1 Synopsis of the protocol 2
2 Data description 4
2.1 Identifiers 4
2.1.1 User ID 4
2.1.2 Terminal ID 4
2.2 Lists 5
2.2.1 Download List 5
2.2.2 Package List 7
2.2.3 Package 8
2.2.4 Collection 10
2.2.5 Collection List 12
2.3 Conventions in file naming 12
2.4 File types 12
2.5 Relations between entities 13
2.6 Launch List 14
3 Implementation 17
3.1 Deployer Agent/Client functions 17
3.2 Getting a Package List 18
3.3 Getting packages 19
3.4 Building a Package List 20
3.5 Terms in a Formula 22
3.5.1 Comparison operators 24
3.6 Silent Launcher at Startup 25
3.6.1 Using Silent launcher to start an application 25
3.6.2 Using Silent launcher to install applications 25
3.7 Wizard 26
3.8 Deployer Client commands 27
3.8.1 Update 27
3.8.2 List 27
3.8.3 Message 28

1 Synopsis of the protocol

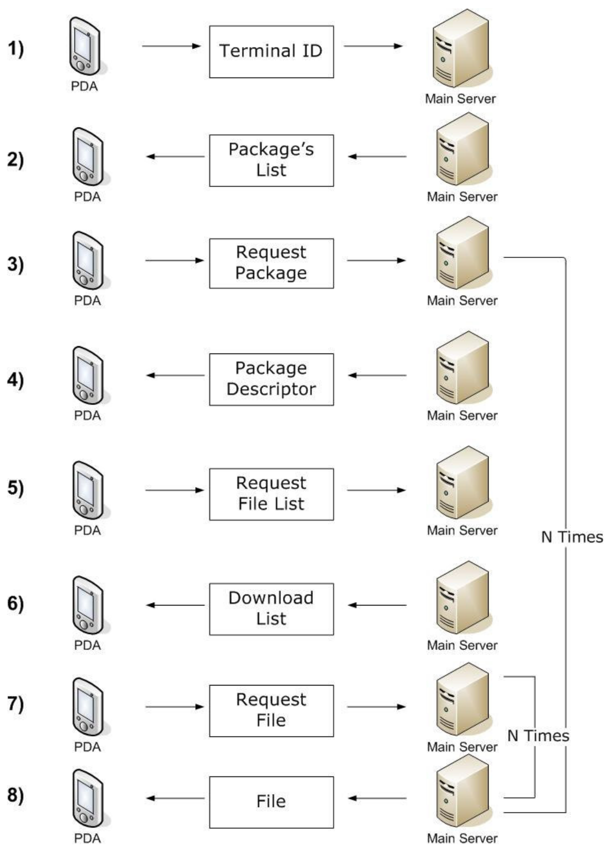
Each software package contains a list of files to be downloaded, and some "installation actions" (See Download list).

The Deployer Agent (Server) keeps a list of packages (See Package) that are grouped into Software Collections.

Once the Deployer Client terminal sends its Terminal ID, the Agent checks if there are any packages for that PDA by applying a formula (belongings to the Collections) (see Collection).

A package can belong to several Collections, A collection can group one or more packages.

Package installation implies the following steps:



1. Once the Deployer Client in the PDA obtains the Server IP (By wizard), it sends its Terminal ID *(see Terminal ID)* to the Deployer Agent (Server).

2. The Agent processes the Terminal ID and checks any collection formula to build a list with the available packages that match.

   For each package:

3. The PDA Deployer Client compares the list of installed packages with the downloaded list of available packages and requests the necessary *package descriptors to the agent*. The Client is in charge of deciding which packages will be downloaded from the Agent and installed in the PDA.

4. The Agent (Server) answers the requests by sending the packages descriptors.

5. The PDA Client processes the *package descriptor* and requests the download list to the Agent.

6. The Agent Server sends the Download List, which contains the files to be downloaded (and some other installation actions).

   For each file:

7. The PDA Client requests the file to the Server Agent.

8. The Server Agent sends the file to the PDA.

9. Process the actions in the download list

2 Data description

2.1 Identifiers

2.1.1 User ID

For a PDA exist a specific number of packages that can be used. With the *User ID,* you can configure a PDA so it works only with certain packages. A PDA with the same ModelName and Platform could be used for different purposes, and consequently it would need to have installed different packages. For example, a PDA with ModelName = MC9000 and Platform = WinCE400 could use “Wireless Telnet VT” and other with the same configuration could use “Wireless Mobile”.

2.1.2 Terminal ID

Identifies a unique terminal used by a user.

ModelName	Model Id or standard name of the terminal. If more than one are available they are separated by a “,”. Example: MC9000, PPT8800, PDT8100, CK30
Platform	Id of the platform in which the package is running. Example: WinCE300, WinCE410, WinCE420
UserID	Id of the user. Supported characters: “a...z”, “0...9”, “-“ and “_” Example: Vendedor01
Ip	Local address or IP range. In a range they are separated by a “-“. Example: 10.10.10.0 .10.10.1-90
Mac	MAC address of the terminal. Example: 00:A0:F8:5F:B9:8D

2.2 Lists

The Agent (Server) keeps a list of packages that are grouped into Software Collections. Once the terminal sends its Terminal ID, the Server checks if there are packages for that PDA by a formula that belongs to the collection (see Collection). A package can belong to several Collections.

2.2.1 Download List

The Download list contains a list of files to be downloaded from Agent to Client, and some install actions to process the installed files.

[DOWNLOAD]	
FileCount	Number of files in the *Download List.*
FileXX	XX: Numeric Id on list of files, starting by "00". The name of the file, in the following format: *Source File > Destination File* Source File and Destination File are separated by a ">". Source File: Name of the file in the Server. Destination File: Path of the file in the PDA. Example: WTnCe.exe>\Application\WtnCe.exe
[ACTION-ZIP]	It indicates if files need to be unzipped once copied.
FileCount	Number of files to unzip.
FileXX	Name of the file to unzip. XX: Numeric Id in the list of files, starting by "00".
[ACTION-CPY]	It indicates if files need to be copied in other directories.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by "00". Name of the file. *Source File > Destination File* Source File and Destination File are separated by a ">". Example: WTnCe.exe>\Application\WtnCe.exe
[ACTION-REG]	It indicates if .reg files need to be applied to registry once copied.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by "00". Name of the file. ".Reg" files is a standard format text file to create or modify keys in the registry.

[ACTION-DLL]	It indicates if there are libraries to be registered once the files are copied **(not yet implemented)**.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by "00". Name of the file.
[ACTION-EXE]	Executable files to launched
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by "00". Name of the file. Full path to an executable file, or file name of a file in executable path. The file will be name of an executable (e.g. ".exe") or a name of a document file associated with a WinCE action (e.g. ".cab").
[ACTION-DEL]	It indicates if files need to be deleted.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by "00". Name of the file. The file name will be a full qualified file name (e.g. "\Application\Myfile.reg") , or a file filter (e.g. "\Application*.tmp").

Example of Download List:

```
[DOWNLOAD]
FileCount = 7
File00 = cour.ttf>\Application\cour.ttf
File01 = critical.wav>\Application\critical.wav
File02 = launcher_9000.reg>\Application\launcher_9000.reg
File03 = WTnCE.lnk>\Windows\Desktop\WTnCE.lnk
File04= WTnCE.exe>\Application\WTnCe.exe
File05 = WTnCE.reg>\Application\WTnCe.reg
File06 = WTnCE.reg>\Application\Help.zip

[ACTION-ZIP]
FileCount = 1
File00 = \Application\Help.zip

[ACTION-CPY]
FileCount = 1
File00 = \Application\cour.ttf > \Windows\cour.ttf

[ACTION-REG]
FileCount = 1
File00 = \Application\launcher_9000.reg

[ACTION-DEL]
FileCount = 1
File00 = \Application\launcher_9000.reg
```

2.2.2 Package List

The package list is built by the Agent Server at request time based on Collection's formulas and the terminal ID sent by the Client. This list contains the available packages Id and versions for this client.

[LISTCOUNT]	It indicates which packages and versions the Download List contains.
Packages	Number of packages in the list.
PackageIdXX	XX: Numeric Id in the list, starting by "00". Package *Id*. Example: WTNVT9000
VersionXX	XX: Numeric Id in the list, starting by "00". Package *Version*. Example: 3-1-2

Example of a Package List:

```
[LISTCOUNT]
Packages = 4
PackageId00 = WTNVT9000
Version00 = 3-1-1
PackageId01 = WTNVT9000
Version01 = 3-1-2
PackageId02 = WTN529000
Version02 = 2-1-1
PackageId03 = WTN529000
Version03 = 3-2-1
```

2.2.3 Package

A software application that contains some files, stored in the Agent Server for a PDA Client.
Id and Version are primary keys in this entity.

[**PACKAGE]**	
Type	Type of package. A: Application (contains an exe file to launch) C: Configuration (contains configuration files)
Id	Identifier of the package. Supported characters: "a...z", "0...9", "_" and "."
Version	Major-Minor-Revision Example: 3-1-4 1-2-1
Name	Name of the package Example: WireLess Telnet VT WireLess Telnet 5250
FileList	Name of the file that contains the list of files to be downloaded. Supported characters: "a...z", "0...9", "-", "_" and "." Example: WtnCE52.cfg The content of this file is described in the entity ***Download List***.
[INFO]	
Description	Description of the content of the package. Example: Telnet Client for Symbol terminals
Platform(s)	Id of the platform where the package is running. Example: WinCE300, WinCE410, WinCE420.
Model(s)	Id of the model or standard name of the terminal. If more than one is available they are separated by a "," Example: MC9000, PPT8800, PDT8100, CK30
Vendor(s)	Id of the manufacturer Supported characters: "a...z", "0...9", "-", "_" and "." If more than one is available they are separated by a "," Example: Symbol, Intermec, Datalogic
[APPLICATION]	
StartExe	Name of the exe file to start the application. Supported characters: "a...z", "0...9", "-", "_" and "." Example WTnCE52.exe WTnCE.exe

Icon	Name of the file that contains the icon of the application. Supported characters: “a…z”, “0…9”, “-“, “_” and “.” Example: WTnCe52.ico

Example of a Package:

```
[PACKAGE]
Id =  WTNVT9000
Version = 3-1-2
Name = Wireless Telnet VT
Description = Telnet for Symbol PDA
Platform = WinCE410
Model = MC9000
Vendor =  Symbol
FileList = wtn9000.cfg

[APPLICATION]
StartExe =  WTnCe.exe
Icon = WTnCe.ico
```

2.2.4 Collection

A Collection groups one or more software packages with specific characteristics, and contains a formula. This formula is used to match with the terminal ID and build the package list

[COLLECTION]	
Id	Id of the collection. Supported characters: "a...z", "0...9", "-", "_" and "."
Name	Name of the collection. Example: Sellers
Description	Description of the collection. Example: Packages for sellers
[FORMULA]	It indicates the terms of the formula of the collection.
FormulaCount	Number of lines of the *FormulaXX*.
FormulaXX	A list of OR (|) between a number of AND (&) XX: Numeric Id of a term in the formula, starting by "00". Example: ModelName == MC9000 & Platform == WinCE300 ModelName == PDT8100 & Platform == WinCE300 The two lines can be joined in a list of OR (|) and the resulting formula is: (ModelName == MC9000 & Platform == WinCE300) | (ModelName == PDT8100 & Platform == WinCE300)
[PACKAGES]	It indicates which packages and versions contain the collection.
PackageCount	Number of packages of the collection.
PackageIdXX	Id of the package. XX: Numeric Id in the list of packages of the collection, starting by "00". Example: WTNVT9000 WTN529000
PackageVersionXX	Version of the package. XX: Numeric Id in the list of packages of the collection, starting by "00". Example: 3-1-2 1-2-1

Example of a Collection:

```
[COLLECTION]
Id = WTN9000
Name = Sellers
Description = Packages for sellers

[FORMULA]
FormulaCount = 3
Formula00 = (ModelName == MC9000 & Platform == WinCE300)
Formula01 = (ModelName  == PDT8100 & Platform == WinCE300)
Formula02 = (ModelName  == PDT8800 & Platform == WinCE420)

[PACKAGES]
PackageCount = 2
PackageId00 = WTNVT9000
PackageVersion00 = 3-2-2
PackageId01 = WTNVT9000
PackageVersion01 = 1-2-3
```

2.2.5 Collection List

The group of collections (main.clst).

[COLLECTION]	
CollectionCount	Number of collections.
CollectionIdXX	Id of the collection. XX: Numeric Id in the list, starting by “00”.

Example of a Collection List:

```
[COLLECTION]
CollectionCount = 3
CollectionId00 = C_WS_PDT8100
CollectionId01 = C_WS_MC9900
CollectionId02 = C_WS_PPT8800
```

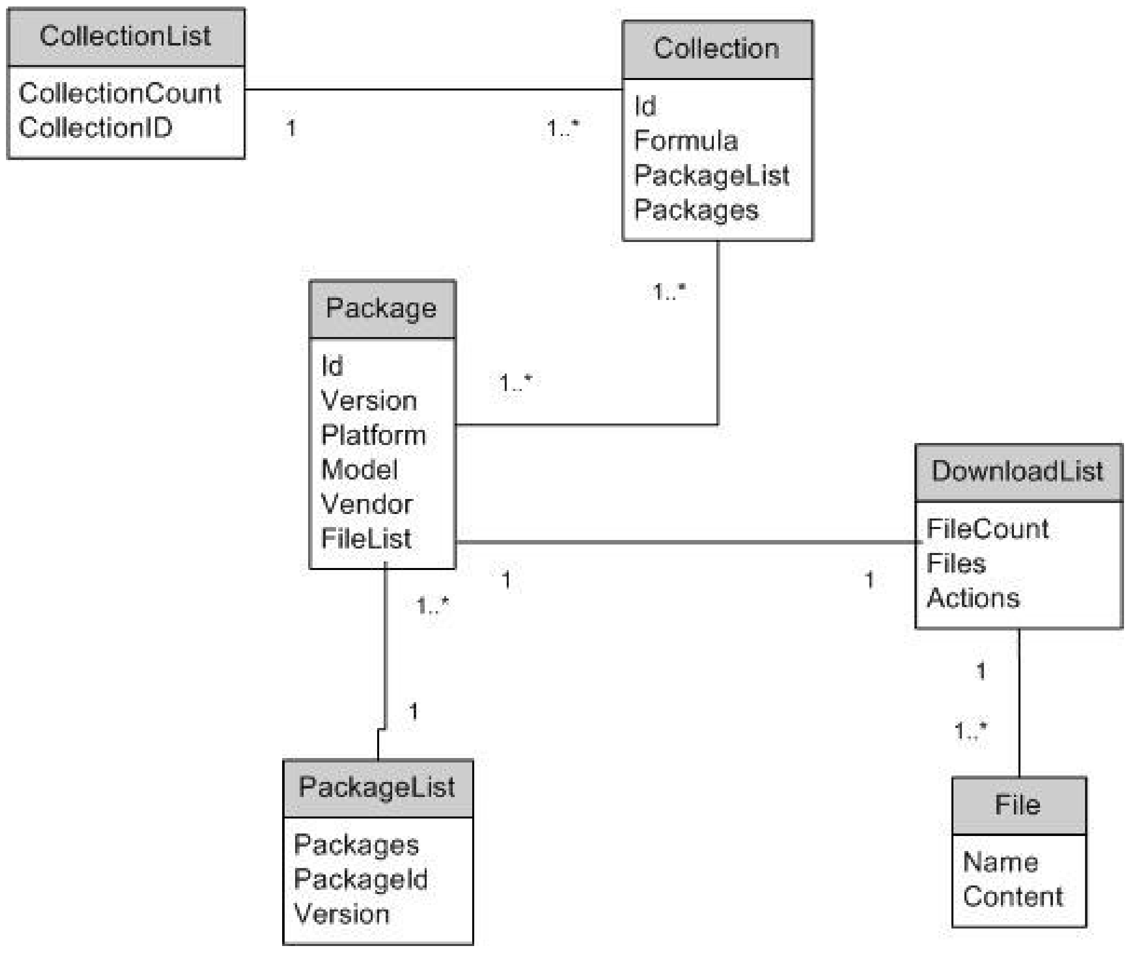
2.3 Conventions in file naming

Package List Download List	File with extension **cfg**
Package	File with extension **pckg**
Collection	File with extension **coll**
Collection List	File with extension **clst**
Launch List	File with extension **wdp**

2.4 File types

The entities that the Server Agent and Client handles are “INI” standard file types, with sections, keys and values, with one section at least.

2.5 Relations between entities



2.6 Launch List

Launch list are used to process previous downloaded files at start–up time.
The Launch list contains a list of install actions to process.
The launch list has an extension suffix of ".wdp".

[ACTION-ZIP]	It indicates if files need to be unzipped.
FileCount	Number of files to unzip.
FileXX	Name of the file to unzip. XX: Numeric Id in the list of files, starting by “00”.
[ACTION-CPY]	It indicates if files need to be copied in other directories.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by “00”. Name of the file. *Source File > Destination File* Source File and Destination File are separated by a “>”. Example: WTnCe.exe>\Application\WtnCe.exe
[ACTION-REG]	It indicates if “.reg” files needed to be applied to the Windows registry
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by “00”. Name of the file. “.Reg” files is a standard format text file to create or modify keys in the registry.
[ACTION-WAIT-EXE]	Executable files to be launched. The Deployer waits until the launched application finishes.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by “00”. Name of the file. Full path to an executable file, or file name of a file in executable path. The file will be name of an executable (e.g. “.exe”) or a name of a document file associated with a WinCE action (e.g. “.cab”).
[ACTION-EXE]	Executable files to be launched
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by “00”. Name of the file. Full path to an executable file, or file name of a file in executable path. The file will be name of an executable (e.g. “.exe”) or a name of a document file associated with a WinCE action (e.g. “.cab”).
[ACTION-DEL]	It indicates if files need to be deleted.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by “00”. Name of the file. The file name will be a full qualified file name (e.g. “\Application\Myfile.reg”) , or a file filter (e.g. “\Application*.tmp”).
[ACTION-REMOVE]	It indicates if files need to be deleted recursively.
FileCount	Number of files.
FileXX	XX: Numeric Id in the list of files, starting by “00”. Name of the file. The file name will be a full qualified file name (e.g. “\Application\Myfile.reg”) , or a file filter (e.g. “\Application*.tmp”).

Example of Launch List (HelpV100.wdp)**:**

```
[ACTION-ZIP]
FileCount = 1
File00 = \Application\Help.zip

[ACTION-CPY]
FileCount = 2
File00 = \Application\Help\HelpViewer.exe > \Windows\HelpViewer.exe
File01 = \Application\Help\HelpVw.lnk > \Windows\Desktop\HelpVw.lnk

[ACTION-REG]
FileCount = 1
File00 = \Application\Help\Config.reg

[ACTION-DEL]
FileCount = 3
File00 = \Application\ Help\Config.reg
File01 = \Application\ Help.zip
File02 = \Application\HelpV100.wdp
```

3 Implementation

3.1 Deployer Agent/Client functions

The PDA has a *Terminal ID* composed by the *ModelName*, *Platform*, User *ID* and *IP*.
ModelName and *Platform* cannot be configured.

The *User ID* will be configured during the installation of the Deployer Agent and cannot be changed later.

The Server IP can be configured manually or obtain it trough a Wizard (Agent) installed in the local network.

The Deployer Agent has three functions: *Download, Polling* and *Update*.

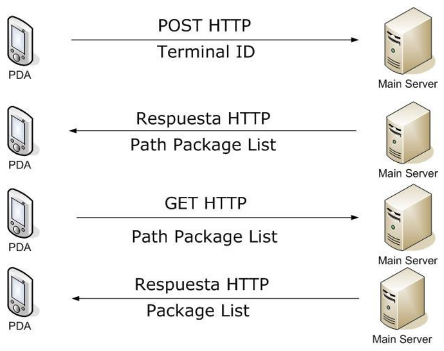
Download is performed manually by the PDA user.

Polling is performed by the PDA, which will ask the Server to see if there are packages to update. It can be configured manually by the user.

Update is performed by the Server, which will send a message to the PDA to update the packages (not implemented yet).

3.2 Getting a Package List

The Deployer Client/Agent uses the HTTP Protocol to obtain a Package List.
The *Terminal ID* is sent from the PDA to the Server through an HTTP POST (**reqpkglst.php**).
The Server answers sending the path of the file that contains the *Package List* for the Terminal ID, which can be obtained later by the PDA through an HTTP GET.



Example:

Terminal ID of the PDA	modelName=MC9000
	platform=WinCE420
	userid=Vendedor01
	ip=10.10.10.0
	mac=00_A0_F8_5F_89_8D

The PDA will send at least the following HTTP Headers with the data of the *Terminal ID*:

```
POST /reqpkglst.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: SofToGo Deployer
Host: 200.169.60.122:8080
Content-Length: 66
Cache-Control: no-cache

modelName=MC9000&platform=WinCE420&userid=Vendedor01&ip=10.10.10.5&mac=00:A0:F8:5F:89:D
```

Then, in response to the POST, the Server will send the path of the Package List and through an HTTP GET it can be obtained. If the path of the *Package List* in the Server were for example: /usr/SoftoGo/Deployer/PackageList/list001.cfg

```
GET /usr/SoftoGo/Deployer/PackageList/list001.cfg HTTP/1.1
User-Agent: SofToGo Deployer
Host: 200.169.60.122:8080
```

3.3 Getting packages

The request of the *Packages Descriptor*, the *Download List* and the files of a package are done through HTTP requests to the Server and will be sent through the GET method.
When the installation of the packages is finished, the terminal will send to the Server through a POST the Terminal ID and the packages installed or updated.

The POST is done on the file **UdpMobileStatus.php**

Example: with an installed package

```
POST /UdpMobileStatus.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: SofToGo Deployer
Host: 200.169.60.122:8080
Content-Length: 66
Cache-Control: no-cache

modelName=MC9000&platform=WinCE420&userid=Vendedor01&ip=10.10.10.5&mac=00:A0:F8:5F:89:8
D&pkg00=WTN52_88xx&Version00=3/1/4
```

3.4 Building a Package List

The Agent Server has a *Collection* that has a number of Collections (See 3.6 Relations between entities). Each time a *Terminal ID is sent from Client to Agent*, the Server Agent follows these steps for the Collec*tion*:

- It gets the *Collection List*
- It gets each *Collection* that belongs to that *Collection List.*
- Verifies if the *Terminal ID* matches with any term of the Formula of the *Collection,* if it matches it gets from the section [PACKAGES] of the Collection the PackageId and PackageVersion (one or more) and they are added to the *Package List.* This step is repeated for every *Collection* that the *Collection List* has*.*

Example:

From the *Terminal ID the Agent* gets ModelName, Platform and UserID. For example, with the following *Terminal ID:*

```
ModelName=MC9000
Platform=WinCE410
UserId=Vendedor01
```

And 3 different Collections:

<u>Collection 1</u>

```
[FORMULA]
FormulaCount = 2
Formula00 = (ModelName == PDT8100 & Platform == WinCE300)
Formula01 = (ModelName == MC9000  & Platform == WinCE300)

[PACKAGES]
PackageCount = 2
PackageId00 = WTNVT8100
PackageVersion00 = 3/2/2
PackageId01 = WTN528100
PackageVersion01 = 1/2/3
```

<u>Collection 2</u>

```
[FORMULA]
FormulaCount = 1
Formula00 = (ModelName == MC9000 & Platform == WinCE410 &  UserId == Vendedor01)

[PACKAGES]
PackageCount = 3
PackageId00 = WTN529000
PackageVersion00 =  3/2/2
PackageId01 = WTNVT9000
PackageVersion01 =  3/2/2
PackageId02 = WTNVT9000
PackageVersion02 =  3/2/1
```

<u>Collection 3</u>

```
[FORMULA]
FormulaCount = 4
Formula00 = (ModelName == PPT8800 & Platform == WinCE420)
Formula01 = (ModelName == MC9000  & Platform == WinCE420 & UserId == Vendedor01)
Formula02 = (ModelName == MC9000 & Platform == WinCE300)
Formula03 = (ModelName == PDT81000 & Platform == WinCE300 & UserId == Vendedor01)

[PACKAGES]
PackageCount = 1
PackageId00 = WKBD
PackageVersion00 =  1/1/1
```

For <u>*Collection 1*</u> no packages will be added to the Package List because although *ModelName* matches, *Platform* doesn´t.
In the case of <u>*Collection* 2</u> and <u>*Collection 3*</u>, the packages will be added to the *Package List* because *ModelName* and *Platform* match one of the terms in the formula.

The ***Package List*** would be as follows:

```
[LISTCOUNT]
Packages = 4
PackageId00 = WTN529000
PackageVersion00 = 3/2/2
PackageId01 = WTNVT9000
PackageVersion01 = 3/2/2
PackageId02 = WTNVT9000
PackageVersion02 = 3/2/1
PackageId03 = WKBD
PackageVersion00 = 1/1/1
```

3.5 Terms in a Formula

A formula is composed by a number of OR between a number of AND. A term is complete if it has the attributes ModelName, Platform and UserId.
For example:

a) (ModelName == MC9000 & Platform == WinCE410 & UserId == Vendedor01)

It also can appear incomplete terms.
For example:

b) (ModelName == MC9000 & Platform == WinCE410)

c) (ModelName == MC9000)

Case a) is a complete term, therefore if the *Terminal ID* doesn´t have the same attributes, it won´t be able to install the packages that belong to that Collec*tion*.
Case b) is an incomplete term, and is independent from the UserId, therefore if the *Terminal ID* matches with the attributes, it would be able to install the packages of the *Collection*.
Case c) is also an incomplete term, independent from the *UserId* and *Platform*, therefore if the ModelName of the *Terminal ID* is *mc9000,* it would be able to install the packages of the *Collection*.

The packages of the Collections that have complete terms will be for specific purposes.
With two different UserId (with the same Platform and ModelName), V1 and V2, they can have different packages, but at the same time they can also share some. For example:

UserId	Packages
V1	A, B & C
V2	A, C & D

At least there will be two Collections.
The first one will have one term as follows:

```
Formula00 = (ModelName == MC9000 & Platform == WinCE410 & UserId==V1)
[PACKAGES]
PackageCount=3
Package00=A
Package01=B
Package02=C
```

The second one will have a term as follows:

```
Formula00 = (ModelName == MC9000 & Platform == WinCE410 &  UserId==V2)
[PACKAGES]
PackageCount=3
Package00=A
Package01=C
Package02=D
```

Because A and C are shared by V1 and V2, they can generate a new *Collection*. Then, quedarían tres *Collections*. (?)

<u>Collection 1</u>:

```
Formula00 = (ModelName == MC9000 & Platform == WinCE410)
[PACKAGES]
PackageCount=2
Package00=A
Package01=C
```

<u>Collection 2</u>:

```
Formula00 = (ModelName == MC9000 & Platform == WinCE410 & UserId== V1)
[PACKAGES]
PackageCount=1
Package00=B
```

<u>Collection 3</u>:

```
Forumula00 = (ModelName == MC9000 & Platform == WinCE410 & UserId== V2)
[PACKAGES]
PackageCount=1
Package00=D
```

3.5.1 Comparison operators

The supported operators for the comparisons in a formula are the following:

==	Equal
<=	Less or equal
>=	Greater or equal
>>	Greater
<<	Less
<> !=	Different

The comparisons are made character by character and they are not case-sensitive.

The following comparisons are the same:

Platform == WinCE410
Platform == WINCE410
Platform == wince410

Examples of comparisons:

a) Platform <= WinCE420	True if Platform is WinCe420, WinCe410 or WinCe310 for example
b) UserId >> 112	False if UserId begins with a letter. For example: a200 or A200.

c) ModelName == PPT8800 & Platform <> WinCE410 & UserId <= 100

3.6 Silent Launcher at Startup

The WireLess Deployer Client will be used as silent installer / Launcher at boot time.

At start time, The WireLess Deployer Agent looks for "*.wdp" (Launch List) files in the flash card directory (e.g. \Application).

If **any** file is found, the file is processed and the Deployer Client **exits**.

One or more files will be found and processed.

To ensure the processing order, the search of suffixes are done in the following order : "*.wdp", "*.wdp0", "*.wdp1", … "*.wdp5".
The launch list files ("*.wdp") will deletes themselves.

3.6.1 Using Silent launcher to start an application

Build a "launch list" with the name of the application ".exe" file in the ACTION-EXE section.
Download in the flash card the WireLess Dseployer Client executable file.
Download in the flash card the user application executable file.
Download in the flash card the Launch List.
Download in the Windows Startup folder a shortcut to the Deployer Client.
Warm boot the terminal.

The WDP Client is launched by WinCE, founds the ".wdp" file, processes it and launches the user application.

3.6.2 Using Silent launcher to install applications

In addition of the start application steps, some other launch list will be added.

Build a "Launch List" with the actions to perform, and request the deletion of the file in ACTION-DEL.
Download in the flash card the Launch List.
Download in the flash card the appropriate files.
Warm boot the terminal.

The WDP Client is launched by WinCE, founds some ".wdp" files, processes it and launches the user application.

3.7 Wizard

The Deployer Client / Agent Wizard function is used to locate the servers (Agent) by broadcast, ant connect to it by selecting in a list if necessary.
The IP of the Agent (Server) can be configured to obtain it through a Wizard installed in the local network. The terminal will send an UDP broadcast with a *Wizard ID* and will wait the answer of any wizard.

Port	Usually the port is UDP/8128, however it can be configured to receive in other port.
Wizard Id	Identifier that the Deployer Agent sends. If the *Wizard ID* of the Wizard doesn't match with the one the terminal sent, the Wizard won't send a response.

The terminal will receive a list of servers with the following fields:

Ip	IP address of the server
Port	Port of the server

The fields are separated by "\n" and if there is more than one server configured is indicated with "\n\n".

Example with two servers:

ip=10.10.10.5\nport=8128\n\nip=200.61.159.7\nport=8128

3.8 Deployer Client commands

The Deployer Client has configured a port (8129) to receive commands, which are sent by any Server and received by the Client. Te fields have to be sorted as follow.

Fields:

CMD	Command: UPDATE LIST MSG
WIZARD	(Y | N) Indicates if the terminal must use a Wizard
IP	IP of the Server. In case of using a Wizard this field is not sent.
PORT	Port of the Server. In case of using a Wizard this field is not sent.
PAGE	Name of the page where data will arrive.
TEXT	Message to display in the screen of the terminal

3.8.1 Update

Updates one or more packages. It can also install new packages.
Is done by sending data through POST method to the Server.

The Server must send the following fields (Example without wizard):

```
CMD = update
WIZARD = N
IP = 10.10.10.5
PORT = 80
PAGE =  ReqPkgList.php
```

3.8.2 List

Sends the list of packages actually installed, in a format that is similar to the *Package List*.

The Server must send the following fields:

```
CMD = list
PAGE =  UpdMobileStatus.php
```

3.8.3 Message

Displays a message in the screen of the terminal.

The Server must send the following fields:

```
CMD = msg
TEXT = Message…
```