

# Documentation SDK Dll .Net pour Zibase

Proposé par <http://www.planete-domotique.com>

Retrouvez nos produits, et les différents accessoires Zibase directement dans notre boutique

Téléchargez la dernière version de la dll ZibaseDll.dll et la dernière documentation à jour à l'adresse suivante : <http://www.planete-domotique.com/zibase>

La Zibase est un produit développé par la société Zodianet : <http://www.zodianet.com>



Version 1.3

## 1. Principe de développement

Le SDK Zibase vous permettra de communiquer avec votre Zibase grâce à un composant .Net facile à intégrer dans vos projets.

Pour développer vos projets, vous pouvez utiliser les produits Visual Studio Express disponibles à l'adresse suivante :

<http://msdn.microsoft.com/fr-fr/express/aa975050.aspx>

Le composant permet de gérer une ou plusieurs Zibase sur votre réseau local. Les Zibase sont détectées automatiquement dès le lancement de l'initialisation.

Vous pourrez gérer différents type d'événements déclenchés par la Zibase :

- 1) Détection d'une nouvelle Zibase
- 2) Détection d'un nouveau capteur
- 3) Mise à jour de la valeur d'un capteur

Vous pourrez aussi gérer les fonctions de contrôle de la Zibase :

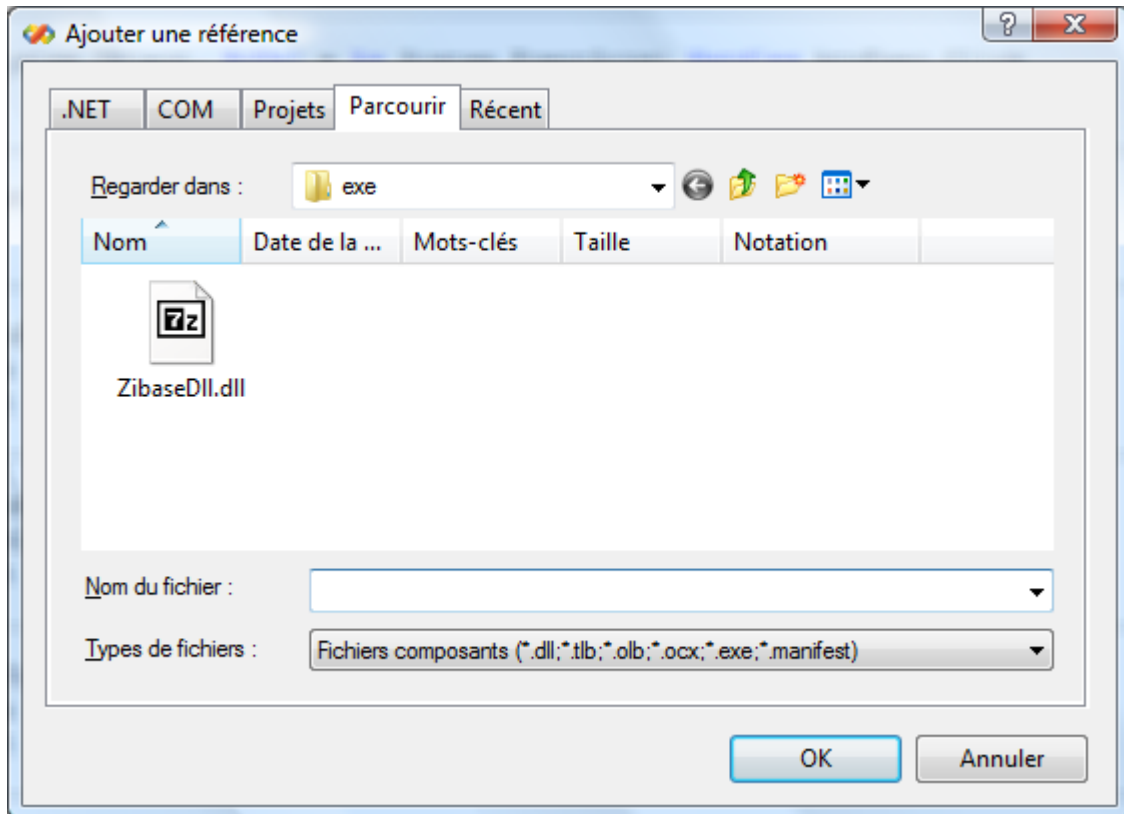
- 1) Envoie d'un ordre sur un périphérique domotique (exemple A1 ON) dans les différents protocoles reconnus par la Zibase (X10, Visonic, Chacon, Domia)
- 2) Exécution d'un scénario
- 3) Exécution d'un script Zibase

Le plugin dispose d'une sauvegarde locale de l'état des capteurs permettant à tout moment d'interroger la valeur de celui-ci.

## 2. Utilisation du module

Pour pouvoir utiliser le composant, il faut d'abord insérer la référence dans votre projet .Net.

Choisissez Projet/Ajouter une référence...



Sélectionnez la Dll ZibaseDll.dll et validez pour pouvoir désormais accéder aux fonctions et événements du module.

Il suffit ensuite d'importer le namespace correspondant :

```
Imports ZibaseDll
```

Les différentes fonctions seront accessible depuis un objet de type Zibase avec gestion d'événement :

```
Dim WithEvents zba As ZiBase = New ZiBase
```

Quatre événements sont disponibles :

- NewSensorDetected
- NewZibaseDetected
- UpdateSensorInfo
- WriteMessage

Les fonctions suivantes sont disponibles :

- StartZB : Permet de lancer l'initialisation des fonctions Zibase
- StopZB : Permet d'arrêter la gestion des fonctions Zibase
- RestartZibaseSearch : Permet de relancer une recherche automatique
- GetSensorInfo : Permet de récupérer la valeur courante d'un capteur
- AddZibase : Permet d'ajouter une connexion direct avec une Zibase
- SendCommand : Permet d'envoyer une commande à la Zibase
- ExecScript : Permet d'exécuter une commande de script Zibase
- RunScenario : Permet de lancer un scénario disponible sur la Zibase
- GetVar : Permet de lire le contenu d'une variable Zibase
- SetVar : Permet d'écrire une variable dans la Zibase
- GetCalendar : Permet de lire un calendrier Zibase
- SetCalendar : Permet de fixer un calendrier Zibase
- GetCalendarAsString : Permet d'avoir le calendrier sous forme de chaîne de caractère composée de '0' et '1' pour indiquer l'activation de chaque zone.
- GetCalendarFromString : Transforme deux chaînes correspondant au jour et heure en un entier dont les bits correspondent à un calendrier Zibase.
- SetVirtualProbeValue : Permet d'envoyer une valeur de sonde virtuelle à la Zibase.
- SetZibaseToken : Permet de définir le token d'une Zibase (pour charger ensuite la liste des scénarios et des périphériques disponibles)
- GetScenarioList : Permet de charger la liste des scénarios disponibles
- GetDevicesList : Permet de charger la liste des périphériques

### 3. Les évènements

#### a. NewSensorDetected

Cet événement est déclenché dès que la Zibase détecte un nouveau capteur. Chaque capteur est identifié par un ID unique et par un type de valeur (hum, tem).

Pour un même ID, on pourra donc avoir plusieurs type (exemple : Température et Humidité pour une sonde THGR228)

Le type de valeur est une chaîne de 3 ou 4 caractères définissant le type de valeur mesurée :

Type de valeur (Tag)	Correspondance
Bat	Niveau de batterie (Ok / Low)
lev	Niveau de réception RF (1 à 5)
tem	Température (°C)
hum	Humidité (%)
Lnk	Etat de la connexion Zibase
sta	Status pour un switch (ON/OFF)
temc	Température de consigne (Thermostat : °C)

kwh	Mesure d'énergie totale (CM119)
Kw	Mesure d'énergie instantanée (CM119)
tra	Niveau de pluie total (Total Rain)
cra	Niveau de pluie courant (Currant Rain)
awi	Mesure de la vitesse du vent
drt	Direction du vent
uvl	Niveau d'UV

Le prototype de l'évènement est le suivant :

```
Private Sub zba_NewSensorDetected(ByVal seInfo As
ZibaseDll.ZiBase.SensorInfo) Handles zba.NewSensorDetected
```

Les données sur le capteur sont transmises grâce à une structure de type  
Zibase.SensorInfo

Cette structure contient les informations sur le capteur ainsi que la dernière valeur  
mesurée :

```
Public Structure SensorInfo
    Dim sName As String
    Dim sType As String
    Dim sID As String
    Dim dwValue As Long
    Dim sValue As String
End Structure
```

Pour les valeurs de type réelles, la variable dwValue contient la valeur multipliée par 100.  
La variable sValue contient la valeur formatée avec l'unité de mesure (exemple : 23.5°C)

#### b. UpdateSensorInfo

Cet événement est géré comme l'évènement NewSensorDetected, mais il est déclenché  
à chaque réception d'une nouvelle valeur par la Zibase.

```
Private Sub zba_UpdateSensorInfo(ByVal seInfo As
ZibaseDll.ZiBase.SensorInfo) Handles zba.UpdateSensorInfo
```

### c. NewZibaseDetected

Cet événement est déclenché sur détection d'une nouvelle Zibase sur le réseau. La détection des Zibase s'effectue à l'initialisation (commande *StartZB*) ou après appel de la fonction *RestartZibaseSearch*

Les informations sur la Zibase sont transmises sous forme d'une structure :

```
Public Structure ZibaseInfo
    Dim sLabelBase As String
    Dim lIpAddress As Long
End Structure
```

Cette structure contient le nom logique de la Zibase (*sLabelBase*) et l'adresse IP de celle-ci.

Les différentes fonctions permettant de déclencher des actions sur la Zibase (*SendCommand*, *RunScenario*, *ExecScript*...) peuvent ensuite être exécutées sur une Zibase particulière en précisant son nom en premier paramètre ou sur toutes les Zibase détectée.

Dans le cas où une seule Zibase est présente sur le réseau, il n'est donc pas nécessaire de spécifier son nom dans les appels des fonctions.

#### d. WriteMessage

La fonction WriteMessage est un événement qui permet de récupérer les informations de Debug de la dll. Elle permet entre autre d'afficher les différentes données qui sont affichées dans le log sur la console de la Zibase (<http://www.zibase.net>)

```
Private Sub zba_WriteMessage(ByVal sMsg As String, ByVal level As Integer)  
Handles zba.WriteMessage
```

Le message transmit est contenu dans la variable sMsg avec une indication de niveau (INFO, DEBUG...)

## 4. Les méthodes

### a. StartZB

La méthode StartZB permet d'initialiser le module et de démarrer la communication avec les Zibase. La première phase consiste à démarrer une recherche des Zibase. Le paramètre *bAutoSearch* permet de désactiver cette recherche automatique des zibases sur le réseau. Dans ce cas, il faudra ajouter les zibases en utilisant la fonction AddZibase.

```
Public Sub StartZB(Optional ByVal bAutoSearch As Boolean = True)
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase  
zba.StartZB(false)
```

### b. StopZB

La méthode StopZB permet d'arrêter la communication avec les Zibase et de libérer les ressources utilisées.

L'appel à cette méthode est indispensable avant de fermer votre application

```
Public Sub StopZB()
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase  
zba.StopZB()
```

### c. RestartZibaseSearch

La méthode RestartZibaseSearch permet de relancer à tout moment la recherche des Zibase du réseau. Par défaut, la recherche des bases s'effectue au démarrage de l'application. Cette méthode permet de relancer le processus à tout moment.

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase  
zba.RestartZibaseSearch()
```

### d. GetSensorInfo

La fonction GetSensorInfo permet de récupérer à tout moment la valeur courante d'un capteur.

En paramètre, l'ID du capteur et le type de mesure. La structure renvoyée est la dernière transmise par l'événement *UpdateSensorInfo*

```
Public Function GetSensorInfo(ByVal sID As String, ByVal sType As String)
As SensorInfo
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
Dim sei As ZiBase.SensorInfo

sei = zba.GetSensorInfo(TextBox4.Text, TextBox3.Text)
MsgBox(sei.sID & "(" & sei.sType & ") = " & sei.sValue)
```

#### e. AddZibase

La fonction AddZibase permet de se connecter directement à une Zibase

En paramètre, l'ID du capteur et le type de mesure. La structure renvoyée est la dernière transmise par l'événement *UpdateSensorInfo*

```
Public Function GetSensorInfo(ByVal sID As String, ByVal sType As String)
As SensorInfo
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
```

#### f. SendCommand

La méthode SendCommand permet d'envoyer une commande à partir de la Zibase.

Les paramètres de la méthode sont :

- sZibaseName : Nom de la Zibase (facultatif)
- sAddress : Adresse du périphérique (format X10)
- iState : Etat à utiliser (ON/OFF/DIM)
- iDim : Valeur à utiliser pour une commande DIM (uniquement Chacon) : 0 à 100%
- iProtocol : Le protocole à utiliser (VISONIC, CHACON, X10, DOMIA)
- iNbBurst : Nombre d'émission de la commande à envoyer (par défaut 1)

```
Public Sub SendCommand(ByVal sZibaseName As String, ByVal sAddress As
String, ByVal iState As State, Optional ByVal iDim As Integer = 0, Optional
ByVal iProtocol As Protocol = Protocol.PROTOCOL_CHACON, Optional ByVal
iNbBurst As Integer = 1)
```

Pour l'état, il est possible de choisir une valeur parmi les états suivants :

```
Public Enum State
STATE_OFF = 0
STATE_ON = 1
STATE_DIM = 3
STATE_ASSOC = 7
End Enum
```

Pour le protocole, les valeurs suivantes sont possibles :

```
Public Enum Protocol
    PROTOCOL_BROADCAST = 0
    PROTOCOL_VISONIC433 = 1
    PROTOCOL_CHACON = 3
    PROTOCOL_DOMIA = 4
    PROTOCOL_X10 = 5
    PROTOCOL_ZWAVE = 6
    PROTOCOL_RFS10 = 7
    PROTOCOL_X2D433 = 8
    PROTOCOL_X2D868 = 9
End Enum
```

La valeur iDim est utilisée uniquement dans le cas du protocole Chacon et si l'état est positionné à DIM : iState = STATE\_DIM

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
zba.SendCommand("A1", ZiBase.State.STATE_OFF, 0,
ZiBase.Protocol.PROTOCOL_CHACON, 1)
```

#### g. ExecScript

La méthode ExecScript permet d'exécuter une instruction sur la Zibase.

Le langage de script de la Zibase n'étant pas public, cette fonction a un intérêt limitée.

```
Public Sub ExecScript(ByVal sZibaseName As String, ByVal sScript As String)
```

Le script est passé directement en paramètre.

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
Zba.ExecScript("Zibase00026d", "ON A1")
```

#### h. RunScenario

La méthode RunScenario permet de lancer un scénario construit sur la Zibase directement à partir de son nom.

Afin d'éviter les problèmes de compatibilité, nous vous conseillons de ne pas utiliser de caractères accentués dans vos noms de scénarios.

```
Public Sub RunScenario(ByVal sZibaseName As String, ByVal sCmd As String)
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
Zba.RunScenario("Zibase00026d", "Allumer Lampes")
```

### i. GetVar

La méthode *GetVar* permet de charger l'état d'une variable sur la Zibase.

```
Public Function GetVar(ByVal sZibaseName As String, ByVal dwNumVar As
UInt32) As UInt32
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
Dim valV2 As Int32

valV2 = Zba.GetVar("Zibase00026d", 2)
```

### j. SetVar

La méthode *SetVar* permet de fixer la valeur d'une variable sur la Zibase.

```
Public Function SetVar(ByVal sZibaseName As String, ByVal dwNumVar As
UInt32, ByVal dwVal As UInt32) As UInt32
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase

Zba.SetVar("Zibase00026d", 2, 24)
```

### k. GetCalendar

La méthode *GetCalendar* permet de charger un calendrier depuis la Zibase.

```
Public Function GetCalendar(ByVal sZibaseName As String, ByVal dwNumCal As
UInt32) As UInt32
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase
Dim calendar2 As Int32

calendar2 = Zba.GetCalendar("Zibase00026d", 2)
```

Pour les calendriers, chaque bit représente l'état actif ou inactif d'une journée et d'une heure donnée.

Dimanche	Samedi	Vendredi	Jeudi	Mercredi	Mardi	Lundi	23h	22h	21h	20h	19h	18h	17h	16h	15h	14h	13h	12h	11h	10h	9h	8h	7h	6h	5h	4h	3h	2h	1h	0h
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

## l. SetCalendar

La méthode SetCalendar permet de fixer la valeur d'un calendrier sur la Zibase.

```
Public Function SetVar(ByVal sZibaseName As String, ByVal dwNumVar As  
UInt32, ByVal dwVal As UInt32) As UInt32
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase  
  
Zba.SetVar("Zibase00026d", 2, 24)
```

## m. GetCalendarAsString

La méthode GetCalendarAsString permet de transformer un calendrier représenté par un entier (format Zibase) en chaîne de caractère.

La chaîne de caractère représente l'activation des 7 jours de la semaine (Lundi à Dimanche) suivi par les 24h de chaque jour (00h à 23h)

```
Public Function GetCalendarAsString(ByVal sZibaseName As String, ByVal  
dwNumCal As UInt32) As String
```

## n. GetCalendarFromString

La méthode GetCalendarFromString permet de transformer deux chaînes de caractères en calendrier compréhensible par la zibase.

La première chaîne représente les jours (Lundi à mardi) et la seconde les heures (00h à 24h)

```
Public Function GetCalendarFromString(ByVal sDay As String, ByVal sHour As  
String) As UInt32
```

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase  
  
Zba.GetCalendarFromString("0110011", "010101010101010101010101")
```

Le calendrier est actif Mardi, Mercredi, Samedi et Dimanche. Et une heure sur deux.

#### o. SetVirtualProbeValue

La méthode SetVirtualProbeValue est très intéressante, elle permet de simuler des capteurs virtuels pour envoyer des données vers la Zibase.

Il existe quatre types de capteur :

```
Public Enum VirtualProbeType
    TEMP_SENSOR = 0
    TEMP_HUM_SENSOR = 1
    POWER_SENSOR = 2
    WATER_SENSOR = 3
End Enum
```

Les capteurs de type TEMP\_SENSOR simulent une sonde THR128.

Les capteurs de type TEMP\_HUM\_SENSOR simulent une sonde THGR228

Les capteurs POWER\_SENSOR simulent des modules OWL CM119

Les capteurs WATER\_SENSOR simulent un pluviomètre.

Le prototype de la fonction est le suivant :

```
Public Sub SetVirtualProbeValue(ByVal sZibaseName As String, ByVal
wSensorID As UInt16, ByVal SensorType As VirtualProbeType, ByVal dwValue1
As UInt32, ByVal dwValue2 As UInt32, ByVal dwLowBat As UInt32)
Public Sub SetZibaseToken(ByVal sZibaseName As String, ByVal sToken As
String)
```

Les valeurs dwValue1 et dwValue2 sont celles identiques à I1 et I2 disponibles sur l'interface Zibase.

Exemple d'utilisation :

```
Dim WithEvents zba As ZiBase = New ZiBase

zba.SetVirtualProbeValue(1, ZiBase.VirtualProbeType.TEMP_HUM_SENSOR,
120, 41, 0)
```

#### p. SetZibaseToken

Les fonctions de lecture de la liste de scénarios et la liste des périphériques (cf ci-dessous) nécessitent l'utilisation du Token API (disponible dans l'onglet Système du mode Expert de votre interface Zibase)

Avant d'utiliser les fonctions `GetScenarioList` et `GetDevicesList`, il faudra donc définir le Token avec `SetZibaseToken`.

```
Public Sub SetZibaseToken(ByVal sZibaseName As String, ByVal sToken As String)
```

#### q. GetScenarioList

La fonction `GetScenarioList` permet de récupérer la liste de tous les scénarios définis sur une Zibase.

```
Public Function GetScenarioList(ByVal sZibaseName As String) As String
```

#### r. GetDevicesList

La fonction `GetDevicesList` permet de récupérer la liste de tous les périphériques définis sur une Zibase.

```
Public Function GetDevicesList(ByVal sZibaseName As String) As String
```