MiniPAVI

Passerelle minitel

Crée ton service Minitel

Minitel is still alive

and will never die ;-)

MiniPaviCli

Communication avec une passerelle MiniPavi

Librairie MiniPaviCli.php et protocole de communication

01/05/2024

www.minipavi.fr

https://github.com/ludosevilla/minipaviCli

Table des matières

Rappel de ce qu'est MiniPavi

Les quelques services Minitel encore existants sont accessibles généralement via websockets, sur le réseau internet.

Quelques-uns le sont également par téléphone.

Pour s'y connecter, on utilise soit un émulateur Minitel sur ordinateur, soit un Minitel avec un périphérique, connecté à sa prise péri-informatique, lui permettant d'utiliser le réseau internet pour communiquer, ou bien directement relié à une ligne fixe dans le cas des services accessibles par téléphone.

Le développement de tels services utilisant les websockets ou une ligne téléphonique n'est pas forcément à la portée de tous.

C'est là qu'intervient MiniPavi, un projet développé afin de faciliter la création de services Minitel, dans une perspective de sauvegarde du patrimoine numérique.

MiniPavi (Mini Point d'Accès Vidéotex) est une passerelle qui permet de développer simplement des services Minitel en technologie standard Serveur Web (type Apache) et PHP (ou tout autre langage tels que Python, etc.).

Ces services sont déportés, c'est-à-dire ne sont pas nécessairement installés sur le même hébergement que MiniPavi : n'importe quel hébergement web moderne dans le monde peut donc suffire.

Ainsi, l'utilisateur (le Minitel) se connecte et communique via websocket ou téléphone (VoIP/RTC) avec MiniPavi, et MiniPavi communique avec le service minitel via le protocole http (utilisé pour le Web).

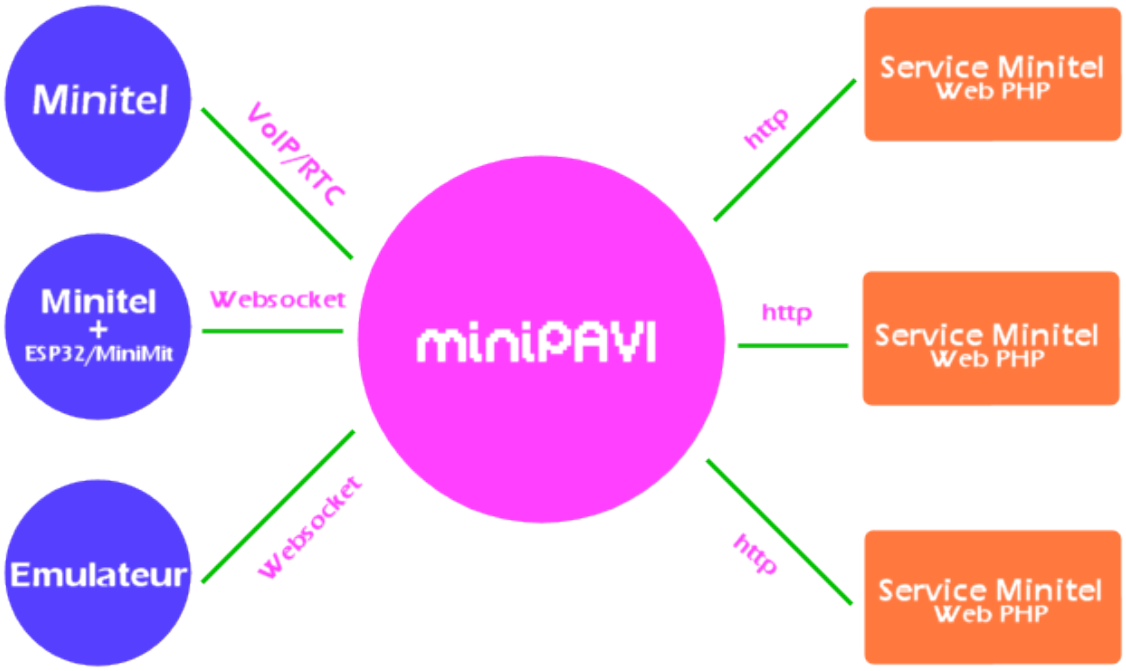
Dans la suite de ce document, le terme « passerelle » se réfère à une passerelle MiniPavi.

Le terme « service » se réfère à un service Minitel.

Ce document est divisé en deux parties distinctes :

- La description du protocole d'échange entre la passerelle et le service
- L'implémentation de ce protocole dans la librairie PHP « MiniPaviCli.php »

Le développement de services Minitel nécessite que vous ayez préalablement quelques connaissances du Videotex et de ses contraintes d'affichage.



- Le site du projet MiniPavi est : **http://www.minipavi.fr**
- L'accès à MiniPavi par émulateur est accessible sur : **https://www.minipavi.fr/emulminitel/index.php**
- Les adresses websockets de MiniPavi sont **ws://go.minipavi.fr:8182** et **wss://go.minipavi.fr:8181**
- L'accès par téléphone à MiniPavi se fait par le **09 72 10 17 21** (ou **00 33 9 72 10 17 21**)

A la date de rédaction de ce document, MiniPavi est également accessible par :

- Boitier MiniMit, accès par le choix 14 du Guide des services.
- Serveur « 3615 IUT Auxerre » au **03 58 43 51 50**
- Boitier Minitel-ESP32 de Iodeo (service préconfiguré)

Protocole d'échange

1 – Principes d'échanges entre une passerelle MiniPavi et un service Minitel

Les échanges sont effectués par requêtes http, toujours à l'initiative de la passerelle.

Par exemple, lors de la connexion d'un utilisateur, la passerelle envoie une requête http au service qui, en retour, lui indique les informations à afficher à l'utilisateur et, éventuellement, quelle saisie utilisateur est attendue.

Ensuite, lorsque l'utilisateur valide une saisie, alors une nouvelle requête http est émise de la part de la passerelle vers le service, et ainsi de suite.

Les requêtes émises par la passerelle envoient leurs données au format JSON en utilisant la méthode POST.

En réponse, le service renvoi des données également au format JSON.

Si vous souhaitez uniquement utiliser la librairie « MiniPaviCli.php », vous pouvez directement consulter la seconde partie du document (Page 19).

2 – Format des données émises par la passerelle vers le service

2.1 Définition :

Le contenu JSON des requêtes émises par la passerelle vers le service ont le format suivant :

```
{
	"PAVI":
	{
		"version":String,
		"uniqueId":String,
		"remoteAddr":String,
		"typesocket":String,
		"versionminitel":String,
		"content":Array,
		"context":String,
		"fctn":String
	},
	"URLPARAMS":Array
}
```

version	*Version de la passerelle*
uniqueId	*Identifiant unique de l'utilisateur connecté au service*
remoteAddr	*Adresse IP (ou téléphone si connu) de l'utilisateur connecté*
typeSocket	*Type de la connexion : websocket, websocketssl ou other*
versionminitel	*3 caractères correspondant au type de Minitel, si connu. Sinon « ??? ».*
content	*Tableau contenant la saisie de l'utilisateur. Si il s'agit d'une saisie de plusieurs lignes, chaque ligne est un élément du tableau. S'il n'y a qu'une seule ligne, la saisie est à l'indice « 0 » du tableau.*
context	*Données libres précédemment envoyées par le service. Sert à sauvegarder le contexte de l'utilisateur tout au long de sa visite du service.*
fctn	*Touche de fonction saisie, ou évènement, ayant initié cette requête.* *Valeurs possibles : ENVOI, SUITE, RETOUR, ANNULATION, CORRECTION, GUIDE, REPETITION, SOMMAIRE, CNX, FIN, DIRECT, DIRECTCNX, DIRECTCALLFAILED,*

	DIRECTCALLENDED, BGCALL, BGCALL-SIMU
URLPARAMS	*Tableau associatif contenant les éventuels paramètres indiqués dans l'Url.* *Clé inexistante si aucun paramètre n'est présent !*

2.2 Exemples de données JSON transmises par la passerelle :

Exemple de contenu JSON lors de la connexion d'un utilisateur à la passerelle via un émulateur Minitel sur le web et sa connexion à un service :

```
{
	"PAVI":
	{
		"version":"1.2",
		"uniqueId":"1714813976258",
		"remoteAddr":"82.65.112.8",
		"typesocket":"websocketssl",
		"versionminitel":"Cv;",
		"content":[],
		"context":"",
		"fctn":"CNX"
	}
}
```

Exemple de contenu JSON lors de la connexion d'un utilisateur à la passerelle via une ligne téléphonique et sa connexion à un service :

```
{
	"PAVI":
	{
		"version":"1.2",
		"uniqueId":"1714813977354",
		"remoteAddr":"0184257626",
		"typesocket":"other",
		"versionminitel":"Cz6",
		"content":[],
		"context":"",
		"fctn":"CNX"
	}
}
```

Exemple de contenu JSON lors de l'envoi d'une zone de saisie texte de 2 lignes par l'appui de la touche **Envoi** :

```
{
        "PAVI":
        {
                "version":"1.2",
                "uniqueId":"1714813977354",
                "remoteAddr":"0184257626",
                "typesocket":"other",
                "versionminitel":"Cz6",
                "content":["ligne 1","ligne 2"],
                "context":"",
                "fctn":"ENVOI"
        }
}
```

3 – Format des données émises par le service vers la passerelle

3.1 Définition :

Le contenu JSON des réponses du service à la passerelle ont le format suivant :

```
{
	"version":String,
	"content":String,
	"context":String,
	"echo":String,
	"next":String,
	"directcall":String,
	"COMMAND":Object
}
```

`version`	*Version du client*
`content`	*Le contenu de la page videotex à afficher, encodée en MIME Base 64*
`context`	*Données libres qui seront renvoyées inchangées par la suite par la passerelle*
`echo`	*Active l'echo par la passerelle des caractères tapés par l'utilisateur, pour que l'utilisateur voie ce qu'il tape.* *Valeurs possibles : « on » ou « off ».* *Généralement, cette clé aura la valeur « on »*
`next`	*Prochaine Url du service qui devra être appelée par la passerelle*
`directcall`	*Demande à la passerelle d'appeler immédiatement l'url indiquée par la clé « next », sans attendre une action de l'utilisateur.* *Valeurs possible : « no », « yes », « yes-cnx ».* *Si la valeur est « yes », l'appel au service aura la clé « fctn » à la valeur « DIRECT ».* *Si la valeur est « yes-cnx », l'appel au service aura la clé « fctn » à la valeur « DIRECTCNX ».*
`COMMAND`	*Commande particulière que doit gérer la passerelle (saisie texte, saisie message, etc.). Voir plus bas.*

3.2 Commandes de la clé « COMMAND » acceptées par la passerelle :

L'objet « COMMAND » a toujours une clé nommée « name » indiquant le nom de la commande à exécuter. Selon cette commande, d'autres paramètres doivent être fournis.

Commande « InputTxt »

Demande à la passerelle de gérer la saisie par l'utilisateur d'une seule ligne de saisie, de longueur définie. Généralement utilisée pour la saisie d'un choix.

```
{
	"name":"InputTxt",
	"param":{
	"x":Integer,
	"y":Integer,
	"l":Integer,
	"char":Char,
	"spacechar":Char,
	"prefill":String,
	"cursor":String,
	"validwith":Integer
}
```

`name`	*« InputTxt »*
`param->x`	*Position de la colonne (1-40) de la zone de saisie.*
`param->y`	*Position de la ligne (1-25) de la zone de saisie.*
`param->l`	*Longueur de la zone de saisie (1-40)*
`param->char`	*Si non vide, quel que soit la caractère tapé par l'utilisateur, ce caractère s'affichera (pour la saisie de mot de passe par exemple)*
`param->spacechar`	*Caractère pour affichage du champ de saisie (espace ou '.' généralement)*
`param->prefill`	*Valeur de pré-remplissage du champ de saisie*
`param->cursor`	*Si « on », le curseur sera visible, sinon « off »*
`param->validwith`	*Valeur indiquant les touches de fonctions possibles qui valideront la saisie (par exemple la touche Envoi). Les touches Correction et Annulation ne peuvent être définies car gérées directement par la passerelle pour la correction de la saisie par l'utilisateur.* *La valeur indiquée est la somme des valeurs des touches de fonctions possibles :* *SOMMAIRE = 1*

	RETOUR = 4 *REPETITION = 8* *GUIDE = 16* *SUITE = 64* *ENVOI = 128*

Commande « InputMsg »

Demande à la passerelle de gérer la saisie par l'utilisateur d'une zone de saisie de plusieurs lignes, de longueur définie. Généralement utilisée pour la saisie d'un message.

```
{
	"name":"InputMsg",
	"param":{
	"x":Integer,
	"y":Integer,
	"w":Integer,
	"h":Integer,
	"spacechar":Char,
	"prefill":Array,
	"cursor":String,
	"validwith":Integer
	}
}
```

`name`	*« InputMsg »*
`param->x`	*Position de la colonne (1-40) de la zone de saisie.*
`param->y`	*Position de la ligne (1-25) de la zone de saisie.*
`param->w`	*Longueur de la zone de saisie (1-40)*
`param->h`	*Hauteur (nombre de lignes) de la zone de saisie*
`param->prefill`	*Tableau contenant les valeurs de pré-remplissage de la zone de saisie. Chaque élément du tableau représente une ligne.*
`param->spacechar`	*Caractère pour affichage du champ de saisie (espace ou '.' généralement)*
`param->cursor`	*Si « on », le curseur sera visible, sinon « off »*
`param->validwith`	*Valeur indiquant les touches de fonctions possibles qui valideront la saisie (per exemple la touche Envoi). Les touches Correction et Annulation ne peuvent être définies car gérées directement par la passerelle pour la correction de la saisie par l'utilisateur. De même que les touches Suite et Retour gérées directement par la*

	passerelle pour le changement de ligne dans la zone de saisie. *La valeur indiquée est la somme des valeurs des touches de fonctions possibles :* *SOMMAIRE = 1* *REPETITION = 8* *GUIDE = 16* *ENVOI = 128*

Commande « libCnx »

Demande à la passerelle de déconnecter l'utilisateur du service. L'utilisateur retourne alors au service par défaut (généralement, l'accueil de la passerelle).

Cette commande n'a aucun paramètre.

```
{
	"name":"libCnx"
}
```

Commande « PushServiceMsg»

Demande à la passerelle de d'afficher un message en ligne « 0 » aux autres utilisateurs.

```
{
	"name":"PushServiceMsg",
	"param":{
	"uniqueids":Array,
	"message":Array
	}
}
```

`name`	*« PushServiceMsg »*
`param->uniqueid`	*Tableau contenant la liste des identifiants uniques des utilisateurs vers lesquels envoyer un message.*
`param->message`	*Tableau contenant les messages à envoyer à chaque utilisateur. Le message peut donc être différent pour chacun d'entre eux.*

Commande « BackgroundCall»

Demande à la passerelle d'effectuer un appel à une url à l'heure indiquée.

```
{
	"name":"BackgroundCall",
	"param":{
	"time":Integer,
```

```
        "simulateUser":Bool,
        "url":String,
        "uniqueId":Array

        }
}
```

name	*« PushServiceMsg »*
param->time	*Timestamp Unix de l'heure prévue de l'appel*
param->simulate	*Si « false », l'appel sera effectué vers l'url indiquée en paramètres. Cet appel devra être vu par le service comme indépendant de l'action d'un utilisateur. La touche de fonction indiquée dans la clé « fctn » aura la valeur « BGCALL ». En retour, le service ne pourra qu'envoyer une commande « PushServiceMsg » à la passerelle.* *Si « true », l'appel sera effectué vers l'url qui a été indiquée dans la clé « nexturl » de l'utilisateur, avec en contenu »saisie » la valeur du paramètre « url » (ci-dessous). La touche de fonction indiquée dans la clé « fctn » aura la valeur « BGCALL-SIMU ». Cet appel devra être vu par le service comme une action de l'utilisateur. En retour, le service peut envoyer toutes commandes et tout contenu, qui sera alors envoyé à l'utilisateur.*
param->uniqueid	*Tableau contenant la liste des identifiants uniques des utilisateurs.* *Si « simulate »est « true », l'url appelée sera celle indiquée dans la clé « nexturl » de l'utilisateur identifié.* *Dans tous les cas, l'identifiant unique sera indiqué dans la clé « uniqueId » de l'appel de la passerelle vers le service.*
param->url	*Si « simulate » est « false », indique l'url qui doit être appelée.* *Si « simulate » est « true », indique les données qui seront indiqué dans la clé « content » de l'appel de la passerelle au service.*

Commande « connectToWs»

Demande à la passerelle de connecter l'utilisateur à un service Minitel accessible par Websocket.

```
{
	"name":"connectToWs",
	"param":{
	"key":String,
	"host":String,
	"path":String,
	"echo":String,
	"case":String,
	"proto":String
	}
}
```

`name`	*« connectToWs »*
`key`	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
`param->host`	*Adresse et port du serveur (Exemple : mntl.joher.com:2018)*
`param->path`	*Eventuel chemin d'accès. Par défaut « / »*
`param->proto`	*Eventuel protocole à utiliser. Vide par défaut.*
`param->echo`	*« on » : l'echo est activé et géré par la passerelle.* *« off » : l'echo est géré directement par le serveur.*
`param->case`	*« lower » : force le clavier de l'utilisateur en minuscules.* *« upper » : force le clavier de l'utilisateur en majuscules.*

En fin de connexion, l'url indiquée dans la clé « nexturl »de la requête sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

Commande « connectToTln»

Demande à la passerelle de connecter l'utilisateur à un service Minitel accessible par Telnet.

```
{
	"name":"connectToTln",
```

```
        "param":{
        "key":String,
        "host":String,
        "echo":String,
        "case":String
        }
}
```

name	*« connectToWs »*
key	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
param->host	*Adresse et port du serveur)*
param->echo	*« on » : l'echo est activé et géré par la passerelle.* *« off » : l'echo est géré directement par le serveur.*
param->case	*« lower » : force le clavier de l'utilisateur en minuscules.* *« upper » : force le clavier de l'utilisateur en majuscules.*

En fin de connexion, l'url indiquée dans la clé « nexturl »de la requête sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

Commande « connectToExt»

Demande à la passerelle de connecter l'utilisateur à un service Minitel accessible par téléphone.

```
{
        "name":"connectToTln",
        "param":{
        "key":String,
        "number":String,
        "RX":Integer,
        "TX":Integer
        }
}
```

name	*« connectToWs »*

`param->key`	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
`param->number`	*Numéro d'appel du serveur*
`param->RX`	*Niveau en Décibels minimal en réception (Ex : -35)*
`param->TX`	*Niveau en Décibels du signal transmis (Ex : -30)*

En fin de connexion, l'url indiquée dans la clé « nexturl »de la requête sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

Commande « duplicateStream»

Demande à la passerelle de connecter l'utilisateur A au flux transmis à un autre utilisateur B (l'utilisateur A voit ce que voit l'utilisateur B)

```
{
        "name":"duplicateStream",
        "param":{
        "key":String,
        "uniqueid":String
        }
}
```

`name`	*« duplicateStream »*
`param->key`	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
`param->uniqueid`	*Identifiant unique de l'utilisateur dont le flux sortant doit être dupliqué.*

En fin de connexion, l'url indiquée dans la clé « nexturl »de la requête sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

3.3 Exemples de données JSON transmises par le service :

Exemple de contenu JSON envoyé par le service à la passerelle pour l'affichage d'une page et l'initialisation d'une zone de saisie de 2 lignes de 40 caractères en position X=1 et Y = 13, curseur allumé avec validation par les touches **Répétition** **Guide** et **Envoi** :

```
{
"version":"1.0",
"content":"H0BBICASZh9BQQwUGzppRRs7YFhRGz…(reste de l'encodage en Base
64)",
"context":"a:1:{s:3:\"url\";s:0:\"\";}",
"echo":"on",
"directcall":"no",
"next":"http:\/\/www.monsite.fr\/index.php?step=20",
"COMMAND":{
        "name":"InputMsg",
        "param":{
                "x":1,
                "y":13,
                "w":40,
                "h":2,
                "spacechar":".",
                "prefill":[],
                "cursor":"on",
                "validwith":152
                }
        }
}
```

Note: la valeur de la clé "context" correspond ici à un tableau PHP « sérialisé ».

Exemple de contenu JSON envoyé par le service à la passerelle pour l'affichage d'une page et l'initialisation d'une zone de saisie de 10 caractères en position X=2 et Y = 20, curseur allumé avec validation par les touches **Répétition** et **Envoi** :

```
{
"version":"1.0",
"content":"H0BBIC… (reste de l'encodage en Base 64)",
"context":"a:0:{}",
"echo":"on",
"directcall":"no",
"next":"http:\/\/www.monsite.fr\/index.php?step=20",
"COMMAND":{
        "name":"InputTxt",
        "param":{
                "x":2,
                "y":20,
                "l":10,
                "char":"",
                "spacechar":".",
                "prefill":"Salut!",
                "cursor":"on",
```

```
                "validwith":136
                }
        }
}
```

Note: la valeur de la clé “context” correspond ici à un tableau PHP « sérialisé ».

Librairie PHP « MiniPaviCli.php »

La librairie « MiniPaviCli.php » est une implémentation du protocole défini ci-dessus en PHP, ainsi que la fourniture de quelques fonctions utilitaires en relation avec le videotex.

Il s'agit d'une classe dont les propriétés et méthodes sont statiques.

La version PHP utilisée est la 8.2 (les versions inférieures n'ont pas été testées, mais devraient fonctionner au moins depuis la version 7).

Le schéma général d'un script utilisant cette librairie sera le suivant :

- Appeler la méthode `start()` en début de script
- Gérer la saisie utilisateur et créer le contenu videotex à afficher à l'utilisateur
- Créer une commande à destination de la passerelle avec l'une des méthodes `create…()`
- Envoyer une réponse à la passerelle avec la méthode `send()`

4 – Propriétés

`uniqueId`	`String`	*Identifiant unique de la connexion*
`remoteAddr`	`String`	*IP de l'utilisateur ou numéro de téléphone si accès par téléphone (et numéro identifié)*
`content`	`Array`	*Contenu saisi. Chaque élément du tableau correspond à une ligne de saisie dans la cas d'une saisie de message.*
`fctn`	`String`	*Touche de fonction utilisée ou évènement ayant provoqué cette requête.* *Valeurs possibles : ENVOI, SUITE, RETOUR, ANNULATION, CORRECTION, GUIDE, REPETITION, SOMMAIRE, CNX, FIN, DIRECT, DIRECTCNX, DIRECTCALLFAILED, DIRECTCALLENDED, BGCALL, BGCALL-SIMU*

urlParams	Array	*Paramètres fournis lors de l'appel à l'url du service*
context	String	*Données libres identiques à celle indiqué lors de la précédente réponse du service à la passerelle*
typeSocket	String	*Type de connexion ('websocket', 'websocketssl' ou 'other')*
versionMinitel	String	*Version Minitel si connue (3 caractères), sinon '???'*

5– Méthodes

```
start () : void
```

Fonction devant être appelée en début de script.

Initialise les propriétés de la classe selon les données JSON envoyées par la passerelle.

```
send ($content, $next, $context='', $echo=true, $cmd=null, $directCall=false) : void
```

Crée et envoie une réponse au format JSON destinée à la passerelle

`content`	`String`	*Contenu videotex de la page à afficher*
`next`	`String`	*Url que la passerelle doit appeler lors de la prochaine action de l'utilisateur ou du prochain évènement.*
`context`	`String`	*Données libres qui seront renvoyées lors de la prochaine requête de la passerelle vers le service.*
`echo`	`Bool`	*« true » si l'echo des caractères doit être actif*
`cmd`	`Array`	*Commande à destination de la passerelle. Ces commandes sont créées par les méthodes décrites plus loin.*
`directCall`		*« yes » : l'url indiquée dans $next est appelée immédiatement, sans attente d'action de l'utilisateur et la valeur de la propriété $fctn sera alors « DIRECT » lors de ce prochain appel.* *« yes-cnx » : équivalent à « yes », mais la valeur de $fctn sera « DIRECTCNX »*

```
createInputTxtCmd ($posX=1, $posY=1, $length=1,
$validWith=MSK_ENVOI, $cursor=true, $spaceChar=' ', $char='',
$preFill='') : Array
```

Crée la commande « InputText » de demande à la passerelle de gérer la saisie par l'utilisateur d'une seule ligne de saisie, de longueur définie. Généralement utilisée pour la saisie d'un choix.

Une requête vers le service ne sera effectuée qu'après appui par l'utilisateur sur l'une des touches de fonction indiquée dans le paramètre « validWith ».

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

posX	Int	*Position de la colonne (1-40) de la zone de saisie.*
posY	Int	*Position de la ligne (1-25) de la zone de saisie.*
length	Int	*Longueur de la zone de saisie (1-40)*
validWith	Int	*Valeur indiquant les touches de fonctions possibles qui valideront la saisie (par exemple la touche Envoi). Les touches Correction et Annulation ne peuvent être définies car gérées directement par la passerelle pour la correction de la saisie par l'utilisateur.* *La valeur indiquée est la somme des valeurs des touches de fonctions possibles :* *SOMMAIRE = MSK_SOMMAIRE* *RETOUR = MSK_RETOUR* *REPETITION = MSK_REPETITION* *GUIDE = MSK_GUIDE* *SUITE = MSK_SUITE* *ENVOI = MSK_ENVOI*

cursor	Bool	*Si « true », le curseur sera visible*
spaceChar	String	*Caractère pour affichage du champ de saisie (espace ou '.' généralement)*
char	String	*Si non vide, quel que soit le caractère tapé par l'utilisateur, ce caractère s'affichera (pour la saisie de mot de passe par exemple)*
preFill	String	*Valeur de pré-remplissage du champ de saisie*

createInputMsgCmd ($posX=1, $posY=1,$width=1, $height=1, $validWith=MSK_ENVOI, $cursor=true, $spaceChar=' ', $preFill=array()): Array

Crée la commande « InputMsg » de demande à la passerelle de gérer la saisie par l'utilisateur d'une zone de texte de plusieurs lignes, de hauteur et de largeur définie. Généralement utilisée pour la saisie d'un message.

Une requête vers le service ne sera effectuée qu'après appui par l'utilisateur sur l'une des touches de fonction indiquée dans le paramètre « validWith ».

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

posX	Int	*Position de la colonne (1-40) de la zone de saisie.*
posY	Int	*Position de la ligne (1-25) de la zone de saisie.*
width	Int	*Largeur de la zone de saisie (1-40)*
height	Int	*Hauteur (nombre de lignes) de la zone de saisie*
validWith	Int	*Valeur indiquant les touches de fonctions possibles qui valideront la saisie (par exemple la touche Envoi). Les touches Correction et Annulation ne peuvent être définies car gérées directement par la passerelle*

		pour la correction de la saisie par l'utilisateur. De même que les touches Suite et Retour gérées directement par la passerelle pour le changement de lignes dans la zone de saisie. *La valeur indiquée est la somme des valeurs des touches de fonctions possibles :* *SOMMAIRE = MSK_SOMMAIRE* *REPETITION = MSK_REPETITION* *GUIDE = MSK_GUIDE* *ENVOI = MSK_ENVOI*
`cursor`	`Bool`	*Si « true », le curseur sera visible*
`spaceChar`	`String`	*Caractère pour affichage du champ de saisie (espace ou '.' généralement)*
`preFill`	`Array`	*Tableau contenant les valeurs de pré-remplissage de la zone de saisie. Chaque élément du tableau représente une ligne.*

`createPushServiceMsgCmd ($tMessage=array(), $tUniqueId=array()): Array`

Crée la commande « PushServiceMsg » de demande d'envoi d'un message en ligne « 0 » à d'autres utilisateurs.

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

`tMessage`	`Array`	*Messages qui doivent être envoyés. Maximum 40 caractères par message.*
`tUniqueId`	`Array`	*Identifiants uniques des utilisateurs à qui envoyer un message.*

Il y a un message par utilisateur : chaque message dans le tableau tMessage correspond au message devant être envoyé à l'utilisateur identifié dans le tableau tUniqueId localisé au même indice.

createBackgroundCallCmd ($tUrl=array(), $tTime=array(), $tUniqueId=array(), $tSimulate=array()): Array

Crée la commande « BackgroundCall » de demande d'appel par la passerelle d'une url en différé.

Plusieurs appels peuvent être programmés : un appel défini pour chaque indice des tableaux passés en paramètres.

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

tUrl	Array	*Pour chaque élément du tableau :* *Si « tSimulate » est « false », indique l'url qui doit être appelée.* *Si «tSimulate » est « true », indique les données qui seront indiqué dans la propriété « content » lors de l'appel de la passerelle au service.*
tTime	Array	*Pour chaque élément du tableau :* *Timestamp Unix de l'heure prévue de l'appel*
tUniqueId	Array	*Pour chaque élément du tableau :* *Si « tSimulate »est « false », l'url appelée sera celle indiquée par le paramètre tUrl.*

		Si « tSimulate »est « true », l'url appelée sera celle indiquée dans la clé « nexturl » de l'utilisateur identifié par son identifiant unique. *Dans tous les cas, l'identifiant unique sera indiqué dans la propriété « uniqueId » lors de l'appel de la passerelle vers le service.*
`tSimulate`	`Array`	*Pour chaque élément du tableau :* *Si « false », l'appel sera effectué vers l'url indiquée par le paramètre « tUrl ». Cet appel devra être vu par le service comme indépendant de l'action d'un utilisateur.* *La touche de fonction indiquée dans la propriété « fctn » aura la valeur « BGCALL ».* *En retour, le service ne pourra qu'envoyer éventuellement une commande « PushServiceMsg » à la passerelle.* *Si « true », l'appel sera effectué vers l'url qui aura été indiquée dans le paramètre « nexturl » lors du dernier appel de la fonction send() de l'utilisateur référencé par le paramètre « tUniqueId ».* *Lors de cet appel, le contenu « saisi » (propriété « content ») aura la valeur du paramètre « tUrl ».* *La touche de fonction*

		indiquée dans la propriété « fctn » aura la valeur « BGCALL-SIMU ». Cet appel devra être vu par le service comme une action de l'utilisateur. En retour, le service peut envoyer toutes commandes et tout contenu, qui sera alors envoyé à l'utilisateur.

createConnectToExtCmd ($number, $RX=-35, $TX=-18, $key='') : Array

Crée la commande « ConnectToExt » de connexion de l'utilisateur à un service Minitel accessible par téléphone.

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

key	*String*	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
number	*String*	*Numéro d'appel du serveur*
RX	*Int*	*Niveau en Décibels minimal en réception (Ex : -35)*
TX	*Int*	*Niveau en Décibels du signal transmis (Ex : -30)*

En fin de connexion, l'url indiquée dans le paramètre « nexturl »de la fonction send() sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

createConnectToTlnCmd ($host, $echo='off', $case='lower', $key='') : Array

Crée la commande « ConnectToTln » de connexion de l'utilisateur à un service Minitel accessible par Telnet.

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

key	*String*	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
host	*String*	*Adresse et port du serveur*
echo	*String*	*« on » : l'echo est activé et géré par la passerelle.* *« off » : l'echo est géré directement par le serveur.*
case	*String*	*« lower » : force le clavier de l'utilisateur en minuscules.* *« upper » : force le clavier de l'utilisateur en majuscules.*

En fin de connexion, l'url indiquée dans le paramètre« nexturl » de la fonction send() sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

```
createConnectToWsCmd ($host, $path='/', $echo='off',
$case='lower', $proto='', $key='') : Array
```

Crée la commande « ConnectToWs » de connexion de l'utilisateur à un service Minitel accessible par Websocket.

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

key	*String*	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
host	*String*	*Adresse et port du serveur (Exemple : mntl.joher.com:2018)*
path	*String*	*Eventuel chemin d'accès. Par défaut « / »*
proto	*String*	*Eventuel protocole à utiliser. Vide par défaut.*

echo	*String*	*« on » : l'echo est activé et géré par la passerelle.* *« off » : l'echo est géré directement par le serveur.*
case	*String*	*« lower » : force le clavier de l'utilisateur en minuscules.* *« upper » : force le clavier de l'utilisateur en majuscules.*

En fin de connexion, l'url indiquée dans la paramètre « nexturl »de la fonction send() sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

createDuplicateStream ($uniqueid, $key='') : Array

Crée la commande « DuplicateStream » de connexion de l'utilisateur A au flux transmis à un autre utilisateur B (l'utilisateur A voit ce que voit l'utilisateur B).

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

key	*String*	*Clé d'autorisation d'utilisation de cette commande (configurée au niveau de la passerelle)*
uniqueid	*String*	*Identifiant unique de l'utilisateur dont le flux sortant doit être dupliqué.*

En fin de connexion, l'url indiquée dans le paramètre « nexturl »de la fonction send() sera appelée et la touche de fonction indiquée sera « DIRECTCALLENDED » si la connexion s'est terminée normalement ou « DIRECTCALLFAILED » si la connexion a échouée.

L'utilisateur peut mettre fin à la connexion par la séquence *** + **Sommaire** ou par la touche **Connexion/fin**

createLibCnxCmd() : Array

Crée la commande « LibCnx » de déconnexion l'utilisateur du service. L'utilisateur retourne alors au service par défaut (généralement, l'accueil de la passerelle).

Retourne un tableau qui servira d'argument $cmd dans la fonction send().

setPos ($col, $line) : String

Retourne le code videotex correspondant au positionnement du curseur sur l'écran aux coordonnées X (col) et Y(line)

col	Int	*Colonne, entre 1 et 40*
line	Int	*Ligne, entre 0 et 24*

writeLine0 ($txt, $blink=false) : String

Retourne le code videotex correspondant à l'affichage d'un message en ligne « 0 » et retour à la position d'origine du curseur.

txt	String	*Message à afficher*
blink	Bool	*Si « true », le message clignotera*

clearScreen () : String

Retourne le code videotex correspondant à l'effacement de l'écran.

repeatChar ($char, $num) : String

Retourne le code videotex correspondant à la répétition d'un caractère un certain nombre de fois.

char	String	*Caractère à afficher*
num	Int	*Nombre de répétition. Maximum 63.*

writeCentered ($line, $text, $attr=''): String

Retourne le code videotex correspondant à l'affichage centré d'une ligne de texte.

text	String	*Texte à afficher (maximum 40 caractères)*
line	Int	*Numéro de la ligne, entre 0 et 24.*
attr	String	*Attrributs videotext à appliquer avant l'affichage de la ligne. Optionnel.*

toG2 ($str) : String

Retourne le code videotex correspondant à la conversion d'une chaîne au jeu de caractères G2 : nécessaire pour l'affichage de caractères accentués et spéciaux.

str	String	*La chaîne à convertir*

6 – Appel du service réalisé

Le service Minitel que vous avez réalisé est accessible de différentes manières :

- Depuis l'accueil MiniPavi, saisir directement l'url du script de votre service, par exemple : **http://www.monsite.fr/monservice.php**

- Créer un « compte créateur » sur MiniPavi et définir un code d'accès direct au service (par exemple `1234*MONSERVICE`) qui pourra être saisi depuis l'accueil MiniPavi. Lors de la définition du code d'accès, vous devrez indiquer l'URL du script de votre service, par exemple : **http://www.monsite.fr/monservice.php**

- Directement depuis un navigateur Web en saisissant l'adresse de l'émulateur avec, en paramètre, l'adresse du script de votre service, par exemple : `https://www.minipavi.fr/emulminitel/index.php?url=`**http://www.monsite.fr/monservice.php**

- Depuis tout autre émulateur ou interface nécessitant une connexion websocket via l'accès websocket MiniPavi, avec, par exemple, l'adresse suivante : `wss://go.minipavi.fr:8181/?url=`**http://www.monsite.fr/monservice.php**

- Si vous avez réalisé votre service en utilisant la librairie « MiniPaviCli.php », votre service peut également être appelé directement pas son url, par exemple : **http://www.monsite.fr/monservice.php**

7 - Exemple

L'exemple suivant est un service très simple qui affiche une page videotex et qui attend que l'utilisateur appuie sur **Suite**.

S'il appuie sur une autre touche de fonction, un message en haut de l'écran lui rappelle d'appuyer sur **Suite**.

Après son appui sur **Suite**, l'utilisateur peut saisir un texte de 20 caractères.

Les touches de fonction acceptées sont alors **Envoi** et **Sommaire**.

Si l'utilisateur appuie sur **Sommaire**, il retourne au début du service.

S'il appuie sur **Suite**, le message saisi est réaffiché.

Enfin, à l'appui de n'importe quelle touche de fonction, l'utilisateur retourne à la saisie du texte.

Ce script est installé à l'adresse suivante : http://www.minipavi.fr/test/test.php

La page videotex utilisée est téléchargeable ici : http://www.minipavi.fr/test/mapage.vdt

```
<?php
require "MiniPaviCli.php";    // Inclusion de la librairie

try {
	MiniPavi\MiniPaviCli::start();


	if (MiniPavi\MiniPaviCli::$fctn == 'CNX' || MiniPavi\MiniPaviCli::$fctn ==
'DIRECTCNX') {
		// C'est une nouvelle connexion.
		// Ici, vous pouvez initialiser ce que vous souhaitez

		// Par exemple, un contexte utilisateur contenant une variable "step" avec la
		// valeur "accueil"
		// Cette variables nous servira pour savoir quelle partie du script exécuter.

		// Toutes les variables du contexte seront dans un tableau associatif
		//clé/valeur qui sera ensuite "serializé" avec
		// la fonction serialize() de PHP

		$context = array('step'=>'accueil');

	} else {
		// Ce n'est pas une nouvelle connexion
		// Une touche de fonction a été saisie ou un évènement a eu lieu

		if (MiniPavi\MiniPaviCli::$fctn == 'FIN') {

			// C'est la déconnexion de l'utilisateur
			// On peut en profiter pour effectuer des tâches nécessaires
			// lors de la déconnexion
			// Attention: cet évènement peut être appelé plusieurs fois de
			//suite

			exit;
		}


		// On récupère le contacte utilisateur.
```

```
		$context = unserialize(MiniPavi\MiniPaviCli::$context);

		// On récupère la touche de fonction

		$fctn = MiniPavi\MiniPaviCli::$fctn;

		// On récupère la saisie utilisateur

		$content = MiniPavi\MiniPaviCli::$content;

	}

	// On initialise quelques variables

	$vdt=''; // Le contenu videotex à envoyer au Minitel de l'utilisateur
	$cmd=null; // La commande à exécuter au niveau de MiniPAVI. Par défaut, aucune.


	while(true) {

		// On execute la partie du script qui correspond à la valeur de la variable
		// "step"

		switch ($context['step']) {
			case 'accueil':
				// Accueil du service : on affiche une page et on attend que
				//l'utilisateur tape sur SUITE

				// On remplit la variable $vdt au fur et à mesure

				// Effacement de l'écran et suppression de l'echo local du
				// Minitel au cas où
				$vdt =
MiniPavi\MiniPaviCli::clearScreen().PRO_MIN.PRO_LOCALECHO_OFF;

				// Récupération du contenu d'une page videotext, que l'on ajoute
				//à l'effacement d'écran précédent
				// Le fichier de la page doit exister et être lisible depuis ce
				//script

				$vdt.= file_get_contents('mapage.vdt');

				// On affiche la date et l'heure en ligne 10, colonne 5,
				// en Rouge

				$vdt.=MiniPavi\MiniPaviCli::setPos(5,10);
				$vdt.=VDT_TXTRED.date('d/m/Y H:i');

				// Lors du prochain appel du script par la passerelle, on
				// exécutera la partie 'accueil-traitement-saisie'
				// pour traiter la saisie de l'utilisateur

				$context['step'] = 'accueil-traitement-saisie';

				// On attend une saisie utilisateur : on ne rappelle pas le
				// script immédiatement

				$directCall=false;

				break 2;	// On sort du bloc "switch" et "while"


			case 'accueil-traitement-saisie':

				// On n'accepte qu'un appui sur la touche SUITE...

				if ($fctn == 'SUITE') {

					// L'utilisateur a appuyé sur 'SUITE'.
					// On veut maintenant exécuter la partie du script 'menu'

					$context['step'] = 'menu';

					// On continue le script dans sa partie 'menu': on sort
					// du bloc 'switch', mais on reste dans le bloc 'while'
```

```
                                        break;

                                }


                                // sinon, on lui affiche un message en ligne "0"
                                // et on attend de nouveau qu'il appuies sur "SUITE"

                                $vdt = MiniPavi\MiniPaviCli::writeLine0('Tapez sur Suite !');

                                // On ne modifie pas la valeur de 'step', qui a déjà la valeur
                                // 'accueil-traitement-saisie'

                                // On attend une saisie utilisateur : on ne rappelle pas le
                                // script immédiatement

                                $directCall=false;

                                break 2;       // On sort du bloc "switch" et "while"

                        case 'menu':

                                // Effacement de l'écran
                                $vdt = MiniPavi\MiniPaviCli::clearScreen();

                                // On positionne le curseur ligne 24, colonne 1
                                // et on attend que l'utilisateur saisisse quelque chose de
                                // maximum 20 caractères.
                                // La zone de saisie est représentée par des '.'
                                // L'utilisateur pourra valider sa saisie avec les touches ENVOI
                                // ou SOMMAIRE
                                // On utilise ici la fonction "createInputTxtCmd"

                                $cmd =
MiniPavi\MiniPaviCli::createInputTxtCmd(1,24,20,MSK_ENVOI|MSK_SOMMAIRE,true,'.','');

                                // Lors du prochain appel du script par la passerelle, on
                                // executera la partie 'menu-saisie'
                                // pour traiter la saisie de l'utilisateur

                                $context['step'] = 'menu-saisie';

                                // On attend une saisie utilisateur : on ne rappelle pas le
                                //script immédiatement

                                $directCall=false;

                                break 2;       // On sort du bloc "switch" et "while"

                        case 'menu-saisie':

                                // L'utilisateur a donc validé sa saisie avec ENVOI ou SOMMAIRE
                                // Si c'est ENVOI : on lui affiche ce qu'il a tapé
                                // Si c'est SOMMAIRE : on revient à l'accueil

                                if ($fctn == 'SOMMAIRE') {

                                        // L'utilisateur a appuyé sur 'SOMMAIRE'.
                                        // On veut maintenant exécuter la partie du script
                                        // 'accueil'

                                        $context['step'] = 'accueil';

                                        // On continue le script dans sa partie 'accueil': on
                                        // sort du bloc 'switch', mais on reste dans le bloc
                                        // 'while'

                                        break;

                                }

                                // C'est donc la touche ENVOI qui a été tapée

                                // Effacement de l'écran
                                $vdt = MiniPavi\MiniPaviCli::clearScreen();

                                // A la ligne 12, on affiche, centré, en magenta, le texte 'Vous
```

```
                                // avez tapé'

                                $vdt.= MiniPavi\MiniPaviCli::writeCentered(12,"Vous avez
tapé",VDT_TXTMAGENTA);

                                // A la ligne 14, on affiche, centré, en jaune, le texte saisi
                                // par l'utilisateur

                                $vdt.=
MiniPavi\MiniPaviCli::writeCentered(14,$content[0],VDT_TXTYELLOW);

                                // Maintenant, on attend qu'il tape n'importe quelle touche de
                                // fonction pour revenir à la saisie d'un texte
                                // (partie "menu" du script)

                                $context['step'] = 'menu';

                                // On attend une saisie utilisateur : on ne rappelle pas le
                                // script immédiatement

                                $directCall=false;

                                break 2;       // On sort du bloc "switch" et "while"
                        }
                }

                // Url à appeler lors de la prochaine saisie utilisateur (ou sans attendre si
                // directCall=true)
                // On reprend l'Url du script courant que l'on va placer dans la variable $nextPage


                if (!empty($_SERVER['HTTPS']) && strtolower($_SERVER['HTTPS']) !== 'off') {
                        $prot='https';
                } elseif (isset($_SERVER['HTTP_X_FORWARDED_PROTO']) &&
strtolower($_SERVER['HTTP_X_FORWARDED_PROTO']) === 'https') {
                        $prot='https';
                } elseif (!empty($_SERVER['HTTP_FRONT_END_HTTPS']) &&
strtolower($_SERVER['HTTP_FRONT_END_HTTPS']) !== 'off') {
                        $prot='https';
                } elseif (isset($_SERVER['SERVER_PORT']) && intval($_SERVER['SERVER_PORT']) === 443) {
                        $prot='https';
                } else
                        $prot='http';

                $nextPage=$prot."://".$_SERVER['HTTP_HOST']."".$_SERVER['PHP_SELF'];


                // On envoi à la passerelle le contenu à afficher ($vdt), l'url du prochain script
                // à appeler ($nextPage)
                // le contexte utilisateur sérialisé ($context), l'éventuelle commande à exécuter
                // On active l'echo de caractères pour que l'utilisateur voit ce qu'il tape
                // Si $directCall  = true, le script sera appellé immédiatement

                MiniPavi\MiniPaviCli::send($vdt,$nextPage,serialize($context),true,$cmd,$directCall);


} catch (Exception $e) {
        throw new Exception('Erreur MiniPavi '.$e->getMessage());
}
exit;
?>
```

8 – Debugging

Votre script, qu'il soit en PHP ou autre, doit répondre à la requête qu'il reçoit par un envoi de données en **format JSON**.

Si tel n'est pas le cas, ou si le format JSON est incorrect, la passerelle renverra une page sur l'écran du Minitel (ou de l'émulateur) vous indiquant un message du type **« ERR#03 Réponse non JSON »**.

Y seront également indiqués l'erreur JSON rencontrée, et le début des données erronées qu'a retourné votre script.

Lors de l'utilisation de la librairie « MiniPaviCli.php », la cause la plus fréquente de cette erreur est que votre script a émis des erreurs ou warning.

Si votre serveur est configuré pour que les informations sur les erreurs et warning s'affichent sur les pages webs, alors ces informations devraient apparaître également sur la page d'erreur « ERR#03 » renvoyée par la passerelle.

Lors du développement de votre script, il est conseillé d'activer l'affichage des erreurs et/ou les enregistrer dans votre fichier de logs d'erreurs PHP (ou du langage que vous utilisez) afin que vous puissiez en prendre connaissance et les corriger.

En PHP, pour permettre l'affichage des erreurs et des warnings sur la page web retournée (et donc dans les informations retournées à la passerelle, qui seront alors affichées sur la page d'erreur « #03 »), vous pouvez ajouter en début de script :

```
error_reporting(E_ERROR|E_WARNING);
ini_set('display_errors',1);
```

Pour supprimer l'affichage des erreurs, warnings etc., lorsque votre script est terminé et fonctionnel :

```
ini_set('display_errors',0);
```