

DomotlcApp

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	EP Namespace Reference . . . . .	9
5.1.1	Function Documentation . . . . .	11
5.1.1.1	__declspec(dllexport) Core *Core_New(char *file) . . . . .	11
5.1.1.2	Core_Delete(Core *core) . . . . .	11
5.1.1.3	Core_New(char *file) . . . . .	11
5.1.1.4	Core_NewFromSave(char *file) . . . . .	11
5.1.1.5	Core_SaveAndDelete(Core *core) . . . . .	11
5.1.1.6	Equipment_getIpFibaro() . . . . .	11
5.1.1.7	Equipment_getIpKira() . . . . .	12
5.1.1.8	Equipment_getLoginFibaro() . . . . .	12
5.1.1.9	Equipment_getPasswordFibaro() . . . . .	12
5.1.1.10	Equipment_setIpFibaro(char *new_ip) . . . . .	12

5.1.1.11	Equipment_setIpKira(char *new_ip)	12
5.1.1.12	Equipment_setLoginFibaro(char *new_login)	13
5.1.1.13	Equipment_setPasswordFibaro(char *new_password)	13
5.1.1.14	EquipmentFibaro_Delete(EquipmentFibaro *eq)	13
5.1.1.15	EquipmentFibaro_New(char *name, char *ico, Node *parent, int equipmentId, char *action)	13
5.1.1.16	EquipmentKira_Delete(EquipmentKira *eq)	14
5.1.1.17	EquipmentKira_New(char *name, char *ico, Node *parent, int buttonId, int page)	15
5.1.1.18	Node_Delete(Node *node)	15
5.1.1.19	Node_New(char *name, char *ico)	15
5.1.1.20	requeteHttpFibaro(char *s1, char *s2, char *user, char *pass)	15
5.1.1.21	requeteHttpKira(char *s1, char *s2)	16
5.1.1.22	Room_Delete(Room *room)	16
5.1.1.23	Room_New(char *name, char *ico)	16
5.1.2	Variable Documentation	16
5.1.2.1	action	16
5.1.2.2	buttonId	16
5.1.2.3	equipmentId	16
5.1.2.4	ico	16
5.1.2.5	page	17
5.1.2.6	parent	17

<b>6</b>	<b>Class Documentation</b>	<b>19</b>
6.1	EP::Core Class Reference	19
6.1.1	Detailed Description	20
6.1.2	Constructor & Destructor Documentation	20
6.1.2.1	Core(char *file)	20
6.1.2.2	~Core()	20
6.1.3	Member Function Documentation	20
6.1.3.1	addRoom(Room *room)	20
6.1.3.2	deleteRoomByIndex(int index)	20
6.1.3.3	deleteRoomByName(char *name)	21
6.1.3.4	getCOMPort()	21
6.1.3.5	getFileSave()	21
6.1.3.6	getIconSize()	21
6.1.3.7	getNumberRooms()	21
6.1.3.8	getRoomByIndex(int index)	21
6.1.3.9	getRoomByName(char *name)	22
6.1.3.10	getRooms()	22
6.1.3.11	getThemeld()	22
6.1.3.12	load()	22
6.1.3.13	save()	22
6.1.3.14	setCOMPort(char *port)	22
6.1.3.15	setFileSave(char *name)	23
6.1.3.16	setIconSize(int size)	23
6.1.3.17	setThemeld(int id)	23
6.2	EP::Equipment Class Reference	23
6.2.1	Detailed Description	24
6.2.2	Constructor & Destructor Documentation	24
6.2.2.1	Equipment(char *name, char *ico, Node *parent, int typeOf)	24
6.2.2.2	~Equipment()	25
6.2.3	Member Function Documentation	25

6.2.3.1	<a href="#">getIpFibaro()</a>	25
6.2.3.2	<a href="#">getIpKira()</a>	25
6.2.3.3	<a href="#">getLoginFibaro()</a>	25
6.2.3.4	<a href="#">getNodeParent()</a>	25
6.2.3.5	<a href="#">getPasswordFibaro()</a>	25
6.2.3.6	<a href="#">getTypeOf()</a>	25
6.2.3.7	<a href="#">sendRequest()=0</a>	25
6.2.3.8	<a href="#">setIpFibaro(char *new_ip)</a>	25
6.2.3.9	<a href="#">setIpKira(char *new_ip)</a>	26
6.2.3.10	<a href="#">setLoginFibaro(char *new_login)</a>	26
6.2.3.11	<a href="#">setPasswordFibaro(char *new_password)</a>	26
6.2.4	<a href="#">Member Data Documentation</a>	26
6.2.4.1	<a href="#">Fibaro_login</a>	26
6.2.4.2	<a href="#">Fibaro_password</a>	26
6.2.4.3	<a href="#">IP_Fibaro</a>	26
6.2.4.4	<a href="#">IP_Kira</a>	27
6.2.4.5	<a href="#">m_roomParent</a>	27
6.2.4.6	<a href="#">m_typeOf</a>	27
6.3	<a href="#">EP::EquipmentFibaro Class Reference</a>	27
6.3.1	<a href="#">Detailed Description</a>	27
6.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	27
6.3.2.1	<a href="#">EquipmentFibaro(char *name, char *ico, Node *parent, int equipmentId, char *action)</a>	27
6.3.2.2	<a href="#">~EquipmentFibaro()</a>	28
6.3.3	<a href="#">Member Function Documentation</a>	28
6.3.3.1	<a href="#">getAction()</a>	28
6.3.3.2	<a href="#">getEquipmentId()</a>	28
6.3.3.3	<a href="#">sendRequest()</a>	28
6.4	<a href="#">EP::EquipmentKira Class Reference</a>	28
6.4.1	<a href="#">Detailed Description</a>	29
6.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	29

6.4.2.1	EquipmentKira(char *name, char *ico, Node *parent, int buttonId, int page) . . .	29
6.4.2.2	~EquipmentKira() . . . . .	29
6.4.3	Member Function Documentation . . . . .	29
6.4.3.1	getButtonId() . . . . .	29
6.4.3.2	getPageNumber() . . . . .	29
6.4.3.3	sendRequest() . . . . .	30
6.4.3.4	setButtonId(int new_id) . . . . .	30
6.4.3.5	setPageNumber(int new_PageNumber) . . . . .	30
6.5	EP::Node Class Reference . . . . .	30
6.5.1	Detailed Description . . . . .	31
6.5.2	Constructor & Destructor Documentation . . . . .	31
6.5.2.1	Node(char *name, char *ico) . . . . .	31
6.5.2.2	~Node(void) . . . . .	31
6.5.3	Member Function Documentation . . . . .	31
6.5.3.1	getIco() . . . . .	31
6.5.3.2	getName() . . . . .	32
6.5.3.3	setIco(char *ico) . . . . .	32
6.5.3.4	setName(char *name) . . . . .	33
6.5.4	Member Data Documentation . . . . .	33
6.5.4.1	m_ico . . . . .	33
6.5.4.2	m_name . . . . .	33
6.5.4.3	m_parent . . . . .	33
6.6	EP::Room Class Reference . . . . .	33
6.6.1	Detailed Description . . . . .	34
6.6.2	Constructor & Destructor Documentation . . . . .	34
6.6.2.1	Room(char *name, char *ico) . . . . .	34
6.6.2.2	~Room() . . . . .	34
6.6.3	Member Function Documentation . . . . .	34
6.6.3.1	addEquipment(Equipment *equip) . . . . .	34
6.6.3.2	deleteEquipmentByIndex(int index) . . . . .	35
6.6.3.3	deleteEquipmentByName(char *name) . . . . .	35
6.6.3.4	getEquipmentByIndex(int index) . . . . .	35
6.6.3.5	getEquipmentByName(char *name) . . . . .	36
6.6.3.6	getEquipments() . . . . .	36
6.6.3.7	getNumberEquipments() . . . . .	36

<b>7 File Documentation</b>	<b>37</b>
7.1 ModelDll/include/Core.h File Reference . . . . .	37
7.1.1 Macro Definition Documentation . . . . .	38
7.1.1.1 DllExport . . . . .	38
7.1.1.2 FILESAVE_NAME_SIZE . . . . .	38
7.2 ModelDll/include/Equipment.h File Reference . . . . .	38
7.2.1 Macro Definition Documentation . . . . .	39
7.2.1.1 DllExport . . . . .	39
7.3 ModelDll/include/Node.h File Reference . . . . .	39
7.3.1 Macro Definition Documentation . . . . .	39
7.3.1.1 DllExport . . . . .	39
7.4 ModelDll/include/RequeteHttp.h File Reference . . . . .	39
7.4.1 Macro Definition Documentation . . . . .	40
7.4.1.1 DllImport . . . . .	40
7.5 ModelDll/include/Room.h File Reference . . . . .	40
7.5.1 Macro Definition Documentation . . . . .	41
7.5.1.1 DllExport . . . . .	41
7.6 ModelDll/src/Core.cpp File Reference . . . . .	41
7.7 ModelDll/src/Equipment.cpp File Reference . . . . .	41
7.8 ModelDll/src/Node.cpp File Reference . . . . .	42
7.9 ModelDll/src/Room.cpp File Reference . . . . .	42
<b>Index</b>	<b>43</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

EP . . . . .	9
--------------	---



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

EP::Core . . . . .	19
EP::Node . . . . .	30
EP::Equipment . . . . .	23
EP::EquipmentFibaro . . . . .	27
EP::EquipmentKira . . . . .	28
EP::Room . . . . .	33



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">EP::Core</a>	Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci . . . . .	19
<a href="#">EP::Equipment</a>	Représente un équipement. Cette classe abstraite est spécialisée par <a href="#">EquipmentKira</a> et <a href="#">EquipmentFibaro</a> . . . . .	23
<a href="#">EP::EquipmentFibaro</a>	Représente un équipement lié à une Fibaro, hérite de <a href="#">Equipment</a> . . . . .	27
<a href="#">EP::EquipmentKira</a>	Représente un équipement lié à une Kira, hérite de <a href="#">Equipment</a> . . . . .	28
<a href="#">EP::Node</a>	Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes <a href="#">Equipment</a> et <a href="#">Room</a> , et n'est donc pas utilisée telle qu'elle . . . . .	30
<a href="#">EP::Room</a>	Représente une pièce de la maison ; Hérite de <a href="#">Node</a> . On peut y retrouver les différents équipements liés à une pièce et les gérer . . . . .	33



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

ModelDll/include/Core.h . . . . .	37
ModelDll/include/Equipment.h . . . . .	38
ModelDll/include/Node.h . . . . .	39
ModelDll/include/RequeteHttp.h . . . . .	39
ModelDll/include/Room.h . . . . .	40
ModelDll/src/Core.cpp . . . . .	41
ModelDll/src/Equipment.cpp . . . . .	41
ModelDll/src/Node.cpp . . . . .	42
ModelDll/src/Room.cpp . . . . .	42





## Chapter 5

# Namespace Documentation

### 5.1 EP Namespace Reference

#### Classes

- class [Core](#)  
*Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.*
- class [Equipment](#)  
*Représente un équipement. Cette classe abstraite est spécialisée par [EquipmentKira](#) et [EquipmentFibaro](#).*
- class [EquipmentFibaro](#)  
*Représente un équipement lié à une Fibaro, hérite de [Equipment](#).*
- class [EquipmentKira](#)  
*Représente un équipement lié à une Kira, hérite de [Equipment](#).*
- class [Node](#)  
*Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes [Equipment](#) et [Room](#), et n'est donc pas utilisée telle qu'elle.*
- class [Room](#)  
*Représente une pièce de la maison ; Hérite de [Node](#). On peut y retrouver les différents équipements liés à une pièce et les gérer.*

#### Functions

- [DllExport Core \\* Core\\_New](#) (char \*file)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Core](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- [DllExport Core \\* Core\\_NewFromSave](#) (char \*file)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Core](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur. Après création de l'objet, la méthode [Core::load\(\)](#) est appelée sur celui-ci.*
- [DllExport void Core\\_Delete](#) (Core \*core)  
*Destructeur statique, pour permettre la destruction des objets [Core](#) en passant par la DLL.*
- [DllExport void Core\\_SaveAndDelete](#) (Core \*core)  
*Destructeur statique, pour permettre la destruction des objets [Core](#) en passant par la DLL. La méthode [Core::save\(\)](#) est appelée avant sa destruction.*
- [DllExport char \\* Equipment\\_getIkpKira](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*

- **DllExport** char \* [Equipment\\_getIpFibaro](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** int [Equipment\\_setIpKira](#) (char \*new\_ip)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** int [Equipment\\_setIpFibaro](#) (char \*new\_ip)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** char \* [Equipment\\_getLoginFibaro](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** char \* [Equipment\\_getPasswordFibaro](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** int [Equipment\\_setLoginFibaro](#) (char \*new\_login)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** int [Equipment\\_setPasswordFibaro](#) (char \*new\_password)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- **DllExport** [EquipmentKira](#) \* [EquipmentKira\\_New](#) (char \*name, char \*ico, [Node](#) \*parent, int buttonId, int page)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [EquipmentKira](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** void [EquipmentKira\\_Delete](#) ([EquipmentKira](#) \*eq)  
*Destructeur statique, pour permettre la destruction des objets [EquipmentKira](#) en passant par la DLL.*
- **DllExport** [EquipmentFibaro](#) \* [EquipmentFibaro\\_New](#) (char \*name, char \*ico, [Node](#) \*parent, int equipmentId, char \*action)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [EquipmentFibaro](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** void [EquipmentFibaro\\_Delete](#) ([EquipmentFibaro](#) \*eq)  
*Destructeur statique, pour permettre la destruction des objets [EquipmentFibaro](#) en passant par la DLL.*
- **DllExport** [Node](#) \* [Node\\_New](#) (char \*name, char \*ico)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Node](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** void [Node\\_Delete](#) ([Node](#) \*node)  
*Destructeur statique, pour permettre la destruction des objets [Node](#) en passant par la DLL.*
- void **DllImport** [requeteHttpKira](#) (char \*s1, char \*s2)  
*Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Kira. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode [EquipmentKira::sendRequest\(\)](#) ; les paramètres sont construits automatiquement à partir des informations du l'équipement.*
- void **DllImport** [requeteHttpFibaro](#) (char \*s1, char \*s2, char \*user, char \*pass)  
*Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Fibaro. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode [EquipmentFibaro::sendRequest\(\)](#) ; les paramètres sont construits automatiquement à partir des informations du l'équipement.*
- **DllExport** [Room](#) \* [Room\\_New](#) (char \*name, char \*ico)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Room](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** void [Room\\_Delete](#) ([Room](#) \*room)  
*Destructeur statique, pour permettre la destruction des objets [Room](#) en passant par la DLL.*
- **\_\_declspec** (dllexport) [Core](#) \*[Core\\_New](#)(char \*file)

## Variables

- char \* ico
- char [Node](#) \* parent
- char [Node](#) int buttonId
- char [Node](#) int int page
- char [Node](#) int equipmentId
- char [Node](#) int char \* action

### 5.1.1 Function Documentation

#### 5.1.1.1 EP::\_\_declspec ( dllexport ) [new]

#### 5.1.1.2 DllExport void EP::Core\_Delete ( Core \* core )

Destructeur statique, pour permettre la destruction des objets [Core](#) en passant par la DLL.

##### Parameters

<i>core</i>	Un pointeur vers l'objet à détruire.
-------------	--------------------------------------

#### 5.1.1.3 DllExport Core\* EP::Core\_New ( char \* file )

Constructeur statique utilisé pour permettre l'utilisation des objets [Core](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

##### Returns

Un pointeur vers l'objet créé.

#### 5.1.1.4 DllExport Core\* EP::Core\_NewFromSave ( char \* file )

Constructeur statique utilisé pour permettre l'utilisation des objets [Core](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur. Après création de l'objet, la méthode [Core::load\(\)](#) est appelée sur celui-ci.

##### Returns

Un pointeur vers l'objet créé.

#### 5.1.1.5 DllExport void EP::Core\_SaveAndDelete ( Core \* core )

Destructeur statique, pour permettre la destruction des objets [Core](#) en passant par la DLL. La méthode [Core::save\(\)](#) est appelée avant sa destruction.

##### Parameters

<i>core</i>	Un pointeur vers l'objet à détruire.
-------------	--------------------------------------

#### 5.1.1.6 DllExport char\* EP::Equipment\_getIpFibaro ( )

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

**Returns**

L'adresse IP de la Fibaro.

**5.1.1.7 DllExport char\* EP::Equipment\_getIpKira ( )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

**Returns**

L'adresse IP de la Kira.

**5.1.1.8 DllExport char\* EP::Equipment\_getLoginFibaro ( )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

**Returns**

Le login de la Fibaro.

**5.1.1.9 DllExport char\* EP::Equipment\_getPasswordFibaro ( )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

**Returns**

Le password de la Fibaro.

**5.1.1.10 DllExport int EP::Equipment\_setIpFibaro ( char \* *new\_ip* )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

**Parameters**

<i>new_ip</i>	La nouvelle adresse IP de la Fibaro.
---------------	--------------------------------------

**Returns**

0 si l'adresse est incorrecte, 1 sinon.

**5.1.1.11 DllExport int EP::Equipment\_setIpKira ( char \* *new\_ip* )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

## Parameters

<i>new↔ _ip</i>	La nouvelle adresse IP de la Kira.
---------------------	------------------------------------

## Returns

0 si l'adresse est incorrecte, 1 sinon.

**5.1.1.12 DllExport int EP::Equipment\_setLoginFibaro ( char \* *new\_login* )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

## Parameters

<i>new_login</i>	Le nouveau login de la Fibaro.
------------------	--------------------------------

**5.1.1.13 DllExport int EP::Equipment\_setPasswordFibaro ( char \* *new\_password* )**

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

## Parameters

<i>new_password</i>	Le nouveau password de la Fibaro.
---------------------	-----------------------------------

**5.1.1.14 DllExport void EP::EquipmentFibaro\_Delete ( EquipmentFibaro \* *eq* )**

Destructeur statique, pour permettre la destruction des objets [EquipmentFibaro](#) en passant par la DLL.

## Parameters

<i>eq</i>	Un pointeur vers l'objet à détruire.
-----------	--------------------------------------

**5.1.1.15 DllExport EquipmentFibaro\* EP::EquipmentFibaro\_New ( char \* *name*, char \* *ico*, Node \* *parent*, int *equipmentId*, char \* *action* )**

Constructeur statique utilisé pour permettre l'utilisation des objets [EquipmentFibaro](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

## Returns

Un pointeur vers l'objet créé.

#### 5.1.1.16 DllExport void EP::EquipmentKira\_Delete ( EquipmentKira \* eq )

Destructeur statique, pour permettre la destruction des objets [EquipmentKira](#) en passant par la DLL.

## Parameters

<i>eq</i>	Un pointeur vers l'objet à détruire.
-----------	--------------------------------------

**5.1.1.17 DllExport EquipmentKira\* EP::EquipmentKira\_New ( char \* *name*, char \* *ico*, Node \* *parent*, int *buttonId*, int *page* )**

Constructeur statique utilisé pour permettre l'utilisation des objets [EquipmentKira](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

## Returns

Un pointeur vers l'objet créé.

**5.1.1.18 DllExport void EP::Node\_Delete ( Node \* *node* )**

Destructeur statique, pour permettre la destruction des objets [Node](#) en passant par la DLL.

## Parameters

<i>node</i>	Un pointeur vers l'objet à détruire.
-------------	--------------------------------------

**5.1.1.19 DllExport Node\* EP::Node\_New ( char \* *name*, char \* *ico* )**

Constructeur statique utilisé pour permettre l'utilisation des objets [Node](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

## Returns

Un pointeur vers l'objet créé.

**5.1.1.20 void DllImport EP::requeteHttpFibaro ( char \* *s1*, char \* *s2*, char \* *user*, char \* *pass* )**

Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Fibaro. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode [EquipmentFibaro::sendRequest\(\)](#) ; les paramètres sont construits automatiquement à partir des informations de l'équipement.

## Parameters

<i>s1</i>	L'adresse IP de la Fibaro.
<i>s2</i>	La deuxième partie de l'adresse correspondant à la requête HTTP.
<i>user</i>	Le login utilisé pour s'authentifier auprès de la Fibaro.
<i>pass</i>	Le login utilisé pour s'authentifier auprès de la Fibaro.

#### 5.1.1.21 void DllImport EP::requeteHttpKira ( char \* *s1*, char \* *s2* )

Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Kira. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode [EquipmentKira::sendRequest\(\)](#) ; les paramètres sont construits automatiquement à partir des informations du l'équipement.

##### Parameters

<i>s1</i>	L'adresse IP de la Kira.
<i>s2</i>	La deuxième partie de l'adresse correspondant à la requête HTTP.

#### 5.1.1.22 DllExport void EP::Room\_Delete ( Room \* *room* )

Destructeur statique, pour permettre la destruction des objets [Room](#) en passant par la DLL.

##### Parameters

<i>room</i>	Un pointeur vers l'objet à détruire.
-------------	--------------------------------------

#### 5.1.1.23 DllExport Room\* EP::Room\_New ( char \* *name*, char \* *ico* )

Constructeur statique utilisé pour permettre l'utilisation des objets [Room](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

##### Returns

Un pointeur vers l'objet créé.

### 5.1.2 Variable Documentation

#### 5.1.2.1 char Node int char\* EP::action

##### Initial value:

```
{
    return new EquipmentFibaro(name, ico, parent, equipmentId,
    action)
```

#### 5.1.2.2 char Node int EP::buttonId

#### 5.1.2.3 char Node int EP::equipmentId

#### 5.1.2.4 char \* EP::ico

##### Initial value:

```
{
    return new Node(name, ico)
```



## 5.1.2.5 char Node int int EP::page

Initial value:

```
{  
    return new EquipmentKira(name, ico, parent, buttonId,  
        page)
```

## 5.1.2.6 char Node \* EP::parent



## Chapter 6

# Class Documentation

### 6.1 EP::Core Class Reference

Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.

```
#include <Core.h>
```

#### Public Member Functions

- [Core](#) (char \*file)  
*Le constructeur de Core.summary.*
- [~Core](#) ()  
*Le destructeur de Core. Vide la liste de pièces.*
- int [save](#) ()  
*Sauvegarde l'état actuel de l'application dans le fichier de sauvegarde.*
- int [load](#) ()  
*Charge l'application depuis le fichier de sauvegarde.*
- int [addRoom](#) ([Room](#) \*room)  
*Ajoute le pointeur d'une pièce donnée dans m\_listRooms. Le nom d'une pièce doit être unique.*
- int [deleteRoomByIndex](#) (int index)  
*Supprime la pièce dont l'index est donné en paramètre.*
- int [deleteRoomByName](#) (char \*name)  
*Supprime la pièce dont le nom est donné en paramètre.*
- std::vector< [Room](#) \* > \* [getRooms](#) ()  
*Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec [getNumberRooms](#) et [getRoomByIndex](#).*
- [Room](#) \* [getRoomByName](#) (char \*name)  
*Donne un pointeur vers la pièce dont le nom est donné en paramètre.*
- [Room](#) \* [getRoomByIndex](#) (int index)  
*Donne un pointeur vers la pièce dont l'index est donné en paramètre.*
- char \* [getFileSave](#) ()
- void [setFileSave](#) (char \*name)
- int [getNumberRooms](#) ()
- char \* [getCOMPort](#) ()
- void [setCOMPort](#) (char \*port)
- int [getThemeld](#) ()
- int [getIconSize](#) ()
- void [setThemeld](#) (int id)  
*Permet de changer le thème utilisé.*
- void [setIconSize](#) (int size)  
*Permet de changer la taille des icônes.*

### 6.1.1 Detailed Description

Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 EP::Core::Core ( char \* *file* )

Le constructeur de Core.summary.

##### Parameters

<i>file</i>	Le nom du fichier de sauvegarde.
-------------	----------------------------------

#### 6.1.2.2 EP::Core::~~Core ( )

Le destructeur de [Core](#). Vide la liste de pièces.

### 6.1.3 Member Function Documentation

#### 6.1.3.1 int EP::Core::addRoom ( Room \* *room* )

Ajoute le pointeur d'une pièce donnée dans m\_listRooms. Le nom d'une pièce doit être unique.

##### Parameters

<i>room</i>	Le pointeur de la pièce à ajouter.
-------------	------------------------------------

##### Returns

0 si tout s'est bien passé, 1 si le nom est déjà pris.

#### 6.1.3.2 int EP::Core::deleteRoomByIndex ( int *index* )

Supprime la pièce dont l'index est donné en paramètre.

##### Parameters

<i>index</i>	L'index de la pièce à supprimer de m_listRooms.
--------------	---

**Returns**

0 si tout s'est bien passé, 1 si l'index ne correspond à aucune pièce.

**6.1.3.3 int EP::Core::deleteRoomByName ( char \* *name* )**

Supprime la pièce dont le nom est donné en paramètre.

**Parameters**

<i>name</i>	Le nom de la pièce à supprimer de m_listRooms.
-------------	--

**Returns**

0 si tout s'est bien passé, 1 si le nom ne correspond à aucune pièce.

**6.1.3.4 char \* EP::Core::getCOMPort ( )****Returns**

Le nom du port série utilisé pour communiquer avec le fauteuil.

**6.1.3.5 char \* EP::Core::getFileSave ( )****Returns**

Le nom du fichier de sauvegarde.

**6.1.3.6 int EP::Core::getIconSize ( )****Returns**

L'identifiant pour la taille des icônes.

**6.1.3.7 int EP::Core::getNumberRooms ( )****Returns**

Le nombre de pièces de m\_listRooms.

**6.1.3.8 Room \* EP::Core::getRoomByIndex ( int *index* )**

Donne un pointeur vers la pièce dont l'index est donné en paramètre.

## Parameters

<i>index</i>	L'index de la pièce qu'on veut récupérer.
--------------	---

## Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

**6.1.3.9 Room \* EP::Core::getRoomByName ( char \* *name* )**

Donne un pointeur vers la pièce dont le nom est donné en paramètre.

## Parameters

<i>name</i>	Le nom de la pièce qu'on veut récupérer.
-------------	--

## Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

**6.1.3.10 vector< Room \* > \* EP::Core::getRooms ( )**

Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec `getNumberRooms` et `getRoomByIndex`.

## Returns

L'attribut `m_listRooms`.

**6.1.3.11 int EP::Core::getThemeld ( )**

## Returns

L'identifiant du thème utilisé.

**6.1.3.12 int EP::Core::load ( )**

Charge l'application depuis le fichier de sauvegarde.

**6.1.3.13 int EP::Core::save ( )**

Sauvegarde l'état actuel de l'application dans le fichier de sauvegarde.

**6.1.3.14 void EP::Core::setCOMPort ( char \* *port* )**

## Parameters

<i>port</i>	Le nom du port série à utiliser pour communiquer avec le fauteuil.
-------------	--

6.1.3.15 void EP::Core::setFileSave ( char \* *name* )

## Parameters

<i>name</i>	Le nouveau fichier de sauvegarde.
-------------	-----------------------------------

6.1.3.16 void EP::Core::setIconSize ( int *size* )

Permet de changer la taille des icônes.

## Parameters

<i>size</i>	Un chiffre correspondant à la nouvelle taille des icônes.
-------------	---

6.1.3.17 void EP::Core::setThemeld ( int *id* )

Permet de changer le thème utilisé.

## Parameters

<i>id</i>	L'identifiant du nouveau thème.
-----------	---------------------------------

The documentation for this class was generated from the following files:

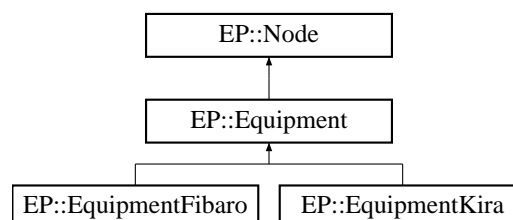
- ModelDll/include/[Core.h](#)
- ModelDll/src/[Core.cpp](#)

## 6.2 EP::Equipment Class Reference

Représente un équipement. Cette classe abstraite est spécialisée par [EquipmentKira](#) et [EquipmentFibaro](#).

```
#include <Equipment.h>
```

Inheritance diagram for EP::Equipment:



## Public Member Functions

- [Equipment](#) (char \*name, char \*ico, [Node](#) \*parent, int typeOf)  
*Appelle le constructeur de [Node](#), initialise m\_roomParent et m\_typeOf.*
- virtual [~Equipment](#) ()
- virtual int [sendRequest](#) ()=0  
*Envoie la requête HTTP permettant d'interagir avec l'équipement.*
- int [get.TypeOf](#) ()
- [Node](#) \* [getNodeParent](#) ()

## Static Public Member Functions

- static char \* [getIpKira](#) ()
- static char \* [getIpFibaro](#) ()
- static char \* [getLoginFibaro](#) ()
- static char \* [getPasswordFibaro](#) ()
- static int [setIpKira](#) (char \*new\_ip)
- static int [setIpFibaro](#) (char \*new\_ip)
- static int [setLoginFibaro](#) (char \*new\_login)
- static int [setPasswordFibaro](#) (char \*new\_password)

## Protected Attributes

- [Node](#) \* m\_roomParent  
*Un pointeur vers la pièce contenant l'équipement.*
- int m\_typeOf  
*1: pour un équipement lié à la Kira, 2: pour un équipement lié à la Fibaro.*

## Static Protected Attributes

- static char [IP\\_Kira](#) [15] = "192.168.1.31"  
*L'adresse IP de la Kira.*
- static char [IP\\_Fibaro](#) [15] = "192.168.81.1"  
*L'adresse IP de la Fibaro.*
- static char [Fibaro\\_login](#) [300] = "admin"  
*Le login de la Fibaro.*
- static char [Fibaro\\_password](#) [300] = "admin"  
*Le password de la Fibaro.*

### 6.2.1 Detailed Description

Représente un équipement. Cette classe abstraite est spécialisée par [EquipmentKira](#) et [EquipmentFibaro](#).

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 EP::Equipment::Equipment ( char \* name, char \* ico, [Node](#) \* parent, int typeOf ) [inline]

Appelle le constructeur de [Node](#), initialise m\_roomParent et m\_typeOf.



## Parameters

<i>parent</i>	La pièce contenant l'équipement
<i>typeOf</i>	1: pour un équipement lié à la Kira, 2: pour un équipement lié à la Fibaro.

**6.2.2.2** `virtual EP::Equipment::~~Equipment ( ) [inline],[virtual]`

### 6.2.3 Member Function Documentation

**6.2.3.1** `static char* EP::Equipment::getIpFibaro ( ) [inline],[static]`

## Returns

L'adresse IP de la Fibaro.

**6.2.3.2** `static char* EP::Equipment::getIpKira ( ) [inline],[static]`

## Returns

L'adresse IP de la Kira.

**6.2.3.3** `static char* EP::Equipment::getLoginFibaro ( ) [inline],[static]`

## Returns

Le login de la Fibaro.

**6.2.3.4** `Node* EP::Equipment::getNodeParent ( ) [inline]`

## Returns

L'attribut `m_roomParent`.

**6.2.3.5** `static char* EP::Equipment::getPasswordFibaro ( ) [inline],[static]`

## Returns

Le password de la Fibaro.

**6.2.3.6** `int EP::Equipment::getTypeOf ( ) [inline]`

## Returns

L'attribut `m_typeOf`.

**6.2.3.7** `virtual int EP::Equipment::sendRequest ( ) [pure virtual]`

Envoie la requête HTTP permettant d'interagir avec l'équipement.

Implemented in [EP::EquipmentFibaro](#), and [EP::EquipmentKira](#).

**6.2.3.8** `int EP::Equipment::setIpFibaro ( char * new_ip ) [static]`

## Parameters

<i>new↔ _ip</i>	La nouvelle adresse IP de la Fibaro.
---------------------	--------------------------------------

## Returns

0 si l'adresse est incorrecte, 1 sinon.

**6.2.3.9** `int EP::Equipment::setIpKira ( char * new_ip )` `[static]`

## Parameters

<i>new↔ _ip</i>	La nouvelle adresse IP de la Kira.
---------------------	------------------------------------

## Returns

0 si l'adresse est incorrecte, 1 sinon.

**6.2.3.10** `int EP::Equipment::setLoginFibaro ( char * new_login )` `[static]`

## Parameters

<i>new_login</i>	Le nouveau login de la Fibaro.
------------------	--------------------------------

**6.2.3.11** `int EP::Equipment::setPasswordFibaro ( char * new_password )` `[static]`

## Parameters

<i>new_password</i>	Le nouveau password de la Fibaro.
---------------------	-----------------------------------

## 6.2.4 Member Data Documentation

**6.2.4.1** `char EP::Equipment::Fibaro_login = "admin"` `[static]`, `[protected]`

Le login de la Fibaro.

**6.2.4.2** `char EP::Equipment::Fibaro_password = "admin"` `[static]`, `[protected]`

Le password de la Fibaro.

**6.2.4.3** `char EP::Equipment::IP_Fibaro = "192.168.81.1"` `[static]`, `[protected]`

L'adresse IP de la Fibaro.

6.2.4.4 `char EP::Equipment::IP_Kira = "192.168.1.31" [static], [protected]`

L'adresse IP de la Kira.

6.2.4.5 `Node* EP::Equipment::m_roomParent [protected]`

Un pointeur vers la pièce contenant l'équipement.

6.2.4.6 `int EP::Equipment::m_typeOf [protected]`

1: pour un équipement lié à la Kira, 2: pour un équipement lié à la Fibaro.

The documentation for this class was generated from the following files:

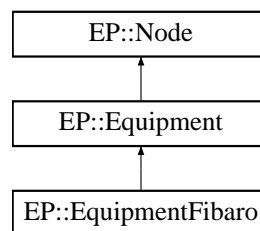
- ModelDll/include/Equipment.h
- ModelDll/src/Equipment.cpp

## 6.3 EP::EquipmentFibaro Class Reference

Représente un équipement lié à une Fibaro, hérite de [Equipment](#).

```
#include <Equipment.h>
```

Inheritance diagram for EP::EquipmentFibaro:



### Public Member Functions

- [EquipmentFibaro](#) (char \*name, char \*ico, [Node](#) \*parent, int [equipmentId](#), char \*action)  
Appelle le constructeur de [Equipment](#) et initialise m\_equipmentId et m\_action.
- virtual [~EquipmentFibaro](#) ()
- virtual int [sendRequest](#) ()  
Exécute la requête HTTP correspondant à cet équipement.
- int [getEquipmentId](#) ()
- char \* [getAction](#) ()

### Additional Inherited Members

#### 6.3.1 Detailed Description

Représente un équipement lié à une Fibaro, hérite de [Equipment](#).

#### 6.3.2 Constructor & Destructor Documentation

6.3.2.1 `EP::EquipmentFibaro::EquipmentFibaro ( char * name, char * ico, Node * parent, int equipmentId, char * action )`

Appelle le constructeur de [Equipment](#) et initialise m\_equipmentId et m\_action.

## Parameters

<i>equipment↵ Id</i>	L'identifiant correspondant à l'équipement dans l'interface web de la Fibaro.
<i>action</i>	L'action que l'équipement devra effectuer.

6.3.2.2 `EP::EquipmentFibaro::~~EquipmentFibaro ( )` [virtual]

### 6.3.3 Member Function Documentation

6.3.3.1 `char * EP::EquipmentFibaro::getAction ( )`

## Returns

L'action que l'équipement devra effectuer.

6.3.3.2 `int EP::EquipmentFibaro::getEquipmentId ( )`

## Returns

L'identifiant correspondant à l'équipement dans l'interface web de la Fibaro.

6.3.3.3 `int EP::EquipmentFibaro::sendRequest ( )` [virtual]

Exécute la requête HTTP correspondant à cet équipement.

Implements [EP::Equipment](#).

The documentation for this class was generated from the following files:

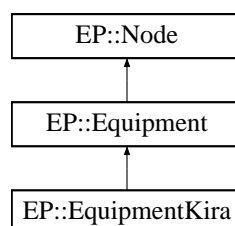
- ModelDll/include/[Equipment.h](#)
- ModelDll/src/[Equipment.cpp](#)

## 6.4 EP::EquipmentKira Class Reference

Représente un équipement lié à une Kira, hérite de [Equipment](#).

```
#include <Equipment.h>
```

Inheritance diagram for EP::EquipmentKira:



## Public Member Functions

- [EquipmentKira](#) (char \*name, char \*ico, [Node](#) \*parent, int buttonId, int page)  
*Appelle le constructeur de [Equipment](#) et initialise m\_buttonId et m\_page.*
- virtual [~EquipmentKira](#) ()
- virtual int [sendRequest](#) ()  
*Exécute la requête HTTP correspondant à cet équipement.*
- int [getButtonId](#) ()
- int [getPageNumber](#) ()
- int [setButtonId](#) (int new\_id)
- int [setPageNumber](#) (int new\_PageNumber)

## Additional Inherited Members

### 6.4.1 Detailed Description

Représente un équipement lié à une Kira, hérite de [Equipment](#).

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 EP::EquipmentKira::EquipmentKira ( char \* name, char \* ico, [Node](#) \* parent, int buttonId, int page )

Appelle le constructeur de [Equipment](#) et initialise m\_buttonId et m\_page.

##### Parameters

<i>buttonId</i>	Le numéro du bouton correspondant à l'équipement dans l'interface web de la Kira.
<i>page</i>	La page du bouton correspondant à l'équipement dans l'interface web de la Kira.

#### 6.4.2.2 EP::EquipmentKira::~~EquipmentKira ( ) [virtual]

### 6.4.3 Member Function Documentation

#### 6.4.3.1 int EP::EquipmentKira::getButtonId ( )

##### Returns

Le numéro du bouton correspondant à l'équipement dans l'interface web de la Kira.

#### 6.4.3.2 int EP::EquipmentKira::getPageNumber ( )

##### Returns

Le numéro de page du bouton correspondant à l'équipement dans l'interface web de la Kira.

#### 6.4.3.3 int EP::EquipmentKira::sendRequest ( ) [virtual]

Exécute la requête HTTP correspondant à cet équipement.

Implements [EP::Equipment](#).

#### 6.4.3.4 int EP::EquipmentKira::setButtonId ( int new\_id )

##### Parameters

<i>new_id</i>	Le nouveau numéro du bouton.
---------------	------------------------------

#### 6.4.3.5 int EP::EquipmentKira::setPageNumber ( int new\_PageNumber )

##### Parameters

<i>new_PageNumber</i>	Le nouveau numéro de page du bouton.
-----------------------	--------------------------------------

The documentation for this class was generated from the following files:

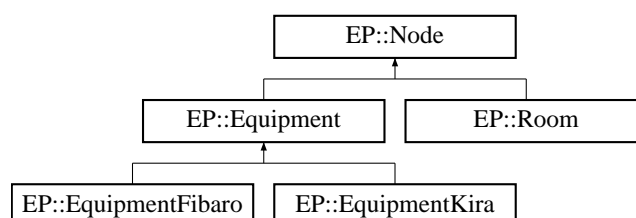
- ModelDll/include/[Equipment.h](#)
- ModelDll/src/[Equipment.cpp](#)

## 6.5 EP::Node Class Reference

Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes [Equipment](#) et [Room](#), et n'est donc pas utilisée telle qu'elle.

```
#include <Node.h>
```

Inheritance diagram for EP::Node:



## Public Member Functions

- [Node](#) (char \*name, char \*ico)  
*Constructeur, initialise les attributs m\_name et m\_ico.*
- virtual [~Node](#) (void)  
*Destructeur, ne fait rien de particulier.*
- char \* [getName](#) ()
- void [setName](#) (char \*name)  
*Change le nom de ce noeud pour celui passé en paramètre.*
- char \* [getlco](#) ()
- void [setlco](#) (char \*ico)  
*Change l'icône du noeud.*

## Protected Attributes

- char [m\\_name](#) [100]  
*Le nom du noeud.*
- char [m\\_ico](#) [100]  
*L'icône du noeud qui sera affichée dans l'application.*
- [EP::Node](#) \* [m\\_parent](#)  
*Le noeud parent de celui-ci. Sera nul pour les pièces ([Room](#)), et correspondra à la pièce contenant dans le cas d'un équipement ([Equipment](#)).*

### 6.5.1 Detailed Description

Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes [Equipment](#) et [Room](#), et n'est donc pas utilisée telle qu'elle.

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 EP::Node::Node ( char \* name, char \* ico )

Constructeur, initialise les attributs m\_name et m\_ico.

#### 6.5.2.2 EP::Node::~~Node ( void ) [virtual]

Destructeur, ne fait rien de particulier.

### 6.5.3 Member Function Documentation

#### 6.5.3.1 char \* EP::Node::getlco ( )

##### Returns

Le nom du fichier correspondant à l'icône de ce noeud.

### 6.5.3.2 `char * EP::Node::getName ( )`

#### Returns

Le nom du noeud.

### 6.5.3.3 `void EP::Node::setIco ( char * ico )`

Change l'icône du noeud.



## Parameters

<i>ico</i>	La nouvelle icône du noeud.
------------	-----------------------------

6.5.3.4 void EP::Node::setName ( char \* *name* )

Change le nom de ce noeud pour celui passé en paramètre.

## Parameters

<i>name</i>	Le nouveau nom du noeud.
-------------	--------------------------

## 6.5.4 Member Data Documentation

## 6.5.4.1 char EP::Node::m\_ico[100] [protected]

L'icône du noeud qui sera affichée dans l'application.

## 6.5.4.2 char EP::Node::m\_name[100] [protected]

Le nom du noeud.

## 6.5.4.3 EP::Node\* EP::Node::m\_parent [protected]

Le noeud parent de celui-ci. Sera nul pour les pièces ([Room](#)), et correspondra à la pièce contenant dans le cas d'un équipement ([Equipment](#)).

The documentation for this class was generated from the following files:

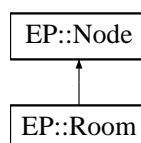
- ModelDll/include/[Node.h](#)
- ModelDll/src/[Node.cpp](#)

## 6.6 EP::Room Class Reference

Représente une pièce de la maison ; Hérite de [Node](#). On peut y retrouver les différents équipements liés à une pièce et les gérer.

```
#include <Room.h>
```

Inheritance diagram for EP::Room:



## Public Member Functions

- [Room](#) (char \*name, char \*ico)  
*Appelle le constructeur de [Node](#).*
- [~Room](#) ()  
*Détruit la liste d'équipements.*
- int [addEquipment](#) ([Equipment](#) \*equip)  
*Ajoute le pointeur d'un équipement donné dans m\_listEquipments. Le nom d'un équipement doit être unique.*
- int [deleteEquipmentByIndex](#) (int index)  
*Supprime l'équipement dont l'index est donné en paramètre.*
- int [deleteEquipmentByName](#) (char \*name)  
*Supprime l'équipement dont le nom est donné en paramètre.*
- std::vector< [Equipment](#) \* > \* [getEquipments](#) ()  
*Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec [getNumberEquipments](#) et [getEquipmentByIndex](#).*
- [Equipment](#) \* [getEquipmentByName](#) (char \*name)  
*Donne un pointeur vers l'équipement dont le nom est donné en paramètre.*
- [Equipment](#) \* [getEquipmentByIndex](#) (int index)  
*Donne un pointeur vers l'équipement dont l'index est donné en paramètre.*
- int [getNumberEquipments](#) ()

## Additional Inherited Members

### 6.6.1 Detailed Description

Représente une pièce de la maison ; Hérite de [Node](#). On peut y retrouver les différents équipements liés à une pièce et les gérer.

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 EP::Room::Room ( char \* name, char \* ico )

Appelle le constructeur de [Node](#).

#### 6.6.2.2 EP::Room::~~Room ( )

Détruit la liste d'équipements.

### 6.6.3 Member Function Documentation

#### 6.6.3.1 int EP::Room::addEquipment ( [Equipment](#) \* equip )

Ajoute le pointeur d'un équipement donné dans m\_listEquipments. Le nom d'un équipement doit être unique.

## Parameters

<i>equip</i>	Le pointeur de l'équipement à ajouter.
--------------	--

## Returns

0 si tout s'est bien passé, 1 si le nom est déjà pris.

**6.6.3.2 int EP::Room::deleteEquipmentByIndex ( int *index* )**

Supprime l'équipement dont l'index est donné en paramètre.

## Parameters

<i>index</i>	L'index de l'équipement à supprimer de m_listEquipments.
--------------	--

## Returns

0 si tout s'est bien passé, 1 si l'index ne correspond à aucun équipement.

**6.6.3.3 int EP::Room::deleteEquipmentByName ( char \* *name* )**

Supprime l'équipement dont le nom est donné en paramètre.

## Parameters

<i>name</i>	Le nom de l'équipement à supprimer de m_listEquipments.
-------------	---

## Returns

0 si tout s'est bien passé, 1 si le nom ne correspond à aucun équipement.

**6.6.3.4 Equipment \* EP::Room::getEquipmentByIndex ( int *index* )**

Donne un pointeur vers l'équipement dont l'index est donné en paramètre.

## Parameters

<i>index</i>	L'index de l'équipement qu'on veut récupérer.
--------------	---

## Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

#### 6.6.3.5 **Equipment** \* EP::Room::getEquipmentByName ( char \* *name* )

Donne un pointeur vers l'équipement dont le nom est donné en paramètre.

##### Parameters

<i>name</i>	Le nom de l'équipement qu'on veut récupérer.
-------------	--

##### Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

#### 6.6.3.6 **vector< Equipment \* >** \* EP::Room::getEquipments ( )

Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec `getNumberEquipments` et `getEquipmentByIndex`.

##### Returns

L'attribut `m_listEquipments`.

#### 6.6.3.7 **int** EP::Room::getNumberEquipments ( )

##### Returns

Le nombre d'équipement que la pièce contient

The documentation for this class was generated from the following files:

- ModelDll/include/[Room.h](#)
- ModelDll/src/[Room.cpp](#)

## Chapter 7

# File Documentation

### 7.1 ModelDll/include/Core.h File Reference

```
#include "Room.h"
```

#### Classes

- class [EP::Core](#)

*Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.*

#### Namespaces

- [EP](#)

#### Macros

- #define [DllExport](#) \_\_declspec(dllexport)
- #define [FILESAVE\\_NAME\\_SIZE](#) 500

#### Functions

- [DllExport](#) Core \* [EP::Core\\_New](#) (char \*file)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Core](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- [DllExport](#) Core \* [EP::Core\\_NewFromSave](#) (char \*file)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Core](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur. Après création de l'objet, la méthode [Core::load\(\)](#) est appelée sur celui-ci.*
- [DllExport](#) void [EP::Core\\_Delete](#) (Core \*core)  
*Destructeur statique, pour permettre la destruction des objets [Core](#) en passant par la DLL.*
- [DllExport](#) void [EP::Core\\_SaveAndDelete](#) (Core \*core)  
*Destructeur statique, pour permettre la destruction des objets [Core](#) en passant par la DLL. La méthode [Core::save\(\)](#) est appelée avant sa destruction.*

## 7.1.1 Macro Definition Documentation

7.1.1.1 `#define DllExport __declspec(dllexport)`

7.1.1.2 `#define FILESAVE_NAME_SIZE 500`

## 7.2 ModelDll/include/Equipment.h File Reference

```
#include "Node.h"
```

### Classes

- class [EP::Equipment](#)  
*Représente un équipement. Cette classe abstraite est spécialisée par [EquipmentKira](#) et [EquipmentFibaro](#).*
- class [EP::EquipmentKira](#)  
*Représente un équipement lié à une Kira, hérite de [Equipment](#).*
- class [EP::EquipmentFibaro](#)  
*Représente un équipement lié à une Fibaro, hérite de [Equipment](#).*

### Namespaces

- [EP](#)

### Macros

- `#define DllExport __declspec(dllexport)`

### Functions

- [DllExport](#) char \* [EP::Equipment\\_getIpKira](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) char \* [EP::Equipment\\_getIpFibaro](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) int [EP::Equipment\\_setIpKira](#) (char \*new\_ip)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) int [EP::Equipment\\_setIpFibaro](#) (char \*new\_ip)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) char \* [EP::Equipment\\_getLoginFibaro](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) char \* [EP::Equipment\\_getPasswordFibaro](#) ()  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) int [EP::Equipment\\_setLoginFibaro](#) (char \*new\_login)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*
- [DllExport](#) int [EP::Equipment\\_setPasswordFibaro](#) (char \*new\_password)  
*Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.*

- **DllExport** `EquipmentKira * EP::EquipmentKira_New` (`char *name`, `char *ico`, `Node *parent`, `int buttonId`, `int page`)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [EquipmentKira](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** `void EP::EquipmentKira_Delete` (`EquipmentKira *eq`)  
*Destructeur statique, pour permettre la destruction des objets [EquipmentKira](#) en passant par la DLL.*
- **DllExport** `EquipmentFibaro * EP::EquipmentFibaro_New` (`char *name`, `char *ico`, `Node *parent`, `int equipmentId`, `char *action`)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [EquipmentFibaro](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** `void EP::EquipmentFibaro_Delete` (`EquipmentFibaro *eq`)  
*Destructeur statique, pour permettre la destruction des objets [EquipmentFibaro](#) en passant par la DLL.*

### 7.2.1 Macro Definition Documentation

7.2.1.1 `#define DllExport __declspec(dllexport)`

## 7.3 ModelDll/include/Node.h File Reference

### Classes

- class [EP::Node](#)  
*Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes [Equipment](#) et [Room](#), et n'est donc pas utilisée telle qu'elle.*

### Namespaces

- [EP](#)

### Macros

- `#define DllExport __declspec(dllexport)`

### Functions

- **DllExport** `Node * EP::Node_New` (`char *name`, `char *ico`)  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Node](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- **DllExport** `void EP::Node_Delete` (`Node *node`)  
*Destructeur statique, pour permettre la destruction des objets [Node](#) en passant par la DLL.*

### 7.3.1 Macro Definition Documentation

7.3.1.1 `#define DllExport __declspec(dllexport)`

## 7.4 ModelDll/include/RequeteHttp.h File Reference

### Namespaces

- [EP](#)

## Macros

- `#define DllImport __declspec(dllimport)`

## Functions

- `void DllImport EP::requeteHttpKira (char *s1, char *s2)`  
*Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Kira. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode [EquipmentKira::sendRequest\(\)](#) ; les paramètres sont construits automatiquement à partir des informations du l'équipement.*
- `void DllImport EP::requeteHttpFibaro (char *s1, char *s2, char *user, char *pass)`  
*Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Fibaro. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode [EquipmentFibaro::sendRequest\(\)](#) ; les paramètres sont construits automatiquement à partir des informations du l'équipement.*

### 7.4.1 Macro Definition Documentation

#### 7.4.1.1 `#define DllImport __declspec(dllimport)`

## 7.5 ModelDll/include/Room.h File Reference

```
#include <vector>
#include "Equipment.h"
```

## Classes

- class [EP::Room](#)  
*Représente une pièce de la maison ; Hérite de [Node](#). On peut y retrouver les différents équipements liés à une pièce et les gérer.*

## Namespaces

- [EP](#)

## Macros

- `#define DllExport __declspec(dllexport)`

## Functions

- `DllExport Room * EP::Room_New (char *name, char *ico)`  
*Constructeur statique utilisé pour permettre l'utilisation des objets [Room](#) en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.*
- `DllExport void EP::Room_Delete (Room *room)`  
*Destructeur statique, pour permettre la destruction des objets [Room](#) en passant par la DLL.*



### 7.5.1 Macro Definition Documentation

#### 7.5.1.1 #define DllExport \_\_declspec(dllexport)

## 7.6 ModelDll/src/Core.cpp File Reference

```
#include "../include/Core.h"
#include <fstream>
#include <string>
#include <iostream>
```

### Namespaces

- [EP](#)

### Functions

- [EP::\\_\\_declspec](#) (dllexport) Core \*Core\_New(char \*file)

## 7.7 ModelDll/src/Equipment.cpp File Reference

```
#include "../include/Equipment.h"
#include "../include/RequeteHttp.h"
#include <iostream>
```

### Namespaces

- [EP](#)

### Functions

- [EP::\\_\\_declspec](#) (dllexport) Core \*Core\_New(char \*file)

### Variables

- char \* [EP::ico](#)
- char Node \* [EP::parent](#)
- char Node int [EP::buttonId](#)
- char Node int int [EP::page](#)
- char Node int [EP::equipmentId](#)
- char Node int char \* [EP::action](#)

## 7.8 ModelDll/src/Node.cpp File Reference

```
#include "../include/Node.h"  
#include <iostream>
```

### Namespaces

- [EP](#)

### Functions

- [EP::\\_\\_declspec](#) (dllexport) Core \*Core\_New(char \*file)

## 7.9 ModelDll/src/Room.cpp File Reference

```
#include "../include/Room.h"
```

### Namespaces

- [EP](#)

### Functions

- [EP::\\_\\_declspec](#) (dllexport) Core \*Core\_New(char \*file)

# Index

- \_\_declspec
    - EP, [11](#)
  - ~Core
    - EP::Core, [20](#)
  - ~Equipment
    - EP::Equipment, [25](#)
  - ~EquipmentFibaro
    - EP::EquipmentFibaro, [28](#)
  - ~EquipmentKira
    - EP::EquipmentKira, [29](#)
  - ~Node
    - EP::Node, [31](#)
  - ~Room
    - EP::Room, [34](#)
- action
  - EP, [16](#)
- addEquipment
  - EP::Room, [34](#)
- addRoom
  - EP::Core, [20](#)
- buttonId
  - EP, [16](#)
- Core
  - EP::Core, [20](#)
- Core.h
  - DllExport, [38](#)
  - FILESAVE\_NAME\_SIZE, [38](#)
- Core\_Delete
  - EP, [11](#)
- Core\_New
  - EP, [11](#)
- Core\_NewFromSave
  - EP, [11](#)
- Core\_SaveAndDelete
  - EP, [11](#)
- deleteEquipmentByIndex
  - EP::Room, [35](#)
- deleteEquipmentByName
  - EP::Room, [35](#)
- deleteRoomByIndex
  - EP::Core, [20](#)
- deleteRoomByName
  - EP::Core, [21](#)
- DllExport
  - Core.h, [38](#)
  - Equipment.h, [39](#)

- Node.h, [39](#)
- Room.h, [41](#)
- DllImport
  - RequeteHttp.h, [40](#)
- EP::Core, [19](#)
  - ~Core, [20](#)
  - addRoom, [20](#)
  - Core, [20](#)
  - deleteRoomByIndex, [20](#)
  - deleteRoomByName, [21](#)
  - getCOMPort, [21](#)
  - getFileSave, [21](#)
  - getIconSize, [21](#)
  - getNumberRooms, [21](#)
  - getRoomByIndex, [21](#)
  - getRoomByName, [22](#)
  - getRooms, [22](#)
  - getThemeld, [22](#)
  - load, [22](#)
  - save, [22](#)
  - setCOMPort, [22](#)
  - setFileSave, [23](#)
  - setIconSize, [23](#)
  - setThemeld, [23](#)
- EP::Equipment, [23](#)
  - ~Equipment, [25](#)
  - Equipment, [24](#)
  - Fibaro\_login, [26](#)
  - Fibaro\_password, [26](#)
  - getIpFibaro, [25](#)
  - getIpKira, [25](#)
  - getLoginFibaro, [25](#)
  - getNodeParent, [25](#)
  - getPasswordFibaro, [25](#)
  - getTypeOf, [25](#)
  - IP\_Fibaro, [26](#)
  - IP\_Kira, [26](#)
  - m\_roomParent, [27](#)
  - m\_typeOf, [27](#)
  - sendRequest, [25](#)
  - setIpFibaro, [25](#)
  - setIpKira, [26](#)
  - setLoginFibaro, [26](#)
  - setPasswordFibaro, [26](#)
- EP::EquipmentFibaro, [27](#)
  - ~EquipmentFibaro, [28](#)
  - EquipmentFibaro, [27](#)
  - getAction, [28](#)
  - getEquipmentId, [28](#)

- sendRequest, 28
- EP::EquipmentKira, 28
  - ~EquipmentKira, 29
  - EquipmentKira, 29
  - getButtonId, 29
  - getPageNumber, 29
  - sendRequest, 29
  - setButtonId, 30
  - setPageNumber, 30
- EP::Node, 30
  - ~Node, 31
  - getIco, 31
  - getName, 31
  - m\_ico, 33
  - m\_name, 33
  - m\_parent, 33
  - Node, 31
  - setIco, 32
  - setName, 33
- EP::Room, 33
  - ~Room, 34
  - addEquipment, 34
  - deleteEquipmentByIndex, 35
  - deleteEquipmentByName, 35
  - getEquipmentByIndex, 35
  - getEquipmentByName, 35
  - getEquipments, 36
  - getNumberEquipments, 36
  - Room, 34
- EP, 9
  - \_\_declspec, 11
  - action, 16
  - buttonId, 16
  - Core\_Delete, 11
  - Core\_New, 11
  - Core\_NewFromSave, 11
  - Core\_SaveAndDelete, 11
  - Equipment\_getIpFibaro, 11
  - Equipment\_getIpKira, 12
  - Equipment\_getLoginFibaro, 12
  - Equipment\_getPasswordFibaro, 12
  - Equipment\_setIpFibaro, 12
  - Equipment\_setIpKira, 12
  - Equipment\_setLoginFibaro, 13
  - Equipment\_setPasswordFibaro, 13
  - EquipmentFibaro\_Delete, 13
  - EquipmentFibaro\_New, 13
  - equipmentId, 16
  - EquipmentKira\_Delete, 13
  - EquipmentKira\_New, 15
  - ico, 16
  - Node\_Delete, 15
  - Node\_New, 15
  - page, 16
  - parent, 17
  - requestHttpFibaro, 15
  - requestHttpKira, 15
  - Room\_Delete, 16
  - Room\_New, 16
- Equipment
  - EP::Equipment, 24
- Equipment.h
  - DllExport, 39
- Equipment\_getIpFibaro
  - EP, 11
- Equipment\_getIpKira
  - EP, 12
- Equipment\_getLoginFibaro
  - EP, 12
- Equipment\_getPasswordFibaro
  - EP, 12
- Equipment\_setIpFibaro
  - EP, 12
- Equipment\_setIpKira
  - EP, 12
- Equipment\_setLoginFibaro
  - EP, 13
- Equipment\_setPasswordFibaro
  - EP, 13
- EquipmentFibaro
  - EP::EquipmentFibaro, 27
- EquipmentFibaro\_Delete
  - EP, 13
- EquipmentFibaro\_New
  - EP, 13
- equipmentId
  - EP, 16
- EquipmentKira
  - EP::EquipmentKira, 29
- EquipmentKira\_Delete
  - EP, 13
- EquipmentKira\_New
  - EP, 15
- FILESAVE\_NAME\_SIZE
  - Core.h, 38
- Fibaro\_login
  - EP::Equipment, 26
- Fibaro\_password
  - EP::Equipment, 26
- getAction
  - EP::EquipmentFibaro, 28
- getButtonId
  - EP::EquipmentKira, 29
- getCOMPort
  - EP::Core, 21
- getEquipmentByIndex
  - EP::Room, 35
- getEquipmentByName
  - EP::Room, 35
- getEquipmentId
  - EP::EquipmentFibaro, 28
- getEquipments
  - EP::Room, 36
- getFileSave
  - EP::Core, 21

- getIco
  - EP::Node, [31](#)
- getIconSize
  - EP::Core, [21](#)
- getIpFibaro
  - EP::Equipment, [25](#)
- getIpKira
  - EP::Equipment, [25](#)
- getLoginFibaro
  - EP::Equipment, [25](#)
- getName
  - EP::Node, [31](#)
- getNodeParent
  - EP::Equipment, [25](#)
- getNumberEquipments
  - EP::Room, [36](#)
- getNumberRooms
  - EP::Core, [21](#)
- getPageNumber
  - EP::EquipmentKira, [29](#)
- getPasswordFibaro
  - EP::Equipment, [25](#)
- getRoomByIndex
  - EP::Core, [21](#)
- getRoomByName
  - EP::Core, [22](#)
- getRooms
  - EP::Core, [22](#)
- getThemeld
  - EP::Core, [22](#)
- getTypeOf
  - EP::Equipment, [25](#)
- IP\_Fibaro
  - EP::Equipment, [26](#)
- IP\_Kira
  - EP::Equipment, [26](#)
- ico
  - EP, [16](#)
- load
  - EP::Core, [22](#)
- m\_ico
  - EP::Node, [33](#)
- m\_name
  - EP::Node, [33](#)
- m\_parent
  - EP::Node, [33](#)
- m\_roomParent
  - EP::Equipment, [27](#)
- m\_typeOf
  - EP::Equipment, [27](#)
- ModelDll/include/Core.h, [37](#)
- ModelDll/include/Equipment.h, [38](#)
- ModelDll/include/Node.h, [39](#)
- ModelDll/include/RequeteHttp.h, [39](#)
- ModelDll/include/Room.h, [40](#)
- ModelDll/src/Core.cpp, [41](#)
- ModelDll/src/Equipment.cpp, [41](#)
- ModelDll/src/Node.cpp, [42](#)
- ModelDll/src/Room.cpp, [42](#)
- Node
  - EP::Node, [31](#)
- Node.h
  - DllExport, [39](#)
- Node\_Delete
  - EP, [15](#)
- Node\_New
  - EP, [15](#)
- page
  - EP, [16](#)
- parent
  - EP, [17](#)
- RequeteHttp.h
  - DllImport, [40](#)
- requeteHttpFibaro
  - EP, [15](#)
- requeteHttpKira
  - EP, [15](#)
- Room
  - EP::Room, [34](#)
- Room.h
  - DllExport, [41](#)
- Room\_Delete
  - EP, [16](#)
- Room\_New
  - EP, [16](#)
- save
  - EP::Core, [22](#)
- sendRequest
  - EP::Equipment, [25](#)
  - EP::EquipmentFibaro, [28](#)
  - EP::EquipmentKira, [29](#)
- setButtonId
  - EP::EquipmentKira, [30](#)
- setCOMPort
  - EP::Core, [22](#)
- setFileSave
  - EP::Core, [23](#)
- setIco
  - EP::Node, [32](#)
- setIconSize
  - EP::Core, [23](#)
- setIpFibaro
  - EP::Equipment, [25](#)
- setIpKira
  - EP::Equipment, [26](#)
- setLoginFibaro
  - EP::Equipment, [26](#)
- setName
  - EP::Node, [33](#)
- setPageNumber
  - EP::EquipmentKira, [30](#)

setPasswordFibaro  
    EP::Equipment, [26](#)  
setThemeld  
    EP::Core, [23](#)