

Composant Serveur API REST pour WinDev®

Léger.

Rapide.

Gratuit & Open Source.

Sans serveur IIS/Apache/...

Sans licence Serveur WebDev®.

Pas de SAAS, pas d'abonnement.

Windows 32/64 & Linux

Présentation

NEW : LightREST V3

La Documentation



Téléchargements

[Un coup de main ?](#)

Classe IrServer



Role

Configuration et exécution d'un serveur LightREST (HTTP/HTTPS), routage, timeouts, headers globaux, hooks (niveau serveur et route), et chiffrement/déchiffrement d'identifiants.



Membres et propriétés

Nom	Type	Usage
IP Adresse	chaîne	Interface réseau et port d'écoute. Si on indique une IP = 0.0.0.0, le serveur LightREST écoutera toutes les interfaces (dont 127.0.0.1). Exemple : 0.0.0.0:9000 ou 192.168.1.150:8888
CustomData	Variante	Données personnalisées associées au serveur, qui seront transmises automatiquement à chaque requête dans le propriété <u><i>IrRequest:ServerCustomData</i></u> . Utile par exemple pour y stocker des paramètres globaux et éviter de les relire à chaque exécution d'une requête.
HTTPMode	Énumération	Mode de chiffrement de la communication. Selon le mode HTTP choisi, il conviendra de

on déterminer les
 en paramètres de gestion
 HT du certificat SSL avec
 TP ces méthodes:
 M - HTTP : Aucun
 od chiffrement. Mode par
 e défaut (utile pour
 développer en mode
 local, déconseillé en
 production)
 - HTTPS_CERT :
 SSL avec certificat
 fourni avec la méthode
SetHTTPSCertificate
 - HTTPS_SELF :
 SSL avec certificat
 auto-généré,
 paramètres fournis
 avec la méthode
SetSelfCertificateParameters
 -
 HTTPS_LETS_ENCRYPT
 T : SSL avec certificat
 Let's Encrypt auto-
 renouvelé, paramètres
 fournis avec la
 méthode
SetLetsEncryptParameters

Sta	En	État courant du
te	u	serveur (lecture seule).
	m	Valeurs possibles :
	ér	<u>STARTING, STARTED,</u>

	ati on en Se rv er St at e	<u>STOPPING, STOPPED.</u>
Tem pFo lde r	ch aî ne	Répertoire temporaire utilisé par le serveur. Par défaut = le répertoire temporaire du user (retourné par la fonction WinDev <u>fRépertoireTemp()</u>). Sert au stockage de fichiers temporaires par le composant.

Ciphering & RegEx

Nom	Type	Usage
Cip her ing Key	Bu ffe r	Clé de chiffrement utilisée par les méthodes <u>CipherID</u> et <u>DecipherID</u>. ⚠ Une clé est définie par défaut lors de

l'instanciation de l'objet. En cas de redémarrage du serveur, une nouvelle clé sera affectée. Donc si des clients envoient des requêtes REST comprenant des IDs chiffrés avec la clé initiale, ceux-ci seront inexploitable. Pour s'en prémunir, il faut donc affecter une clé fixe, ou bien avec une rotation régulière; ainsi les IDs chiffrées resteront valides en cas de redémarrage.

NoR ege x	Bo ol ee n	Désactive la vérification RegEx de la syntaxe des routes lors de leur ajout avec la méthode <u>:AddRoute.</u>
-----------------	---------------------	---

Valeur par défaut =
Faux

Timeouts & MaxRequests

Nom	Type	Usage
Idl	Du	Timeout d'inactivité

entre deux requêtes
sur une même
connexion HTTP
(keep-alive).
Valeur par défaut =
60s (constante
:DEFAULT_IDLE_TIMEO
UT)

Ce délai s'applique
uniquement
lorsqu'aucune requête
n'est en cours et que le
serveur attend une
nouvelle requête sur la
même connexion.
Lorsque la requête
suivante arrive, elle est
traitée immédiatement
dans renégociation de
la connexion, donc
gain de performances.
En cas de
dépassement du délai,
le serveur ferme
proprement la
connexion et libère les
ressources associées.

i IdleTimeout ne
limite pas la durée
d'exécution maximale
d'une route. Pour cela,
utiliser WriteTimeout
(et/ou un timeout
spécifique à la route).

ReadTimeout	Durée	Timeout de lecture sur la connexion du client REST. Valeur par défaut = 30s (constante :DEFAULT_READ_TIMEOUT) En cas de dépassement, les ressources sont libérées côté serveur. Aucune erreur n'est renvoyée au client (car on considère que la connexion est tombée, donc communication impossible).
RequestTimeout	Durée	Timeout par défaut appliqué aux routes. Valeur par défaut = 30s (constante :DEFAULT_REQUEST_TIMEOUT) En cas de dépassement, l'exécution du Handler REST WinDev est annulée et une erreur HTTP 408 (<u>StatusRequestTimeout</u>) est envoyée au

client.

 Peut être surchargé spécifiquement sur une route avec la propriété *IrRoute:Timeout*

WriteTimeout

Durée

Timeout d'écriture sur la connexion du client REST.
Valeur par défaut = 30s (constante :DEFAULT_WRITE_TIMEOUT).

En cas de dépassement, les ressources sont libérées côté serveur. Aucune erreur n'est renvoyée au client (car on considère que la connexion est tombée, donc communication impossible).

 WriteTimeout doit être supérieur au plus grand timeout des routes (ou du serveur) + une marge pour laisser le temps d'envoyer les entêtes. Si WriteTimeout < Timeout route, la

connexion avec le client sera coupée avant que le timeout soit arrivé à échéance.

Max Req ues ts	en tie r	Nombre maximum de requêtes traitées simultanément par le serveur. Valeur par défaut : 100 (constante :DEFAULT_MAX_REQUEST) Permet d'éviter la surcharge du serveur. Lorsque le nombre de requête maximal est atteint, une erreur HTTP 503 (<i>StatusServiceUnavailable</i>) est envoyée au client.
-------------------------	----------------	---

Monitoring & log

Nom	Type	Usage
Log ger	bo ol ée n	Active ou désactive le logger interne du moteur LightREST. Par défaut = Faux. Chronophage et

verbeux, à n'utiliser principalement qu'en phase de debug du composant. Ecrit par défaut dans le fichier Ir_log situé dans le répertoire de l'exécutable. Pour indiquer un autre nom, utiliser la propriété <i>:LoggerFile</i>.		
LoggerFile	chaine	Chemin du fichier de log lorsque le logger est activé.
Monitoring	boolean	Active le monitoring des requêtes. Valeur par défaut = Faux Ajoute dans le HEADER de chaque réponse les temps : - global du traitement de la requête dans <i>Chrono-Global-Millisec</i> - d'exécution du Handler REST WinDev dans <i>Chrono-Methode-Millisec</i>
WinDevErr	entier	Niveau de détail pour des erreurs retournées au client.

orD	Défaut = ErrComplet
eta	(constante
ils	:DEFAULT_WINDEV_ER ROR_LEVEL)

Contrôle la verbosité des erreurs WinDev envoyées au client REST.

LightREST récupère (dans la mesure du possible) les erreurs et exceptions survenues lors de l'exécution des méthodes REST, et envoie automatiquement au client une erreur HTTP 500 (StatusInternalServerError rror) accompagné du contenu de ErreurInfo() ou ExceptionInfo().

Le membre :WindevErrorDetails permet de choisir le niveau de détail de l'erreur envoyée, pour éviter par exemple d'exposer des informations techniques sensibles). Il doit correspondre à

une des constantes
Err* supportées par
WinDev®.

Types

enServerState – État du serveur

Type

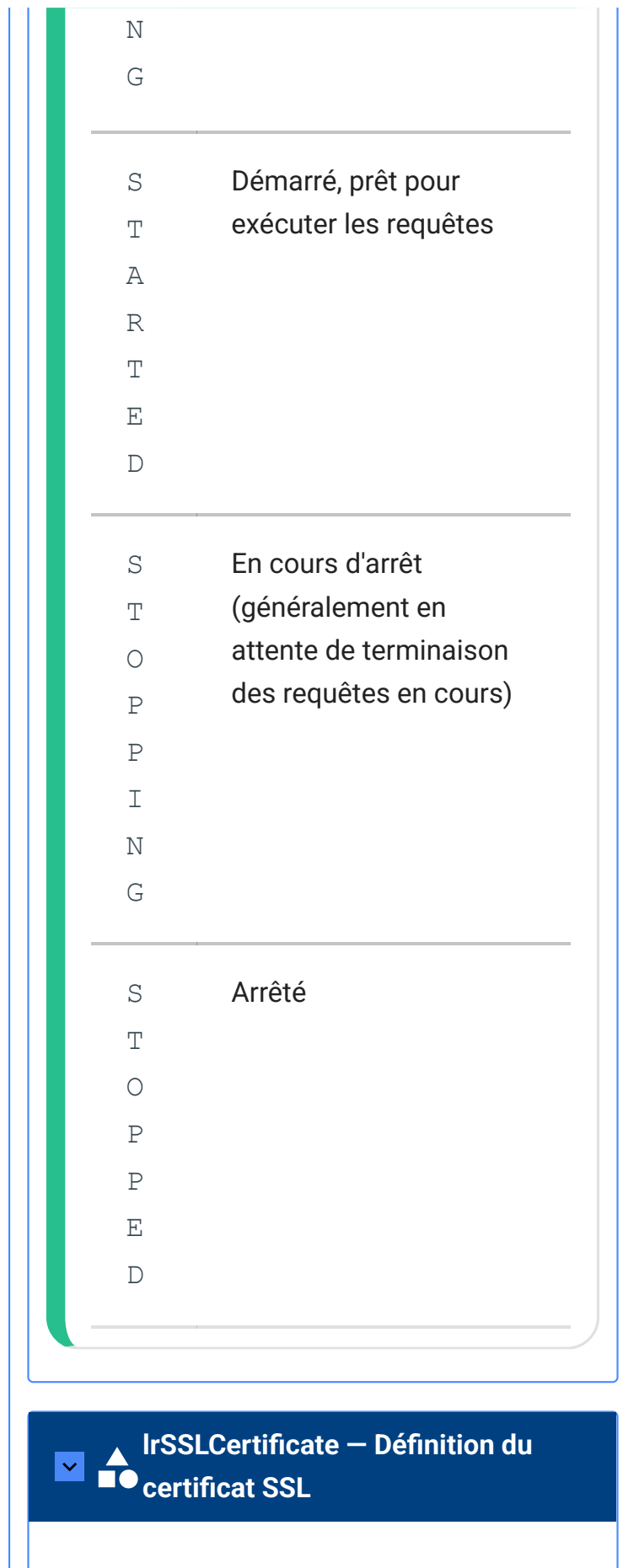
Enumération

Présentation

Indique l'état actuel du serveur
LightREST. Type retourné par la
propriété propriété :State

Valeurs

Valeur	Usage
STARTING	En cours de démarrage



 Type

Structure

 Présentation

Détermine le contenu du certificat SSL et de sa clé privée lorsque `:HttpMode = HTTPS_CERTIFICATE`. Structure passée en paramètre à la méthode `:SetHTTSPSCertificate()`.

 Structure

No m	T y p e	Usage
C e r t i f i c a t e	c h â n e	Contenu du certificat SSL
P	c	Clé privée

rh
a
î
n
e

K
e
y



stLogger — Paramètres de journalisation



Type

Structure



Présentation

Paramètres de la journalisation bas niveau (moteur LightREST GO)



Structure

No m	T y p e	Usage
b L	b o	Activation de la journalisation

o o ⚠ Chronophage,
g l à réserver au
g é debug du
e e composant.
r n

s c Nom du fichier de
L h log
o a
g î
g n
e e
r
F
i
l
e



IrLetsEncrypt – Paramètres Let's Encrypt



Type

Structure



Présentation

Structure passée en paramètre à la méthode

`:SetLetsEncryptParameters()`

**Structure**

No m	T y p e	Usage
Do m a i n s	tab l e a u d e c h a î n e s	Domaines pour lesquels le certificat Let's Encrypt doit être généré.
E m a i l	ch a î n e	Email de l'administrateur
C e r	ch a	Chemin de stockage du certificat

t
i
f
i
c
a
t
e
P
a
t
h

î
n
e

R	e	Délai de
e	n	renouvellement
n	ti	(en nombre de
e	e	jours avant
w	r	l'expiration)
D		
e		
l		
a		
y		

 **IrSelfCertificate — Paramètres du certificat auto-signé**

 **Type**

Structure

 **Présentation**

Structure passée en paramètre à la méthode
:SetSelfCertificateParameters()

 **Structure**

No m	T y p e	Usage
KeySize	entier	Taille de la clé (2048 par défaut)

 **enHTTPMode — Mode de fonctionnement HTTP/HTTPS**

 Type

Enumération

 Présentation

Permet de déterminer le mode de fonctionnement HTTP/HTTPS. Voir la propriété :HTTPMode

 Valeurs**Va
le
ur****Usage**H
T
T
P

Pas de chiffrement de la communication, pas de certificat

H
T
T
P
S

—
C
E
R
T

Certificat SSL fourni (voir :SetHTTPSCertificate())

H
T
T
P
S
—
S
E
L
F

Certificat SSL auto
généré. (voir
[:SetSelfCertificateParameters\(\)](#))

H
T
T
P
S
—
L
E
T
S
—
E
N
C
R
Y
P
T

Certificat Let's Encrypt
(voir
[:SetLetsEncryptParameters\(\)](#))

C Constantes		
Nom	Type	Usage
ROUTE_REGEX	chaîne	Expression régulière pour vérification de la syntaxe des routes REST par la méthode <u><i>AddRoute()</i></u> Valeur : <code>"^/(\$ ([A-Za-z0-9_-]+ {[A-Za-z0-9_-]+}) (/([A-Za-z0-9_-]+ {[A-Za-z0-9_-]+}))*\$)"</code>
Méthodes HTTP		
Nom	Type	Usage
Méthode GET	chaîne	Méthode REST <u><i>GET</i></u>
Méthode POST	chaîne	Méthode REST <u><i>POST</i></u>
Méthode PUT	chaîne	Méthode REST <u><i>PUT</i></u>

PUT	ne	
Met hod DEL ETE	ch aî ne	Méthode REST <u>DELETE</u>
Met hod HEA D	ch aî ne	Méthode REST <u>HEAD</u>
Met hod PAT CH	ch aî ne	Méthode REST <u>PATCH</u>
Met hod OPT ION S	ch aî ne	Méthode REST <u>OPTIONS</u>
Met hod TRA CE	ch aî ne	Méthode REST <u>TRACE</u>
Valeurs par défaut		
Nom	Ty	Usage

pe		
DEF	en	Valeur par défaut du
AUL	tie	membre
T_M	r	<u>:MaxRequests</u>
AX_		
REQ		
UES		
T		
DEF	Du	Valeur par défaut du
AUL	ré	membre
T_R	e	<u>:RequestTimeout</u>
EQU		
EST		
_TI		
MEO		
UT		
DEF	Du	Valeur par défaut du
AUL	ré	membre
T_R	e	<u>:ReadTimeout</u>
EAD		
_TI		
MEO		
UT		
DEF	Du	Valeur par défaut du
AUL	ré	membre
T_W	e	<u>:WriteTimeout</u>
RIT		
E_T		

IME		
OUT		
DEF AUL T_I DLE _TI MEO UT	Du ré e	Valeur par défaut du membre <u>:IdleTimeout</u>
DEF AUL T_W IND EV_ ERR OR_ LEV EL	En tie r	Valeur par défaut du membre <u>:WindevErrorDetails</u>

Méthodes

Routing / Hooks

  AddRoute
 Présentation

Ajoute une route REST au serveur en lui associant une fonction Handler.

i Avant ajout, la syntaxe de la route est vérifiée avec une expression régulière (`:ROUTE_REGEX`). Pour désactiver cette vérification, positionner la propriété `:NoRegex=vrai`.

Prototypage

Syntaxe 1 : `AddRoute`

Syntaxe 2 (dépréciée) : `AddRoute`

Paramètres

No m	T y p e	Usage
po Ro ut e	O bj et lr R o u t e	Définition complète de la route (méthode, route, handler, timeout, hooks, ...)

 Valeurs retournées

Ty pe	Usage
bo olé en	Vrai si la route est ajoutée.
ch aîn e	Message d'erreur (vide si succès).

 Exemple

- Démarrage d'un serveur LightREST sur le port 9876
- Création d'une route */ping* qui renvoie l'heure système et l'hôte du client REST

```
oServer est objet lrSer  
oRoute  est objet lrRoute  
bOK     est booléen  
sErr    est chaîne  
  
oServer:IPAndPort =  
"127.0.0.1:9876"  
  
oRoute.Method  =  
lrRoute::MethodGET  
oRoute.Route   = "/ping"  
oRoute.Handler = piPing
```

Copier

```

(bOK, sErr) =
oServer:AddRoute(oRoute)
SI PAS bOK ALORS
    Erreur("Erreur lors de
la création de la route : ",
sErr)
    RETOUR
FIN

(bOK, sErr) =
oServer:Start()
SI PAS bOK ALORS
    Erreur("Erreur lors du
démarrage : ", sErr)
    RETOUR
FIN

Info("Call
http://[%oServer:IPAndPort%]
[%oRoute:Route%]",
CRLF+"Click on OK to close
LightREST server")

oServer:Terminate()

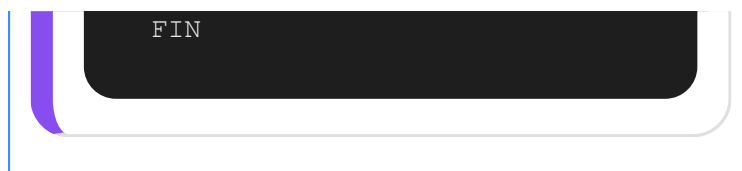
////////////////////////////////////
////////////////////////////////////

PROCEDURE INTERNE
piPing(pRequest objet
lrRequest) : objet
lrResponse
    oResp est objet
lrResponse

    oResp.Body =
"Sending pong to host " +
pRequest.Host + " at " +
DateVersChaîne(DateHeureSys(
))
    oResp.ContentType =
lrResponse::ContentTXT
    oResp.Status =
lrResponse::StatusOK

    RENVOYER oResp

```



 **AddHook**

 **Présentation**

Ajout de hook(s) serveur (interception d'événements). Déclenchement automatique de(s) hook en fonction des événements définis dans chaque objet IrHook (ouverture d'une connexion, réception requête, avant/après méthode, résultat, warning, error, panic, etc...).

 Pour un hook spécifique à une seule route (ou un ordre différent), utiliser IrRoute:AddHook().

 **Prototypage**

```
AddHook(poHook objet IrHook, [pc
```

 **Paramètres**

No m	T y p e	Usage

po	O	Hook(s) à ajouter
Ho	bj	(type
ok	et	d'évènements,
	lr	handler, options
	H	d'action sur la
	o	requête/réponse).
	o	
	o	
	k	

Valeurs retournées

(aucune)

Exemple

- Ajout d'un hook global
- Le handler de Hook, lancé automatiquement à chaque réception d'une requête REST, ajoute un header sur les réponses
- Retourne EVE_OK pour poursuivre l'exécution

```
oServer est objet lrSer
oHook  est objet lrHook
```

Copier

```
oHook:Events =
  lrHook::EVE_RECEIVED
oHook:Handler = fnHook
```

```
oServer:AddHook(oHook)
```

```
PROCEDURE INTERNE
fnHook(pEventInfo
```

```
lrHook:EventInfo) :  
lrHook:EventReturn)  
  
    // pEventInfo dépend de  
    lrHook (événement, request,  
    response, message...)  
    SI pEventInfo.Response <>  
    Null ALORS  
  
    pEventInfo.Response:SetHeade  
    rValue("X-LightREST-Hook",  
    "1")  
    FIN  
  
    RENVOYER lrHook::EVE_OK  
    FIN
```



SetAuthenticationCheckFunction



Présentation

(Déprécié)

Définit une fonction appelée pour valider l'authentification avant l'exécution de la route.

Si la fonction de contrôle retourne Faux, la requête est rejetée avec un statut 401 - StatusUnauthorized.



Préférer un hook ajouté via :AddHook() ou lrRoute:AddHook() ce qui permet un filtrage fin et un ordre d'exécution contrôlé.



Prototypage

```
SetAuthenticationCheckFunction()
```

Paramètres

No m	T y p e	Usage
pf un ct io n	p r o c é d u r e	Procédure de contrôle qui reçoit un IrRequest et retourne un booléen (Vrai/Faux).

Valeurs retournées

(aucune)

Exemple

- Ajout d'un contrôle d'accès simple basé sur un header X-API-KEY
- La fonction sera appelée automatiquement avant chaque exécution de requête REST

```
oServer est objet lrSer
```

Copier

```
oServer:SetAuthenticationCheckFunction(fnAuth)
```

```
PROCEDURE INTERNE  
fnAuth(pRequest objet  
lrRequest) : booléen  
    sKey est chaîne =  
pRequest:GetHeaderValue("X-  
API-KEY")  
    RENVoyer  
    (sKey="MY_SECRET_KEY")  
FIN
```

Contrôle du serveur

  **Start**

 **Présentation**

Démarre le serveur LigthREST

 **Prototypage**

Start() : (booléen, chaîne)

 **Paramètres**

(aucun)

 **Valeurs retournées**

Ty pe	Usage
----------	-------

bo olé en	Vrai si démarrage réussi.
-----------------	---------------------------

ch aîn e	Message d'erreur (vide si succès).
----------------	------------------------------------

Exemple

- Démarrage d'un serveur LightREST sur le port 9876
- Création d'une route /ping qui renvoie l'heure système et l'hôte du client REST

```
oServer est objet lrSer
oRoute  est objet lrRoute
bOK     est booléen
sErr    est chaîne

oServer:IPAndPort =
"127.0.0.1:9876"

oRoute.Method =
lrRoute::MethodGET
oRoute.Route  = "/ping"
oRoute.Handler = piPing

(bOK, sErr) =
oServer:AddRoute(oRoute)
SI PAS bOK ALORS
```

[Copier](#)

```

        Erreur("Erreur lors de
la création de la route : ",
sErr)
        RETOUR
    FIN

    (bOK, sErr) =
oServer:Start()
    SI PAS bOK ALORS
        Erreur("Erreur lors du
démarrage : ", sErr)
        RETOUR
    FIN

    Info("Call
http://[%oServer:IPAndPort%]
[%oRoute:Route%]",
CRLF+"Click on OK to close
LightREST server")

    oServer:Terminate()

    //////////////////////////////////
    //////////////////////////////////

    PROCEDURE INTERNE
    piPing(pRequest objet
lrRequest) : objet
lrResponse
        oResp est objet
lrResponse

        oResp.Body =
"Sending pong to host " +
pRequest.Host + " at " +
DateVersChaîne(DateHeureSys(
))
        oResp.ContentType =
lrResponse::ContentTXT
        oResp.Status =
lrResponse::StatusOK

        RENVOYER oResp
    FIN

```

  Kill

 **Présentation**

Arrêt forcé du serveur.

 Les routes REST en cours d'exécution sont interrompues abruptement. Pour un arrêt "propre", préférer la méthode `:Terminate()`.

 **Prototypage**

```
Kill() : (booléen, chaîne)
```

 **Paramètres**

(aucun)

 **Valeurs retournées**

Ty pe	Usage
bo olé en	Vrai si arrêt forcé réussi.
ch aîn e	Message d'erreur (vide si succès).

 Exemple

- Arrêt abrupt du serveur
- À utiliser en dernier recours (debug / shutdown forcé)

```
oSrv est objet lrServer
bOK est booléen
sErr est chaîne

(bOK, sErr) = oSrv:Kill()
SI PAS bOK ALORS

ExceptionDeclenche(exception
Avertissement, sErr)
FIN
```

Copier **Terminate** **Présentation**

Arrêt propre du serveur.

 Les routes REST en cours d'exécution sont complétées avant arrêt (ou timeout). Pour un arrêt immédiat, utiliser la méthode `:Kill()`

 Prototypage

```
Terminate(pTimeout Durée = :Req
```

 Paramètres

No m	T y p e	Usage
ptimeout	Durée	Durée maximale d'attente.

 Valeurs retournées

Ty pe	Usage
booléen	Vrai si arrêt réussi.
chaîne	Message d'erreur (vide si succès).

 Exemple

- Arrêt propre du serveur (attente fin des

requêtes)

- Timeout = 10 secondes

```
oSrv est objet lrServer
bOK est booléen
sErr est chaîne

(bOK, sErr) =
  oSrv:Terminate(10s)

SI PAS bOK ALORS

ExceptionDeclenche(exception
Avertissement, sErr)
FIN
```

Copier



SetLoggerOn



Présentation

Active le logger interne et définit le fichier de log.



Prototypage

```
SetLoggerOn(pLoggerFile chaîne)
```



Paramètres

No	T	Usage
m	y	

**p
e**

pL	c	Chemin du fichier
og	h	de log (vide =>
ge	aî	fichier par défaut).
rF	n	
il	e	
e		

 Valeurs retournées

Ty pe	Usage
bo olé en	Vrai si activation OK.
ch aîn e	Message d'erreur (vide si succès).

 Exemple

- Activation du logger LightREST dans un fichier
- Utile pour diagnostiquer un problème en phase de debug

```
oServer est objet lrServer
bOK      est booléen
sErr     est chaîne

(bOK, sErr) =
oServer:SetLoggerOn("c:\temp\lr_log.txt")
SI PAS bOK ALORS
    Erreur("Logger not
    enabled: ", sErr)
FIN
```

Copier

SetLoggerOff

Présentation

Désactive le logger interne.

Prototypage

```
SetLoggerOff() : (booléen, chaîne)
```

Paramètres

(aucun)

Valeurs retournées

Type	Usage
------	-------

bo Vrai si désactivation OK.
olé
en

ch Message d'erreur (vide si
aîn succès).
e

Exemple

- Désactivation du logger LightREST
- À utiliser après une phase de debug pour retrouver les performances nominales

```
oServer est objet lrServer
bOK      est booléen
sErr     est chaîne

(bOK, sErr) =
oServer:SetLoggerOff()
SI PAS bOK ALORS
    Erreur("Logger not
disabled: ", sErr)
FIN
```

Copier

ID Cipherring

  CipherID

Présentation

Chiffre un identifiant de la base de données. Évite le pillage des données en utilisant par exemple une route `/ {id}` en incrémentant simplement une valeur.

La méthode `CipherID` transformera une simple valeur numérique en un long identifiant chiffré. Il devient alors compliqué voire impossible d'aspirer l'intégralité de la base de données sans connaître la clé de chiffrement.

Exemple avec une
même clé de
chiffrement :

L'ID 1 devient
03980608-5abe9581-
7fee65e9-194ae106
L'ID 2 devient
d3e8b3a7-94319945-
f7484a0d-aecf776c

 Pour chiffrer plusieurs identifiants il est beaucoup plus rapide d'appeler une fois `CipherIDs` sur un tableau que de boucler unitairement sur `CipherID`.

 Note

La clé de chiffrement des IDs est stockée dans le membre `:CipherringKey` et initialisée aléatoirement à l'instanciation de l'objet. Cela signifie qu'en cas de redémarrage du serveur, les identifiants chiffrés déjà transmis aux clients seront inexploitable. Pour éviter ceci, il suffira de donner une clé de chiffrement fixe au membre `:CipherringKey` (dans ce cas on peut imaginer un changement régulier du contenu de la clé, par exemple quotidiennement).

 Prototypage

```
CipherID(pIDName chaîne, pIDVal
```

 Paramètres

No m	T y p e	Usage
pI	c	Nom logique de

DN	h	l'identifiant. Utile pour discriminer les IDs identiques sur des tables différents. Généralement égal au nom de la table concernée.
am	aî	
e	n	
	e	

Ainsi l'ID 123 des tables CLIENT et USER ne seront pas chiffrés de façon identique.

pI	c	Valeur à chiffrer (identifiant unique ou autre valeur à dissimuler)
DV	h	
al	aî	
ue	n	
	e	

Valeurs retournées

Type	Usage
ch	Valeur chiffrée format
aî	xxxxxxxx-xxxxxxxx-
e	xxxxxxxx-xxxxxxxx

Exemple

- Récupération du nom d'un client passé dans l'URL (exemple *https://monapi.fr/client?name=dupont*)
- Envoie une erreur `StatusBadRequest` si le paramètre n'a pas été reçu
- Recherche de l'enregistrement correspondant (si KO, retour `StatusNotFound` au client)
- Conversion de l'enregistrement en Variant (méthode `IrResponse:RecordToVariant`)
- Chiffrement de l'ID de la base de données
- Envoi des données au format JSON

```
FUNCTION
ClientGETByName (pRequest
objet lrRequest, poResponse
objet lrResponse)

vClient    est variant
sName      est chaine =
pRequest:GetURLValue("name")

SI sName="" ALORS
    poResponse:Body          =
    "No value provided"
    poResponse:ContentType   =
    lrResponse::ContentTXT
    poResponse:Status        =
    lrResponse::StatusBadRequest
SINON
    SI
    hLitRecherchePremier("CLIENT
    ", "NAME", sName) ALORS
        vClient =
        lrResponse::RecordToVariant(
        "CLIENT",
```

[Copier](#)

```
"NAME, FIRSTNAME, ADRESS, ZIP, C
ITY")
```

```
vClient.ID =
oServer:CipherID("CLIENT",
CLIENT.ID)
```

```
poResponse:Body
= VariantVersJSON(vClient)
poResponse:ContentType
= lrResponse::ContentJSON
poResponse:Status
= lrResponse::StatusOK
SINON
poResponse:Status =
lrResponse::StatusNotFound
FIN
FIN
```



CipherIDs



Présentation

Chiffre un tableau d'identifiants de la base de données.



Pour chiffrer plusieurs identifiants il est beaucoup plus rapide d'appeler une fois CipherIDs sur un tableau que de boucler unitairement sur CipherID.



Prototypage

Syntaxe 1 : CipherIDs(pIDName c

Syntaxe 2 : CipherIDs(pIDName c



Paramètres

No m	T y p e	Usage
pI DN am e	c h a î n e	Nom logique de l'identifiant.
pI DV al ue s	t a bl e a u d e c h a î n e s	IDs à chiffrer (syntaxe 1).
pA rr ay	t a bl e a	Tableau d'éléments contenant l'ID (syntaxe 2).

u		
pC le ar Co l	c h aî n e	Nom du membre qui contient l'ID clair (syntaxe 2).
pC ip he re dC ol	c h aî n e	Nom du membre qui contiendra l'ID chiffré (syntaxe 2).

Valeurs retournées

Ty pe	Usage
tab lea u de ch aîn es (sy nta xe 1)	IDs chiffrés format xxxxxxxx-xxxxxxxx- xxxxxxxx-xxxxxxx.

Au
cu
n
(sy
nta
xe
2)

Le tableau pArray est
modifié sur place.

Exemple

- Chiffrement d'une liste d'IDs (optimisé par lot)
- Renvoi d'un tableau de chaînes chiffrées.

```
oServer est objet IrServer  
tIDs est tableau de  
chaînes  
tEnc est tableau de  
chaînes
```

```
tIDs.Ajoute("1")  
tIDs.Ajoute("2")  
tIDs.Ajoute("3")
```

```
tEnc =  
oServer:CipherIDs("CLIENT",  
tIDs)
```

Copier



 Présentation

Déchiffre un identifiant.

Prototypage

```
DecipherID(pIDName chaîne, pIDV
```

Paramètres

No m	T y p e	Usage
pI DN am e	c h aîne	Nom logique de l'identifiant.
pI DV al ue	c h aîne	Valeur chiffrée format xxxxxxxx- xxxxxxxx-xxxxxxx- xxxxxxx.

Valeurs retournées

Ty pe	Usage
----------	-------

ch Valeur déchiffrée (chaîne
 aîn vide si échec).
 e



Exemple

- Récupération d'un ID chiffré dans la route */client/{id}*
- Déchiffrement de l'ID
- Suppression de l'enregistrement

Copier

```

FONCTION
ClientDELETE(pRequest objet
lrRequest, poResponse objet
lrResponse)

sID      est chaine =
pRequest:GetRouteValue("id")

SI sID="" ALORS
    poResponse:Body      =
    "No ID provided"
    poResponse:ContentType =
    lrResponse::ContentTXT
    poResponse:Status     =
    lrResponse::StatusBadRequest
SINON
    sID =
    oServer:DecipherID("CLIENT",
sID)
    SI sID="" ALORS
        poResponse:Body
        = "illegal or expired ID
        provided"
        poResponse:ContentType
        = lrResponse::ContentTXT
        poResponse:Status
        =
        lrResponse::StatusBadRequest
    SINON
  
```

```

        SI
        hLitRecherchePremier("CLIENT
        ", "ID", sID) ALORS

        hSupprime("CLIENT")
            poResponse:Status
        = lrResponse::StatusOK
        SINON
            poResponse:Status
        = lrResponse::StatusNotFound
        FIN
    FIN
FIN

```



DecipherIDs


Présentation

Déchiffre plusieurs identifiants.


Prototypage

```

Syntaxe 1 : DecipherIDs (pIDName
Syntaxe 2 : DecipherIDs (pArray,

```


Paramètres

No m	T y p e	Usage

pI DN am e	c h aî n e	Nom logique de l'identifiant.
pI DV al ue s	t a bl e a u	Valeurs chiffrées (syntaxe 1).
pA rr ay	t a bl e a u	Tableau d'éléments contenant les IDs chiffrés (syntaxe 2).
pC ip he re dC ol	c h aî n e	Nom du membre contenant l'ID chiffré (syntaxe 2).
pC le ar Co l	c h aî n e	Nom du membre qui contiendra l'ID clair (syntaxe 2).

 Valeurs retournées

Ty pe	Usage
tab lea u de ch aîn es (sy nta xe 1)	Valeurs déchiffrées format xxxxxxxx-xxxxxxxx- xxxxxxxx-xxxxxxx.
Au cu n (sy nta xe 2)	Le tableau pArray est modifié sur place.

 Exemple

- Déchiffrement par lot d'un tableau d'IDs
- Utile quand tu reçois plusieurs IDs chiffrés dans une même requête

Copier

```

oServer est objet lrServer
tEnc est tableau de chaînes
tPlain est tableau de chaînes

tEnc.Ajoute("03980608-5abe9581-7fee65e9-194ae106")
tEnc.Ajoute("d3e8b3a7-94319945-f7484a0d-aecf776c")

tPlain =
  oServer:DecipherIDs("CLIENT", tEnc)

```

HTTP Headers

▼

SetGlobalHeader

📖

Présentation

Ajoute (ou remplace) un header HTTP global envoyé dans toutes les réponses,

⚙️

Prototypage

```
SetGlobalHeader(pHeader chaîne,
```

🔧

Paramètres

No	T	Usage
----	---	-------

m y
p
e

pH ea de r	c h a n e	Nom du header (ex: X-App-Version).
---------------------	-----------------------	---------------------------------------

pV al ue	c h a n e	Valeur.
----------------	-----------------------	---------

Valeurs retournées

(aucune)

Exemple

- Ajout d'un header global *X-App-Version* sur toutes les routes
- Utile pour le debug client et la traçabilité

```
oServer est objet lrSer  
oServer:SetGlobalHeader("X-  
App-Version", "LightREST  
v3")
```

Copier

 **SetGlobalHeaders**

 **Présentation**

Copie un ensemble de headers HTTP globaux (tableau associatif) vers le serveur.

 **Prototypage**

```
SetGlobalHeaders (pGlobalHeaders
```

 **Paramètres**

No m	T y p e	Usage
pG lo ba lH ea de rs	t a bl e a u a s s o c i	Tableau des headers globaux (Tag => Valeur).

a
ti
f
d
e
c
h
aî
n
e
s

Valeurs retournées

(aucune)

Exemple

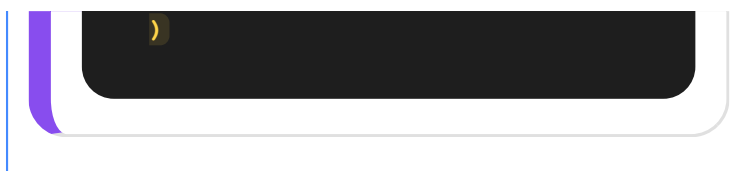
- Définition de plusieurs headers globaux en une seule opération
- Pratique pour centraliser version, cache, options d'application

```
oServer est objet IrServer  
tas      est tableau  
associatif de chaînes
```

```
tas["X-App-Version"] =  
"LightREST v3"  
tas["X-Server"]      = "CODE  
LINE"  
tas["Cache-Control"] = "no-  
store"
```

```
oServer:SetGlobalHeaders(tas
```

Copier



 **SetGlobalOptionsHeader**

 **Présentation**

Ajoute (ou remplace) un header global pour les réponses OPTIONS (CORS).

 **Prototypage**

```
SetGlobalOptionsHeader (pHeader
```

 **Paramètres**

No m	T y p e	Usage
pH ea de r	c h aî n e	Nom du header OPTIONS (ex: Access-Control- Allow-Origin).
pV al ue	c h aî	Valeur.

n
e

 **Valeurs retournées**

(aucune)

 **Exemple**

- Ajout d'un header CORS global pour les réponses OPTIONS
- Permet aux clients navigateur d'appeler l'API cross-domain

```
oServer est objet lrSer
oServer:SetGlobalOptionsHead
er("Access-Control-Allow-
Origin", "*")
```

Copier

  **SetGlobalOptionsHeaders**

 **Présentation**

Copie un ensemble de headers globaux pour OPTIONS (CORS).

 **Prototypage**

```
SetGlobalOptionsHeaders (pGlobal
```

 Paramètres

No m	T y p e	Usage
pG lo ba lo pt io ns He ad er s	t a bl e a u s o ci a ti f d e c h aî n e s	Tableau des headers OPTIONS globaux (Tag => Valeur).

 Valeurs retournées

(aucune)

Exemple

- Définition d'un jeu complet de headers CORS (OPTIONS)
- Autorise méthodes et headers personnalisés

oServer est objet lrServer
tasOpt est tableau
associatif de chaînes

Copier

```
tasOpt["Access-Control-  
Allow-Origin"] = "*"
tasOpt["Access-Control-  
Allow-Methods"] =  
"GET, POST, PUT, DELETE, OPTIONS"  
tasOpt["Access-Control-  
Allow-Headers"] = "Content-  
Type, X-API-KEY, SESSION_ID"
```

```
oServer:SetGlobalOptionsHead  
ers(tasOpt)
```

Certificats



SetHTTPSCertificate

Présentation

Définit le certificat SSL et la clé privée

(mode HTTPS_CERT).

Prototypage

Syntaxe 1 : SetHTTPSCertificate
Syntaxe 2 : SetHTTPSCertificate

Paramètres

No m	T y p e	Usage
pc er ti fi ca te	c h â n e / lr S S L C e rt ifi c a te	Certificat (chemin, contenu, ou structure lrSSLCertificate).

pP	c	Clé privée (syntaxe
ri	h	1).
va	aî	
te	n	
Ke	e	
y		

Valeurs retournées

(aucune)

Exemple

- Passage du serveur en HTTPS avec certificat fourni (HTTPS_CERT)
- Définition certificat + clé privée
- Démarrage du serveur

```
oServer est objet lrSer  
bOK      est booléen  
sErr     est chaîne
```

```
oServer:IPAndPort =  
"0.0.0.0:443"  
oServer:HTTPMode =  
lrServer::HTTPS_CERT
```

```
oServer:SetHTTPCertificate(  
fChargeTexte("c:\ssl\server.  
crt"),  
fChargeTexte("c:\ssl\server.  
key"))
```

```
(bOK, sErr) =  
oServer:Start()  
SI PAS bOK ALORS
```

[Copier](#)

```
Erreur("HTTPS start  
failed: ", sErr)  
FIN
```

SetLetsEncryptParameters

Présentation

Configure les paramètres Let's Encrypt
(mode HTTPS_LETS_ENCRYPT).

Prototypage

```
Syntaxe 1 : SetLetsEncryptParam  
Syntaxe 2 : SetLetsEncryptParam  
Syntaxe 3 : SetLetsEncryptParam
```

Paramètres

No m	T y p e	Usage
pD om ai n	c h aî n e	Domaine principal (syntaxe 1).

pDomains	taudenechaines	Liste de domaines (syntaxe 2).
ptEmail	chaine	Email Let's Encrypt (syntaxe 1 et 2).
ptCertificatePath	chaine	Chemin de stockage des certificats (syntaxe 1 et 2, optionnel).
ptRenewalTime	en	Délai de renouvellement (syntaxe 1 et 2, optionnel).

la r
y

pL	<u>lr</u>	Structure complète (syntaxe 3).
et	<u>L</u>	
sE	<u>e</u>	
nc	<u>ts</u>	
ry	<u>E</u>	
pt	<u>n</u>	
	<u>c</u>	
	<u>r</u>	
	<u>y</u>	
	<u>p</u>	
	<u>t</u>	

Valeurs retournées

(aucune)

Exemple

- Passage en HTTPS avec Let's Encrypt (certificats auto-renouvelés)
- Définition du domaine et de l'email
- Démarrage du serveur

```
oServer est objet lrSer
bOK      est booléen
sErr     est chaîne
stLets   est lrLetsEncrypt
```

```
oServer:IPAndPort =
"0.0.0.0:443"
```

Copier

```
oServer:HTTPMode =  
lrServer::HTTPS_LETS_ENCRYPT  
  
stLets.Domains.Ajoute("api.m  
ondomaine.fr")  
stLets.Domains.Ajoute("priva  
te.mondomaine.fr")  
stLets.Email =  
"admin@mondomaine.fr"  
stLets.CertificatePath =  
"./certificates"  
stLets.RenewDelay = 30  
  
oServer:SetLetsEncryptParam  
eters(stLets)  
  
(bOK, sErr) =  
oServer:Start()  
SI PAS bOK ALORS  
    Erreur("LetsEncrypt start  
failed: ", sErr)  
FIN
```

SetSelfCertificateParameters

Présentation

Configure les paramètres du certificat auto-généré (mode HTTPS_SELF).

Prototypage

```
SetSelfCertificateParameters (pl
```

Paramètres

No m	T y p e	Usage
pl rS el fC er ti fi ca te	lr S el f C e rt ifi c a te	Structure contenant les paramètres du certificat auto- génééré (ex: KeySize).

Valeurs retournées

(aucune)

Exemple

- Passage en HTTPS avec certificat auto-génééré (HTTPS_SELF)
- Définition de la taille de clé
- Démarrage du serveur

```
oServer est objet lrSer  
stSelf est  
lrSelfCertificate
```

[Copier](#)

```
bOK      est booléen
sErr     est chaîne

oServer:IPAndPort =
"0.0.0.0:8443"
oServer:HTTPMode  =
lrServer::HTTPS_SELF

stSelf.KeySize = 2048

oServer:SetSelfCertificateParameters(stSelf)

(bOK, sErr) =
oServer:Start()
SI PAS bOK ALORS
    Erreur("HTTPS self start
failed: ", sErr)
FIN
```