

WireLess Deployer

Guide de référence des paquets

Introduction.....	3
Package Descriptor	3
Section PACKAGE :.....	3
Type	3
Id	4
Version.....	4
Name.....	4
FileList	4
UninstallList.....	4
Persistent.....	5
Section INFO :.....	5
Description.....	5
Platform.....	5
Model	6
Vendor.....	6
Section APPLICATION :	6
StartExe.....	6
Section CONFIG :	7
Filelist	7
Exemple de fichier descripteur d'un paquet d'application (application package descriptor) :	7
Exemple de fichier descripteur d'un paquet de configuration (configuration package descriptor) :	8
FileList	9
Section DOWNLOAD :	9
Section MAIN :	9
Sections ACTION :	9
Variables de l'Unité	11
WDP Files.....	12
Section MAIN :	13
Sections ACTION :	13
Guide des actions	15
ACTION-ZIP	16
ACTION-CPY	16
ACTION-ATTRIB.....	17
ACTION-REG	17
ACTION-DLL	18
ACTION-EXE	18
ACTION-WAIT-EXE.....	19
ACTION-REGEXP.....	19
ACTION-DEL	20
ACTION-REMOVE	21

ACTION-KILL.....	22
ACTION-DELAY.....	22
ACTION-BOOT	23
Annexe A : création des fichiers CPY	24

Introduction

Ce document décrit les fichiers impliqués dans le chargement des paquets avec WireLess Déployer :

- le **fichier descripteur du paquet** (Id-Version.pckg)
- le **fichier d'installation** (le nom de ce fichier varie)
- le **fichier de désinstallation** (le nom de ce fichier varie)

Ces fichiers peuvent être modifiés avec le Packet Express ou à la main (non conseillé). La version actuelle du Packet Express permet de construire des **paquets d'application** uniquement.

Package Descriptor

Ce fichier –placé sur `X:\Program Files\WireLessDeployer\Packages-` possède les informations sur le paquet.

Le nom de ce fichier est composé du numéro d'Id et du numéro de version du paquet : *Id-Version.pckg*

L'*Id* et la *version* **doivent coïncider** avec ceux de la section PACKAGE, autrement, le paquet ne sera pas valide.

Il y a deux types de paquet : des paquets d'application et des paquets de configuration.

- Les fichiers qui décrivent des paquets d'**application** contiennent les sections PACKAGE, INFO et APPLICATION.
- Les fichiers qui décrivent des paquets de **configuration** contiennent les sections PACKAGE, INFO et CONFIG.

Section PACKAGE :

Type

Type = A | C

A = application

C = configuration

Exemple :
Type = A

Id

Identifiant.

La combinaison Id-Version identifie un paquet univoquement. Cet Id est aussi le nom du répertoire et du fichier du paquet, de sorte que cet Id peut être uniquement conformé par des caractères permis pour des répertoires et des fichiers de Windows.

Version

Version = Major-minor-revision

Exemple :
Version = 1-2-3

Name

Nom du paquet. Si c'est un fichier d'application, le nom du paquet est affiché à l'utilisateur sur la Console WDp et sur le Launcher.

FileList

Nom du fichier d'installation. Ce fichier contient une liste des fichiers à charger et des actions à exécuter (voir **FileList**). Ce fichier est requis au moment de l'installation du paquet.

Exemple :
FileList = install.lstdnl

UninstallList

Nom du fichier de désinstallation. Ce fichier peut avoir une liste des fichiers à charger et des actions à exécuter (voir **FileList**). L'UninstallList est requise au moment de la désinstallation du paquet.

Exemple :
FileList = uninstall.lstdnl

Persistent

persistent = yes | no

Cette option permet de spécifier si l'information du paquet sera persistante au colboot. Pour que les fichiers d'un paquet soient persistants, il faut que le contenu du paquet soit enregistré sur un **répertoire flash persistant** au coldboot. Utilisez des ACTIONS, comme [ACTION-CPY] (voir la **Guide des Actions**) ou installez les fichiers dans le répertoire flash directement.

Section INFO :

Cette section n'a que des renseignements puisqu'elle est utilisée pour afficher de des informations du paquet sur la Console WDP. Aucun de ces champs ne sera utilisé sur le PDA.

Description

Description du paquet. Cette option est utilisée à titre indicatif.

Platform

WINCE410 | WINCE420 | WINCE50 | WINCE2003 | WINCE2005

Plate-forme où sera installé le paquet.

WINCE410 : Windows CE 4.10

WINCE420 : Windows CE 4.20

WINCE50 : Windows CE 5.0

WINCE2003 : Windows Mobile 2003 / Pocket PC 2003

WINCE2005 : Windows Mobile 5.0

La Console WDP et l'Agent utilisent cette valeur dans les collections et les formules pour évaluer les conditions de chargement des paquets.

Il est convenable –mais non obligatoire– que ces informations correspondent à la version de système d'exploitation que les PDAs qui chargeront le paquet ont vraiment. Le Client ne vérifie pas cette valeur.

Exemple :

Platform = WINCE50

Model

Nom du modèle ou nom standard du PDA.

La Console WDP et l'Agent utilisent cette valeur dans les collections et les formules pour évaluer les conditions de chargement des paquets.

Il est convenable –mais non obligatoire– que ces informations correspondent au modèle que les PDAs qui chargeront le paquet ont vraiment. Le Client ne vérifie pas cette valeur.

Exemple :

Model = mc9090

Vendor

Ce champ indique le créateur du paquet. Cette option est utilisée à titre indicatif.

Exemple :

vendor = SofToGo SA

Section APPLICATION :

Cette section est facultative et existe uniquement dans les paquets d'application exécutés par le Launcher.

StartExe

Chemin complet vers un raccourci ou vers un fichier exécutable sur le PDA.

Voyez la section [ACTION-EXE] dans le chapitre **WDP files**.

L'icône affiché sur le Launcher est extrait de ce fichier exécutable. S'il n'y a aucun icône disponible, ou si le fichier n'est pas un exe, un icône standard sera affiché.

Exemple :

StartExe = \windows\calc.exe

Section CONFIG :

Cette section est facultative et existe uniquement dans les paquets de configuration. Elle permet à l'utilisateur de modifier des fichiers des paquets sur la Console WDp. Les fichiers inclus dans cette liste ne seront pas chargés sur le PDA.

Filelist

Nom du fichier qui contient la liste de configuration.

Exemple :

Filelist = config.lstcfg

Exemple de fichier descripteur d'un paquet d'application (application package descriptor) :

```
# APP_SETUP-1-0-0.pkg

[PACKAGE]
type=A
id= APP_SETUP
name= APP setup
version=1-0-0
filelist=install.lstdn1
uninstalllist=uninstall.lstdn1
persistent=yes

[INFO]
description= APP application setup
platform=WINCE2005
model=MC9090
vendor=Softogo Packet Express

[APPLICATION]
StartExe= \windows\app.exe
```

Exemple de fichier descripteur d'un paquet de configuration (configuration package descriptor) :

```
# APP_CONFIG-1-0-0.pckg

[PACKAGE]
type=c
id= APP_CONFIG
name= APP configuration
version=1-0-0
filelist=install.lstdnl
uninstalllist=uninstall.lstdnl
persistent=yes

[INFO]
description= APP configuration
platform=WINCE2005
model=MC9090
vendor=Softogo Packet Express

[CONFIG]
filelist = config.lstcfg
```

FileList

Ce fichier contient la liste de fichiers à charger et des actions à exécuter après le processus de chargement.

Le fichier est composé de trois sections : MAIN, DOWNLOAD y ACTIONS.

Section MAIN :

```
[main]
sequentialActions = yes | no
```

L'option *SequentialActions* permet d'exécuter les actions dans l'ordre qu'elles ont dans le fichier. Si cette option n'est pas activée ou spécifiée, les actions seront exécutées suivant un ordre prédéterminé (voir Sections ACTION).

Section DOWNLOAD :

Détermine les fichiers qui seront chargés.

```
[download]
filecount = nombre de lignes (NN+1)
file00    = fichier source > fichier de destination
...
fileNN    = source > destination
```

La source et la destination doivent être séparées par un signe ">".

Source : chemin vers le fichier sur le serveur.
Destination : chemin vers le fichier sur le PDA.

Il n'est pas nécessaire d'utiliser des guillemets dans ces chemins.

Sections ACTION :

Ces sections permettent de définir des actions à exécuter après le processus de chargement.

Les actions supportées sont :

ACTION-ZIP, ACTION-CPY, ACTION-ATTRIB, ACTION-REG, ACTION-DLL,
ACTION-EXE, ACTION-REGEXP, ACTION-DEL, ACTION-REMOVE, ACTION-KILL,
ACTION-DELAY and ACTION-BOOT.

Voyez la **Guide des actions** pour avoir une description plus détaillée sur chaque action.

Il y a deux modes de traitement des actions : séquentiel et non séquentiel.

Dans le mode non séquentiel, les actions sont traitées suivant l'ordre ci-dessous :

- 1) ACTION-ZIP
- 2) ACTION-CPY
- 3) ACTION-ATTRIB
- 4) ACTION-REG
- 5) ACTION-DLL
- 6) ACTION-EXE
- 7) ACTION-REGEXP
- 8) ACTION-DEL
- 9) ACTION-REMOVE
- 10) ACTION-KILL
- 11) ACTION-DELAY
- 12) ACTION-BOOT

Pour utiliser activer le mode non séquentiel, désactivez l'option *sequentialActions* dans la section MAIN.

Dans le mode séquentiel, les actions sont traitées suivant l'ordre qu'elles ont dans le fichier, à l'exception d'ACTION-BOOT, qui est toujours exécutée à la fin, n'importe quelle soit sa position. Pour utiliser ce mode d'exécution, activez l'option *sequentialActions* dans la section MAIN.

Variables de l'Unité

Il existe un groupe de variables qui peuvent être utilisées dans la section DOWNLOAD d'un fichier FileList. Ces variables caractérisent une unité.

Les variables sont :

```
%HOSTNAME%
%IP%
%MAC%
%MUID%
%UID%
%MODEL%
%PLATFORM%
```

Le client cherche ces variables dans la section DOWNLOAD et les remplace par la valeur réelle. La recherche est effectuée sur la partie « source » de la section, c'est-à-dire, sur le chemin du fichier distant.

Par exemple, s'il y a une unité avec les valeurs ci-dessous :

```
HOSTNAME = PDA001
IP        = 192.168.1.9
MAC       = AA:BB:CC:DD:EE:FF
MUID      = ABC123DEF2345
UID       = john
MODEL    = MC9090
PLATFORM  = WINCE50
```

et le fichier FileList d'un paquet contient les valeurs suivantes :

```
[DOWNLOAD]
filecount = 4
file00    = PDA\App\%MODEL%\app.exe > \Application\App\app.exe
file01    = PDA\Cfg\%HOSTNAME%.cfg > \Application\App\app.cfg
file02    = PDA\%UID%\file1.txt > \Application\App\file1.txt
file03    = PDA\%UID%\file2.txt > \Application\App\file2.txt
```

le client WDP remplacera les variables comme ci-dessous :

```
PDA\App\%MODEL%\app.exe      PDA\App\MC9090\app.exe
PDA\Cfg\%HOSTNAME%.cfg      PDA\Cfg\PDA001.cfg
PDA\%UID%\file1.txt         PDA\john\file1.txt
PDA\%UID%\file2.txt         PDA\john\file2.txt
```

WDP Files

Les fichiers .wdp contiennent une liste d'actions à exécuter lors du démarrage du PDA. Le client cherche ces fichiers sur le répertoire flash du PDA au moment du démarrage. Après avoir trouvé les fichiers, le client les exécute l'un après l'autre. Il existe plusieurs niveaux de fichiers wdp :

.wdp, .wdp0, .wdp1, .wdp2, .wdp3, .wdp4 et .wdp5

Après un **coldboot**, sont traités les fichiers :

.wdp, .wdp0, .wdp1, .wdp2, .wdp3, .wdp4 et .wdp5.

Après un **warmboot**, sont traités les fichiers :

.wdp2, .wdp3, .wdp4 et .wdp5.

Ces fichiers sont exécutés en ordre numérique ascendant, c'est-à-dire, le fichier wdp sera exécuté au début et le fichier wdp5 sera exécuté à la fin. Si deux ou plusieurs fichiers ont le même numéro (par exemple "cfg.wdp2" et "app.wdp2") l'ordre est indéterminé.

Il peut avoir plusieurs actions ACTION-BOOT pendant le traitement des fichiers wdp. Pourtant, une seule de ces actions sera exécutée. La sélection de l'action ACTION-BOOT qui sera exécutée dépende des paramètres *type* et *immediate*, et se réalise selon les règles ci-dessous :

- Si *immediate=yes*, l'action est exécutée immédiatement après le traitement du fichier wdp.
- Si les deux actions, COLDBOOT et WARMBOOT, sont trouvées, COLDBOOT sera exécutée.
- S'il y a plusieurs WARMBOOT, celui qui ait le delay le plus long sera exécuté.
- S'il y a plusieurs COLDBOOT, celui qui ait le delay le plus long sera exécuté.

L'action ACTION-BOOT doit être utilisée avec attention, parce qu'elle peut provoquer un loop infini sur le PDA. Par conséquent, il n'est pas conseillé de l'utiliser dans des fichiers wdp qui soient exécutés après un warmboot (par exemple, wdp2, wdp3, wdp4 et wdp5).

Le client Wdp peut être utilisé comme **installateur** en exécutant un fichier wdp5. Si le client trouve un fichier wdp5 il sera fermé, encore si le fichier est vide ou une action échoue.

Le client Wdp crée un fichier log ("\\WdpInstallerLog.txt") où les actions sont écrites. Ce fichier peut être utilisé pour le débogage.

Les fichiers wdp ont deux sections : MAIN et ACTIONS.

Section MAIN :

```
[main]
sequentialActions = yes | no
autodelete = yes | no
delayBeforeRunning = temps d'attente en millisecondes
```

L'option *SequentialActions* permet d'exécuter les actions dans l'ordre qu'elles ont dans le fichier. Si cette option n'est pas activée ou spécifiée, les actions seront exécutées suivant un ordre prédéterminé (voir Sections ACTION).

L'option *Autodelete* permet d'effacer le fichier wdp après avoir exécuté toutes les actions.

L'option *DelayBeforeRunning* permet de définir la durée du temps d'attente avant l'exécution des actions.

Sections ACTION :

Ces sections permettent de définir des actions à exécuter.
Les actions supportées sont :

ACTION-ZIP, ACTION-CPY, ACTION-ATTRIB, ACTION-REG, ACTION-DLL, ACTION-EXE, ACTION-WAIT-EXE, ACTION-REGEXP, ACTION-DEL, ACTION-REMOVE, ACTION-KILL, ACTION-DELAY et ACTION-BOOT.

Voyez la **Guide des actions** pour avoir une description plus détaillée sur chaque action.

Il y a deux modes de traitement des actions : séquentiel et non séquentiel.

Dans le mode non séquentiel, les actions sont traitées suivant l'ordre ci-dessous :

- 13) ACTION-ZIP
- 14) ACTION-CPY
- 15) ACTION-ATTRIB
- 16) ACTION-REG
- 17) ACTION-DLL
- 18) ACTION-EXE
- 19) ACTION-REGEXP
- 20) ACTION-DEL

- 21) ACTION-REMOVE
- 22) ACTION-KILL
- 23) ACTION-DELAY
- 24) ACTION-BOOT

Pour utiliser activer le mode non séquentiel, désactivez l'option *sequentialActions* dans la section MAIN.

Dans le mode séquentiel, les actions sont traitées suivant l'ordre qu'elles ont dans le fichier, à l'exception d'ACTION-BOOT, qui est toujours exécutée à la fin, n'importe quelle soit sa position. Pour utiliser ce mode d'exécution, activez l'option *sequentialActions* dans la section MAIN.

Guide des actions

Celle-ci est la liste des actions possibles :

ACTION-ZIP
ACTION-CPY
ACTION-ATTRIB
ACTION-REG
ACTION-DLL
ACTION-EXE
ACTION-WAIT-EXE
ACTION-REGEXP
ACTION-DEL
ACTION-REMOVE
ACTION-KILL
ACTION-DELAY
ACTION-BOOT

Toutes les actions ont la même structure, à l'exception d'ACTION-BOOT et ACTION-DELAY (voir plus bas) :

```
[action-name]
stopOnError = yes | no
filecount   = number of files (NN+1)
file00      = file and/or parameters
...
fileNN      = file and/or parameters
```

Option StopOnError :

Cette option permet de contrôler l'exécution des actions : si l'action échoue et cette option est activé, les actions suivantes ne seront pas exécutées. Cela se rend utile quand certaines actions dépendent de la réussite des actions précédentes.

ACTION-ZIP

Description :

Permet de décompresser des fichiers. Une fois décompressé, le fichier n'est pas supprimé.

Syntaxe :

fileXX = nom du fichier

Si le chemin ou le nom du fichier contient des espaces à blanc, utilisez des guillemets.

Les arguments sont optionnels et n'ont pas besoin de guillemets.

Exemple :

```
[action-zip]
filecount =2
file00 = \flash\application.zip
file01 = \configuration.zip
```

ACTION-CPY

Description :

Applique un fichier cpy. Voyez l'annexe "Création des fichiers cpy" pour en avoir plus des renseignements.

Syntaxe :

fileXX = nom du fichier

Exemple :

```
[action-cpy]
filecount = 1
file00 = \flash\persist.copy
```

ACTION-ATTRIB

Description :

Modifie les attributs du fichier. Cette action accepte des caractères joker ("?" y "*"").

Syntaxe :

fileXX = nom du fichier [-r | +r] [-h | +h]

Si le chemin ou le nom du fichier contient des espaces à blanc, utilisez des guillemets.

Les arguments sont obligatoires et n'ont pas besoin de guillemets.

Paramètres :

+r/-r: active/désactive l'attribut "lecture seule"

+h/-h: active/désactive l'attribut "caché"

Exemple :

```
[action-attrib]
filecount = 4
file00 = \flash\myapp\file.txt -r
file01 = \flash\*.cfg +r
file02 = \myapp.exe -r -h
file03 = \flash\backup +r +h
```

ACTION-REG

Description :

Importe des clés de registre à partir d'un fichier.

Syntaxe :

fileXX = nom du fichier

Les guillemets ne sont pas nécessaires pour les chemins ou les noms de fichiers qui aient des espaces à blanc.

Exemple :

```
[action-reg]
filecount = 2
file00 = \flash\wifi.reg
file01 = \app.reg
```

ACTION-DLL

Non supporté

ACTION-EXE**Description :**

Exécute un fichier exécutable ou un document. Pour pouvoir exécuter un document, l'extension du fichier doit être associée à une application.

Après avoir réalisé l'exécution, le client continue avec les actions suivantes.

Syntaxe :

fileXX = nom du fichier

Si le chemin ou le nom du fichier contient des espaces à blanc, utilisez des guillemets.

Les arguments sont optionnels et n'ont pas besoin de guillemets.

Exemple :

```
[action-exe]
filecount = 1
file00 = \windows\app.exe
```

```
[action-exe]
filecount = 1
file00 = \windows\MyPage.htm
```

```
[action-exe]
filecount = 1
file00 = \windows\MyLink.lnk
```

```
[action-exe]
filecount = 1
file00 = "\Program Files\app.exe" -d MyArguments
```

ACTION-WAIT-EXE

Description :

Exécute un fichier exécutable ou un document. Pour pouvoir exécuter un document, l'extension du fichier doit être associée à une application.

Après avoir réalisé l'exécution, le client attend jusqu'à ce que l'application soit fermée avant de continuer avec les actions suivantes.

Syntaxe :

fileXX = nom du fichier

Si le chemin ou le nom du fichier contient des espaces à blanc, utilisez des guillemets.

Les arguments sont optionnels et n'ont pas besoin de guillemets.

Exemple :

```
[action-wait-exe]
filecount = 1
file00 = \windows\app.exe
```

ACTION-REGEXP

Description :

Exporte des clés de registre à partir d'un fichier.

Syntaxe :

fileXX = fichier [-a] reg-key

Si le chemin ou le fichier contient des espaces à blanc, utilisez des guillemets.

Abréviation des clés principales :

- HKLM : HKEY_LOCAL_MACHINE
- HKCU : HKEY_CURRENT_USER
- HKCR : HKEY_CLASSES_ROOT
- HKU : HKEY_USERS

Pour exporter la clé avec ses sous-clés, utilisez des crochets comme ci-dessous :
[HKLM\Comm]

Pour exporter uniquement les valeurs de la clé, utilisez de crochets comme ci-dessous :
[HKLM]\Comm

Il n'est pas possible d'exporter une seule valeur d'une clé.

Paramètres :

-a: ajouter le contenu de la clé du registre à la fin du fichier.

Exemple :

```
[action-regexp]
filecount = 4
file00 = \myapp.reg [HKCU\software\myapp]
file01 = \devices.reg [HKLM\comm\device1]
file02 = \devices.reg -a [HKLM\comm\device2]
file03 = "\Program Files\devices.reg" -a [HKLM\my key\device2]
```

ACTION-DEL

Description :

Efface un fichier. Cette action accepte des caractères joker ("?" et "*").

Syntaxe :

fileXX = nom du fichier

Les guillemets ne sont pas nécessaires pour les chemins ou les noms de fichiers qui aient des espaces à blanc.

Exemple :

```
[action-del]
filecount = 3
file00 = \myapp.reg
file01 = \flash\1.txt
file02 = \doc\*.txt
```

ACTION-REMOVE

Description :

Efface des répertoires et des fichiers. Cette action accepte des caractères joker ("?" et "*").

Syntaxe :

fileXX = répertoire

Exemple :

```
# remove a directory
[action-remove]
filecount = 1
file00 = \flash\test
```

```
# remove only the content of the directory
[action-remove]
filecount = 1
file00 = \flash\test\*.*
```

ACTION-KILL

Description :

Cette action réalise un kill d'un processus en exécution.

Syntaxe :

fileXX = nom du fichier exécutable

Le paramètre "file" peut être le nom d'un fichier exécutable ou le chemin complet vers un fichier exécutable. Si vous spécifiez un nom de fichier, le client cherchera le premier processus qui coïncide avec ce nom parmi les processus en exécution.

Spécifiez un nom de fichier si vous êtes sûr qu'il y a un seul processus en exécution avec ce nom.

Exemple :

```
[action-kill]
filecount = 1
file00 = \flash\myapp.exe
```

ACTION-DELAY

Description :

Cette action réalise une pause.

Syntaxe :

```
[action-delay]
delay = durée de la pause en millisecondes
```

Exemple :

```
# delay 3 seconds
[action-delay]
delay = 3000
```

ACTION-BOOT

Description :

Cette action effectue un warmboot ou un coldboot. Elle peut apparaître une seule fois dans un fichier et elle est toujours la dernière action exécutée.

Syntaxe :

```
[action-boot]
type = warmboot | coldboot
immediate = yes | no
delay = delay-time in seconds
```

Paramètres :

immediate : cette option est valide pour des fichiers .wdp uniquement (voir **WDP Files**). Si cette option est mise à "yes", ACTION-BOOT est exécuté immédiatement après la finalisation du traitement du fichier wdp.

delay : attendre X secondes avant effectuer un boot

Exemples :

```
# warmboot
[action-boot]
type = warmboot
immediate = no

# coldboot
[action-boot]
type = coldboot
immediate = yes
```

Annexe A : création des fichiers CPY

Les fichiers CPY ont une liste de fichiers à copier.

Syntaxe :

source > destination

source : chemin complet vers le fichier source.

destination : chemin complet vers le fichier de destination.

Les guillemets ne sont pas nécessaires pour le chemin du fichier source ni le chemin du fichier de destination.

Exemple des fichiers CPY :

```
\wdp\flexvnc.exe > \Windows\flexvnc.exe  
\wdp\flexftpserver.exe > \Windows\flexftpserver.exe  
\wdp\flexrpvserver.exe > \Windows\flexrpvserver.exe
```