    <wl0 ssid="SFR_34E0" enc="OPEN" keytype="ascii" wpakey="12345678" wepkey="↔
      1234567890123" />
  </wlan>
</rsp>
```

3.19.3 wlan5.setChannel

- Méthode HTTP : POST
- Accès : privé
- Description : définir le canal WiFi 5Ghz.
- Paramètre requête :
 - **channel** (auto|36|40|44|48|52|56|60|64|100|104|108|112|116|132|136|140)

Exemple

```
$curl --data 'channel=36' 'http://192.168.1.1/api/1.0/?method=wlan5.setChannel'
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok" version="1.0">
</rsp>
```

3.19.4 wlan5.setWl0Enc

- Méthode HTTP : POST
- Accès : privé
- Description : définir la sécurité du réseau WiFi 5Ghz.
- Paramètre requête :
 - **enc** = (OPEN|WPA-PSK|WPA2-PSK|WPA-WPA2-PSK)

3.19.5 wlan5.setWl0Enctype

- Méthode HTTP : POST
- Accès : privé
- Description : définir le type de clé WPA 5Ghz.
- Paramètre requête :
 - **enctype** = (tkip|aes|tkipaes)

3.19.6 wlan5.setWl0Keytype

- Méthode HTTP : POST
- Accès : privé
- Description : définir le type de clé WEP 5Ghz.
- Paramètre requête :
 - **keytype** = (ascii|hexa)

3.19.7 wlan5.setWl0Ssid

- Méthode HTTP : POST
- Accès : privé
- Description : définir le nom du réseau WiFi 5Ghz.
- Paramètre requête :
 - **ssid**

3.19.8 wlan5.setWl0Wepkey

- Méthode HTTP : POST
- Accès : privé
- Description : définir la clé WEP du réseau WiFi 5Ghz.
- Paramètre requête :
 - **wepkey**

3.19.9 wlan5.setWl0Wpakey

- Méthode HTTP : POST
 - Accès : privé
 - Description : définir la clé WPA du réseau WiFi 5Ghz.
 - Paramètre requête :
 - **wpakey**
-

3.19.10 wlan5.setWlanMode

- Méthode HTTP : POST
- Accès : privé
- Description : définir le mode radio WiFi 5Ghz.
- Paramètre requête :
 - **mode** = (11n|11ac|auto)

3.20 guest

3.20.1 guest.getInfo

- Méthode HTTP : GET
- Accès : privé
- Description : informations sur le WiFi guest.
- Retour :
 - balise **guest** > attribut **active** = (on|off). Activation.
 - balise **guest** > attribut **ssid**. Nom du réseau.
 - balise **guest** > attribut **wpakey**. Clé WPA.

Exemple

```
$ curl 'http://192.168.1.1/api/1.0/?method=guest.getInfo'  
<?xml version="1.0" encoding="UTF-8"?>  
<rsp stat="ok" version="1.0">  
  <guest active="on"  
    <config ssid="SFR_34E0_GUEST" wpakey="abcdefgh" />  
  </guest>  
</rsp>
```

3.20.2 guest.enable

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

Exemple

```
$ curl --data '' 'http://192.168.1.1/api/1.0/?method=guest.enable'  
<?xml version="1.0" encoding="UTF-8"?>  
<rsp stat="ok" version="1.0">  
</rsp>
```

3.20.3 guest.disable

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

Exemple

```
$ curl --data '' 'http://192.168.1.1/api/1.0/?method=guest.disable'  
<?xml version="1.0" encoding="UTF-8"?>  
<rsp stat="ok" version="1.0">  
</rsp>
```

3.20.4 guest.setSsid

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

Exemple

```
$ curl --data 'ssid=SFR_34E0_GUEST' 'http://192.168.1.1/api/1.0/?method=guest.setSsid'  
<?xml version="1.0" encoding="UTF-8"?>  
<rsp stat="ok" version="1.0">  
</rsp>
```

3.20.5 guest.setWpakey

- Méthode HTTP : POST
- Accès : privé
- Description : définir la clé WPA du réseau WiFi GUEST.
- Paramètre requête :
 - wpakey

Exemple

```
$ curl --data 'wpakey=abcdefgh' 'http://192.168.1.1/api/1.0/?method=guest.setWpakey'  
<?xml version="1.0" encoding="UTF-8"?>  
<rsp stat="ok" version="1.0">  
</rsp>
```

4 Annexe

4.1 Code de hashage en C

hash.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++)
    {
        sprintf((*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