Channel Designer

WMChannelDesigner.doc

20051018 Version 1.0.1	Pablo Cepeda
20051017 Version 1.0.0	Karina Spak

INDEX

CHANNEL INTRODUCTION

A channel is a web application that runs on the wireless mobile client. It is composed of various items or files.

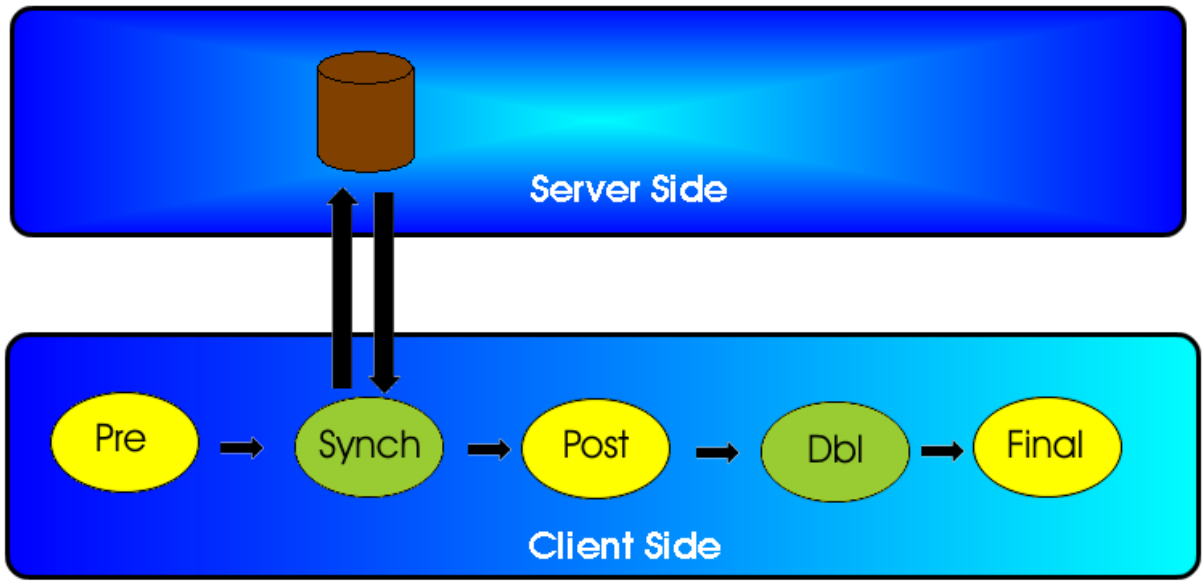
CHANNEL COMPONENTS

A channel is composed of different items.

- **Homepage**: Every channel has a default homepage with named `<ChannelName>hp.html`.
- **Templates**: Files with a special structure with extension `tpl`. The WireLess Mobile client transforms these files replacing structures (variables) with data, generating html files. [3]
- **Queries**: Queries made to the client database. The extension of the file is .**`qry`**. [3]
- **DBL file**: Updates and modifies the client (PDA) database. The dbl file is unique and has the format `<ChannelName>.dbl`.
- **Form Submit**: Forms that permit submitting data without having to wait for the synchronization.
- **Pre, Post and Final DataSubmits**: They allow to send data to some backend through a specific url and to modify the contents of the database before and after the synchronization. They are named : `<ChannelName>_pre.mbq`, `<ChannelName>_post.mbq` and `<ChannelName>_last.mbq`.

CHANNEL EXECUTION

When a channel is downloaded or updated during the synchronization process, several steps are taken.



Presynchronization: (Optional). In this step some queries are executed creating result files that can be send to the server. To do this, a file named *ChannelName*_pre.mbq must be coded. [1]

Synchronization: Data is sent to the server (if there is any), new data and/or applications are downloaded from the server (if there are any). The data downloaded is a .zip file that contains all the content (queries, templates, etc) of the channel. The file is unzipped during this process.

PostSyncronization: (Optional). This step is done after synchronization and only if synchronization was successful. It is executed to make updates on the database. It can be used for example to delete data that is no longer necessary in the client PDA because it has already been sent to the server. [1]

Execution of dbl file: This file is used to change the database structure and/or to fill and/or delete rows of tables in the PDA client. The file is transferred from server to client during synchronization. The name of the file is *ChannelName.DBL*. [2]

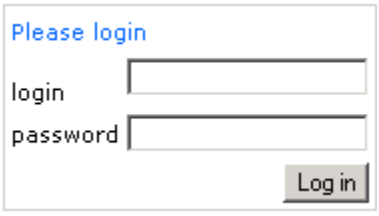
Finalization: (optional). The structure is the same as that of postsync, with the difference that final is executed after the database is updated in the WireLess Mobile client.

CHANNEL DESIGN AND CONSTRUCTION

To create new channels take the following steps

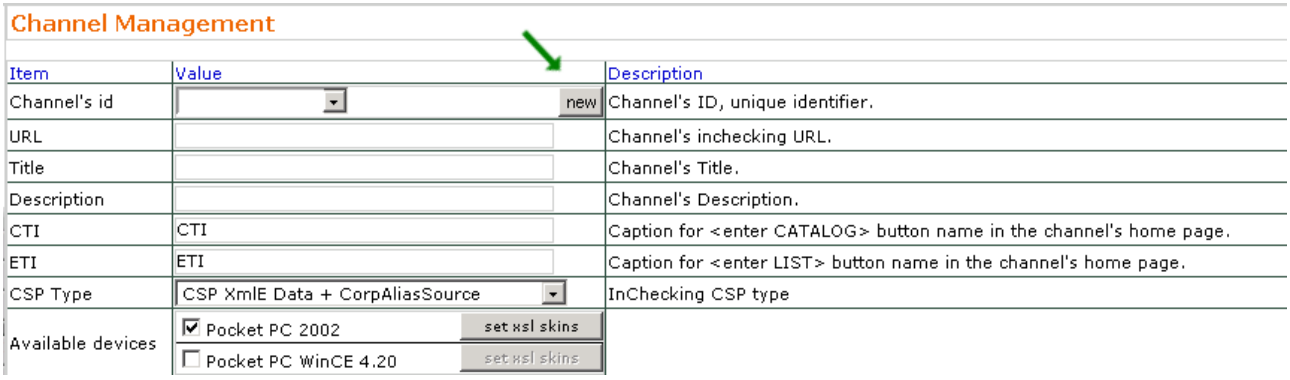
1) Open your browser, and write **http://localhost/wmbsync/admin/** in the address bar**.**

2) Login with your user and password



3) Choose Channel management from the menu.

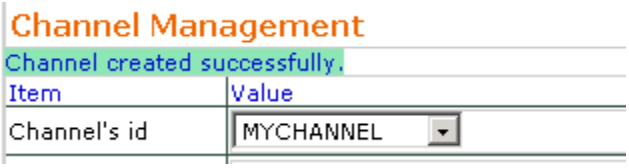
4) Click on New.



5) Enter the channel Name and click on Accept. DO NOT use special characters (accent, space, dot, underscore, apostrophe).

For this example write `MYCHANNEL`

A message like this should appear



6) Complete the form. Here is a description of the fields.

URL: Channel's inchecking URL for data submission. This is an advance item and we won't be using it for now. It can be used for example to download different data for different users.
In this example leave it blank.

Title: Channel's title. It appears on the PDA homepage.
Example: EXAMPLE CHANNEL.

Description: Channel's Description.
Example: This is my first channel.

CTI: Catalogue title. Is the catalog button name, enter the button text that will appear on the PDA home page as access to the catalog. (Only here for backward compatibility.)

ETI: List title. (Only here for backward compatibility.)

CPS Type: is the format of the input data.

- CSP TXT Data 6 Col + CorpAliasSource (Deprecated)
- CSP TXT Data 14 Col + CorpAliasSource (Deprecated)
- CSP Full XmlE + CorpAliasSource (Deprecated)
- CSP Xml/Xsl + CorpAliasSource (Deprecated)
- **CSP XmlE Data + CorpAliasSource** Always select this option. The others are there only for backward compatibility.

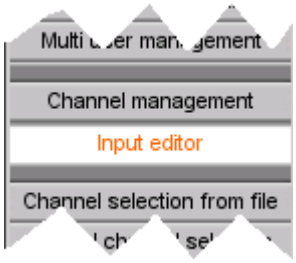
Available Devices: You can select the clients. For each client, a folder will be created when generating the content of the channel.

- **Pocket PC 2002:** Windows CE 2002 and 2003.
- **Pocket PC WinCE 4.2**

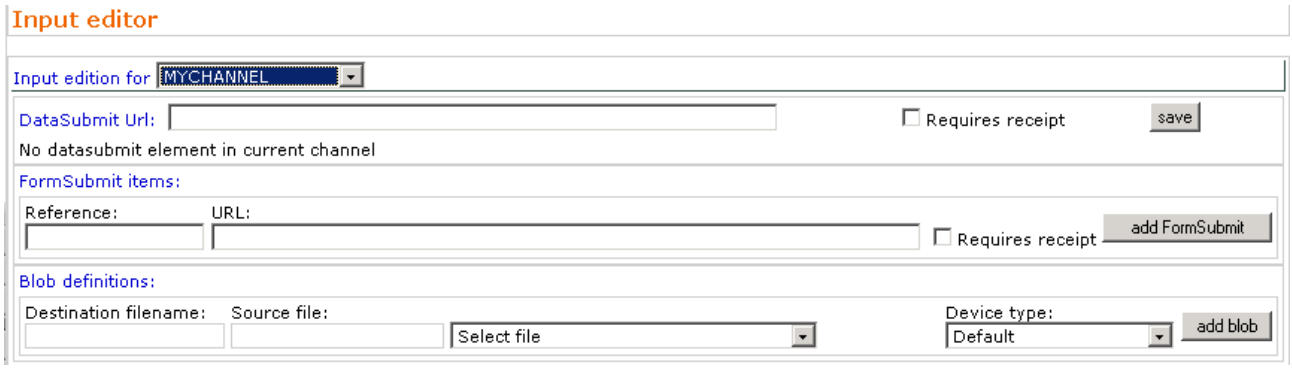
In this example we select Pocket PC 2002 because we are generating this channel for a Symbol 8100 with PocketPC 2002.

Set xls skins button: This is an advanced item and we are not going to use it in this example.

7) Now click on the Input Editor from the menu.



8) The next screen should appear.



Next there is a description of the fields.

DataSubmit Url: Url for data submission. In this example write http://localhost/getorders.asp. [1]
This field becomes necessary when a _pre synchronization file exists, all the data is sent to this page.
After filling the field click on the **save** button.

Form Submit Items: Pages that generates form data to be sent via WMServer at synchronization time. FormSubmit items can be generated with the WireLess Mobile Server. They require a destination url for outgoing data and there's possible to select a custom XSL for each form.

Blob Definitions: Management of attached files also called BLOBs (for Binary Large Objects).
They are all the files that belong to a channel. This includes:

- Templates: .tpl files
- Queries: .qry files
- Form submit: html included in .aar files
- Home page: .html (there is only one)
- DBL file: .dbl file (there is only one)
- _pre, _post and final: . mbq files

Blobs have a source file and a destination file name that may have different names.
Destination file names must be unique. If not, an error message will appear.
After selecting the source file, the destination file name and the device type click on the **add blob** button.

To follow the example described here you should download the example file (**`WmMyChannel.zip`**) from www.softogo.com from the wireless mobile section, and then unzip it into the ctgData folder of `MYCHANNEL`.

In `MYCHANNEL` example:

- select *home.html* from the listbox. In Destination Filename change with `MYCHANNELhp.html`, then add it.
- select ChannelName.dbl from the listbox. In Destination Filename change with `MYCHANNEL.dbl`, then add it.
- select `CHANNELNAME_pre.mbq`, `CHANNELNAME_post.mbq` and `CHANNELNAME_final.mbq` and change them to `MYCHANNEL_pre.mbq`, `MYCHANNEL_post.mbq` and `MYCHANNEL_final.mbq` respectively, then add them.

Changing destination filenames makes easier to reuse the same file in different channels.

- Select all the other files in the clgdata directory (except `CorpAliasSource.xml`) and add them to the channel.

9) Your channel is ready. You can always go back to the Channel Management screen or the Input Editor to change anything.

EXAMPLES

The channel example we just made has various simple examples to see how a channel works.
We will look at them thoroughly. You can execute each of them by clicking on the respective link.

First of all we will look at the .dbl file to see what format the database has. For these simple examples we will use the same .dbl all the time, but as we mentioned before the .dbl is generated every time we do the synch process so the .dbl can be different each time.

Database

If you edit the .dbl file you will see this code:

```
<DBLFILE>
<VERSION NUM="1"></VERSION>
<DATABASE>

<SQL>DROP TABLE CLIENTS</SQL>
<SQL>CREATE TABLE CLIENTS ( CodCli integer PRIMARY KEY, Client
nvarchar(20), CliMail nvarchar(20))</SQL>
<DATA-ADD TABLE="CLIENTES" FIELD01="CodCli" FIELD02="Client"
FIELD03="CliMail">
0001|FRODO|frodo@mail.com
0002|GANDALF|gandalf@mail.com
0003|LEGOLAS|legolas@mail.com
0004|ARAGORN|aragorn@mail.com
0005|EOWIN|eowin@mail.com
</DATA-ADD>
<SQL>CREATE INDEX iCodCli ON CLIENTES(CodCli)</SQL>

<SQL>DROP TABLE NEWCLIENTS</SQL>
<SQL>CREATE TABLE NEWCLIENTS ( CodCli integer PRIMARY KEY, CliName
nvarchar(20), CliMail nvarchar(20), Sended integer)</SQL>

</DATABASE>
</DBLFILE>
```

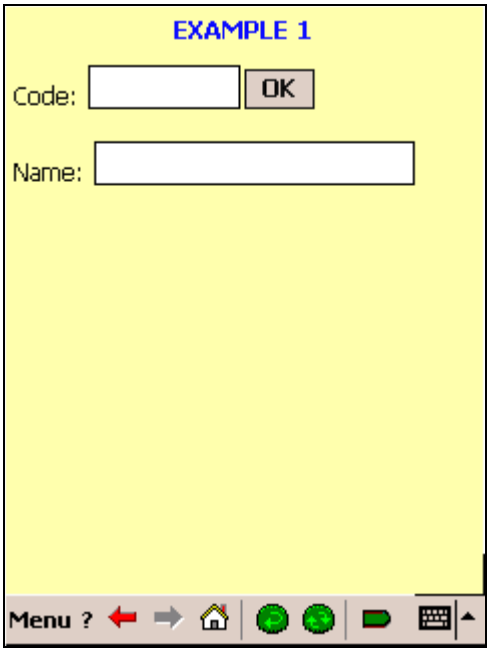
This means that the next tables are generated

CLIENTS		
CodCli	Client	CLiMail
0001	FRODO	frodo@mail.com
0002	GANDALF	gandalf@mail.com
0003	LEGOLAS	legolas@mail.com
0004	ARAGORN	aragorn@mail.com
0005	EOWIN	eowin@mail.com

NEWCLIENTS		
CodCli	CliName	CliMail

EXAMPLE 1

In this example you have to write a client code (in this case a number from 1 to 5), then click ok. This will perform a query to the database (on the PDA) and display the client name.



Files involved:

Home.html
Example1.tpl
Example1.qry

How it works:

If you edit the homepage, the part of the code that involves example1 is

```
<a href="call://DisplayList?tpl=Example1.tpl">
example 1
</a>: Input a Client Code with the keyboard and bring the data of that
client
```

This means that when you click on example1, the template example1.tpl will be displayed.

We will examine example1.tpl. We will only look at the part of the code that is relevant.

```
<form method="post" name="client"
action="call://DisplayList?qry=example1.qry&tpl=example1.tpl">
      Code: <input size="8" name="CODCLI" value="$CODCLI$">
      <input type="submit" value=" OK " name="ok"><br><br>
      Name: <input disabled type="text" value="$NOMCLI$" name="ok">
</form>
```

This form has two fields, Code and name of the client. Each time the form is submitted a query is performed (example1.qry), and the results are displays through the template example1.tpl (the same template each time).

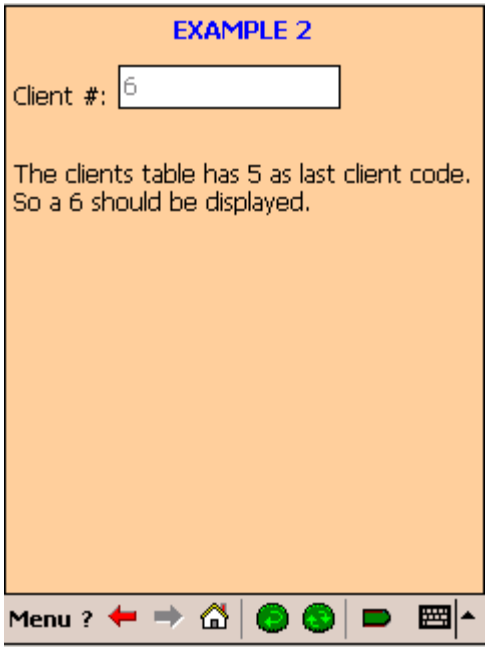
Example1.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT Client
FROM CLIENTS
WHERE CodCli = '$CODCLI$'
.#.EVAL
(SET NOMCLI $1$)
```

This file performs an sql query that returns the client name of the client with code CodCli (which value is the variable CODECLI that was in the .tpl). It then sets the result in a variable called NOMCLI.

EXAMPLE 2

This example performs a query to the database and displays the next valid client code.



Files involved:

Home.html
Example2.tpl
Example2.qry

How it works:

If you edit the homepage, the part of the code that involves example2 is

```
<a ref="call://DisplayList?qry=Example2.qry&tpl=Example2.tpl">
  example 2
</a>: Generate next client code automatically
```

When you click the example 2 link the query named Example2.qry will be performed and the results displayed through the template Example2.tpl.

Let´s first look at the query:

```
Example2.qry

.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT MAX(CodCli)
FROM CLIENTS
.#.EVAL
(SET LASTCODE $1$)
(INCREMENT LASTCODE)
```

The query selects the Client code with the biggest value, then set a variable with that value named LASTCODE. Finally it increments that variable by one.

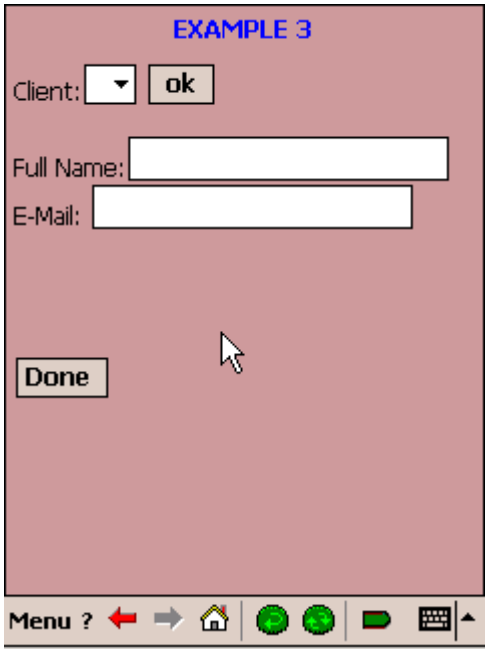
Now let´s look at the important part of the .tpl

```
<form method="post" name="client" action="">
  Client #: <input disabled size="12" name="LASTCODE" value="$LASTCODE$">
</form>
```

In this form we receive the variable LASTCODE we queried from the hompage and display it in an input field.

EXAMPLE 3

This example performs a query and displays a list box with all the client codes that are in the database. You have to choose one and then click ok. This will perform another query and the name and email of the client will be displayed.



Files involved:

Home.html
Example3.tpl
Example3.qry
Example3a.qry

How it works:

If you edit the homepage, the part of the code that involves example3 is

```
<a href="call://DisplayList?qry=Example3a.qry&tpl=Example3.tpl">
      example 3
</a>: Select a client code in a dropdown and Display the data in the
      same page.
```

Again, when we click on the link, we have a query performed and the results displayed through a template, in this case Example3a.qry and Example3.tpl respectively.

Let´s look at the query.

Example3a.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT CodCli
FROM CLIENTS
WHERE NOT CodCli = '$PATTERN$'
ORDER BY CodCli
```

This query brings back some client codes in order. At this time the value of PATTERN is an empty string.

Now the .tpl

Example3.tpl

```
<!--line:pre-->
<form method="post" name="clients"
action="call://DisplayList?qry=example3.qry&tpl=example3.tpl">
      <input type="hidden" name="CODCLI" value="$CODCLI$">
      Client:<select name="PATTERN">
      <option value="$PATTERN$" selected>$PATTERN$</option>
            <!--line:do-->
            <option value="$1$">$1$</option>
            <!--line:post-->
      </select>
      <input type="submit" value=" ok " name="ok"><br><br>
      Full Name:<input disabled value="$NOMCLI$"/><br>
      E-Mail: <input disabled value="$MAIL$"/><br><br>
</form>
```

There are some new statements here that are part of the tpl structure.

```
<!--line:pre-->
<!--line:do-->
<!--line:post-->
```

The lines of code following `<!--line:pre-->` are executed before the query is called.
For each result row of the query, the `<!--line:do-->` executes some code
The lines of code following `<!--line:post-->` execute all the rest of the code

In this template we perform the next query passing to the query the variable with value $PATTERN$.

Example3.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT CodCli,Client, CliMail
FROM CLIENTS
WHERE CodCli = '$PATTERN$'
.#.EVAL
(SET CODCLI $1$)
(SET NOMCLI $2$)
(SET MAIL $3$)
.#.SQL
SELECT CodCli
FROM CLIENTS
WHERE NOT CodCli = '$PATTERN$'
ORDER BY CodCli
```

First we select some data from the table clients where the client code is the same as the value of PATTERN.
Then we set some variables for the result.
Finally we select de client codes which are different from PATTERN. This is used not to repeat values in the drop down box.

When the results are brought back to the template the `<!--line:do-->` code is executed.
In this case `<option value="$1$">$1$</option>` fills the drop down box with the client codes.
After all the results are processed, all code after `<!--line:post-->` is executed, in this case some data fields which values were obtained in the query are displayed.

In the same .tpl we have some other important part of code:

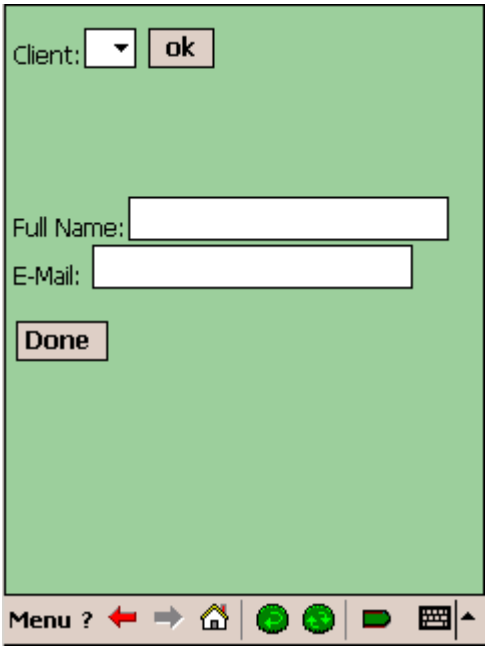
Example3.tpl(Cont)

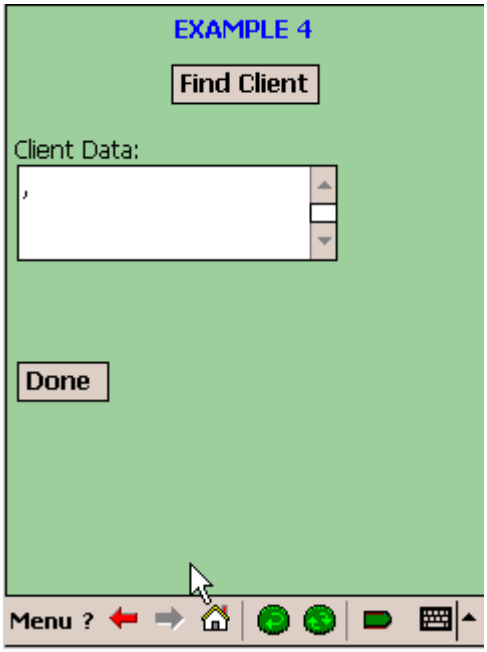
```
<form method="get" action="$cid$hp.html">
    <input type="submit" value="Done " name="return">
</form>
```

The only thing this code does, is displaying a button that when clicked, it takes you to the homepage. There is a special variable defined, the channel id (cid) than can be used to call the homepage (as the example shows) of the channel (provided that the homepage has the name <ChannelName>hp.html)

EXAMPLE 4

This example shows how to bring data from another page. Click on the **Find Client** button. This will take you to the example 3 page, so you have to do the same as in that example. When you click on the **Done** button you will return to the other page but with the data displayed there.





Files involved:

Home.html
Example4.tpl
Example4.qry
Example4a.tpl
Example4a.qry

How it works:

If you edit the homepage, the part of the code that involves example4 is

```
<a href="call://DisplayList?tpl=Example4.tpl">
      example 4
</a>:Select a client code and then display the data in another page.
```

When clicking on example 4, the template example4.tpl will be displayed.

Let´s take a look at this template:

Example4.tpl

```
<form action="call://DisplayList?qry=Example4a.qry&tpl=Example4a.tpl"
method="post">
      <center><input type="Submit" value="Find Client"></center><br>
      Client Data:<br>
     <textarea rows="3" cols="20">$CliName$,$CliMail$</textarea>
</form>
```

This page displays a button and has a text area. When clicking the button, that in this case submits the form, a query is performed (Example4a.qry) and the results displayed through the template Example4a.tpl.

Here we have the query just mentioned:

Example4a.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT CodCli
FROM CLIENTS
WHERE NOT CodCli = '$PATTERN$'
ORDER BY CodCli
```

This query is identical to Example3.qry, and we already explained it.

And if we look at the template Example4a.tpl is identical to that of Example3.tpl, just replacing the .qry and .tpl files by example4.qry and example4a.tpl. So it doesn´t need an explanation either.

Example4a.tpl

```
<!--line:pre-->
<form method="post" name="clients"
action="call://DisplayList?qry=example4.qry&tpl=example4a.tpl">
      <input type="hidden" name="CODCLI" value="$CODCLI$">
      Client:<select name="PATTERN">
      <option value="$PATTERN$" selected>$PATTERN$</option>
            <!--line:do-->
            <option value="$1$">$1$</option>
            <!--line:post-->
      </select>
      <input type="submit" value=" ok " name="ok"><br><br>
      Full Name:<input disabled value="$NOMCLI$"/><br>
      E-Mail: <input disabled value="$MAIL$"/><br><br>
</form>
```

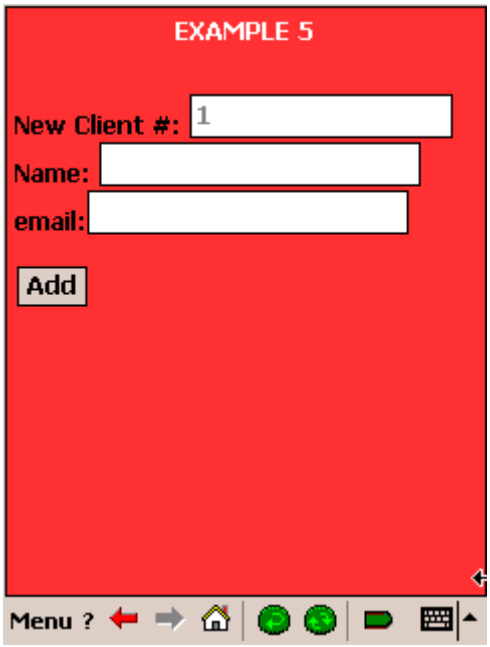
Example4.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT CodCli,Client, CliMail
FROM CLIENTS
WHERE CodCli = '$PATTERN$'
.#.EVAL
(SET CODCLI $1$)
(SET NOMCLI $2$)
(SET MAIL $3$)
.#.SQL
SELECT CodCli
FROM CLIENTS
WHERE NOT CodCli = '$PATTERN$'
ORDER BY CodCli
```

The query performed in the last .tpl is also identical to that of example3.

EXAMPLE 5

This is an example of how the pre, post and final files for synchronization work. It will also demonstrate how an incheck url can be used.
What you have to do is add some clients by writing the name and email, then click **Add.** After you added the clients you wanted, synchronize. You will see at the end of the synchronization a message of what sync steps were taken.



Files involved:

Home.html
Example5.tpl
Example5.qry
Example5add.qry
ChannelName_pre.mbq
ChannelName_post.mbq
ChannelName_final.mbq
getOrders.asp

How it works:

If you edit the homepage, the part of the code that involves example4 is

```
<a href="call://DisplayList?qry=Example5.qry&tpl=Example5.tpl">
      example 5
</a>:Show how _pre.mbq, _post.mbq and _final.mbq works
```

When clicking on example 5, the query Example5.qry will be performed and the results displayed through the template example5.tpl.
As you see Example5.qry is identical to Example2.qry, so no explanation is needed.

Example5.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT MAX(CodCli)
FROM NEWCLIENTS
.#.EVAL
(SET LASTCODE $1$)
(INCREMENT LASTCODE)
```

Example5.tpl

```
<form method="post"
action="call://DisplayList?qry=Example5Add.qry&tpl=example5.tpl">
New Client #: <input type="text" disabled size="12" name="lc"
value="$LASTCODE$"><br>
<input type="hidden" name="LASTCODE" value="$LASTCODE$">
Name: <input name="CliName" value="$NOMCLI$"/><br>
email:<input name="CliMail" value="$MAIL$"/><br><br>
<input type="submit" value="Add">
</form>
```

The first part of this templates is the same as example2.tpl, where the first client code that is not in the database is displayed. This time you have two fields to complete: name and email. When they are completed click on the add (submit) button and the next query will be performed.

Example5Add.qry

```
.#.EVAL
(EXITIF CID EQ "")
(DBOPEN $CID$)
.#.SQL
INSERT INTO NEWCLIENTS ( CodCli, CliName, CliMail, Sended)
                  VALUES ( $LASTCODE$, '$CliName$', '$CliMail$', 0 )
.#.SQL
SELECT MAX(CodCli)
FROM NEWCLIENTS
.#.EVAL
(SET LASTCODE $1$)
(INCREMENT LASTCODE)
```

This query first insterts into the pda database the new client with it´s code, name and email. It also sets the column Sended with a 0.
After that it queries again for the next possible client code.

Now we will take a took at the pre, post and final synchronization files.

```
PRE

.#.QRYTOSEND
FILENAME = FileToSend
SECTION = 1
SQL  = SELECT CliName, CliMail FROM NEWCLIENTS
```

In the presync we are preparing the new data (in this case the new clients from our table NEWCLIENTS). So we are selecting all the pairs Client Name and Client Mail and writing them in a file named FileToSend. [1]

In PostSync we set the rows of the table NEWCLIENTS as Sended. In this simple example we are sending all the rows in the table to the server in the synchronization, but it doesn´t need to be this way.

```
POST

.#.SQL
Update NEWCLIENTS SET Sended=1
```

In the final section we just play a sound meaning that the process is finished.

```
FINAL

.#.ACTION
DO "play sound"
URL_FILE "/windows/Alarm4.wav"
```

Finally we will see an example of a submitting script that we use in the example.

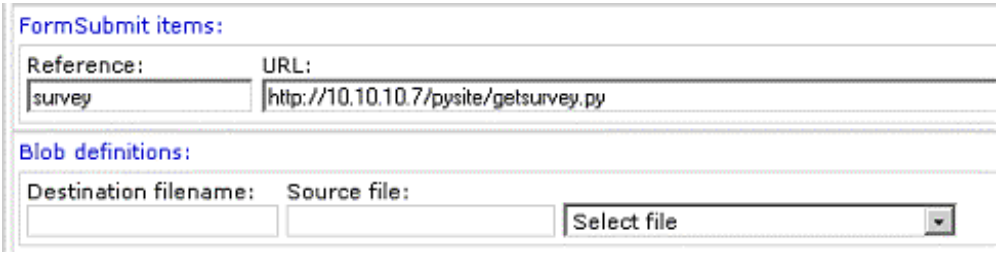
```
<%@ LANGUAGE="VBScript" %>
<%

Function BinaryToString(Binary)

  Dim cl1, cl2, cl3, pl1, pl2, pl3
  Dim L
  cl1 = 1
  cl2 = 1
  cl3 = 1
  L = LenB(Binary)

  Do While cl1<=L
    pl3 = pl3 & Chr(AscB(MidB(Binary,cl1,1)))
    cl1 = cl1 + 1
    cl3 = cl3 + 1
    If cl3>300 Then
      pl2 = pl2 & pl3
      pl3 = ""
      cl3 = 1
      cl2 = cl2 + 1
      If cl2>200 Then
        pl1 = pl1 & pl2
        pl2 = ""
        cl2 = 1
      End If
    End If
  Loop
  BinaryToString = pl1 & pl2 & pl3
End Function

'Increase Script Timeout to 1 hour
Server.ScriptTimeout = 3600

'Get size of POST data
PostSize = Request.TotalBytes

Const ForWriting =2

Set oFS = Server.CreateObject("Scripting.FileSystemObject")
Set oFSFile = oFS.OpenTextFile("C:\inetpub\wwwroot\Newclients.txt",_
                                                    ForWriting,True)


'Read POST data in 1K chunks
BytesRead = 0
For i = 1 to (PostSize/1024)
   ReadSize=1024
   PostData = Request.BinaryRead(ReadSize)
   oFSFile.Write(PostData)
   BytesRead = BytesRead + ReadSize
Next
```

```
'Read remaining fraction of 1K
ReadSize = PostSize - BytesRead
If ReadSize <> 0 Then
   PostData = Request.BinaryRead(ReadSize)
   oFSFile.Write(PostData)
   BytesRead = BytesRead + ReadSize
End If

' Send results back to client
Response.Write "Total size: "
Response.Write PostSize
Response.Write "Read size: "
Response.Write BytesRead

oFSFile.Close
Set oFSFile = Nothing
Set oFS = Nothing
%>
```

EXAMPLE 6

This is an example of how the FormSubmit items work. It will also demonstrate how this channel elements can be generated from the WireLess Mobile Server pages.

To create a FormSubmit item, URL and reference must be entered as mandatory values.



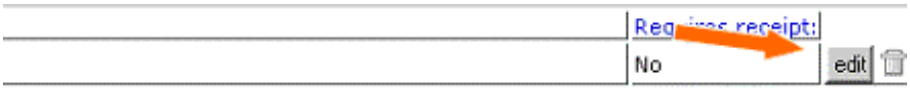
FormSubmit reference is the name of the html page resulting of the processing of the XML FormSubmit data. FormSubmit references will have the prefix **I-FSUB_** before the reference written.

The URL will be the application that will receive the data from the post sent by the client via WireLess Mobile Server at synchronization time.

To make a FormSubmit accessible from another page of the channel, it is necessary to make an hyperlink in a channel's page with href="I-FSUB_survey.html" in this case, or replacing "survey" with the chosen reference at FormSubmit creation time.

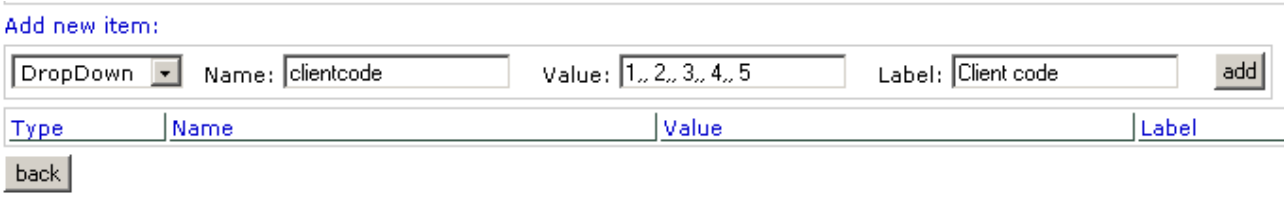
```
<a href="I-FSUB_survey.html">
      example 6
</a>:Show a FormSubmit page and generate outgoing data.
```

When added a FormSubmit item, it is possible to add some input html items defined in the FormSubmit edition page. This page can be accessed from the "edit" button of the form in the Input Editor page.



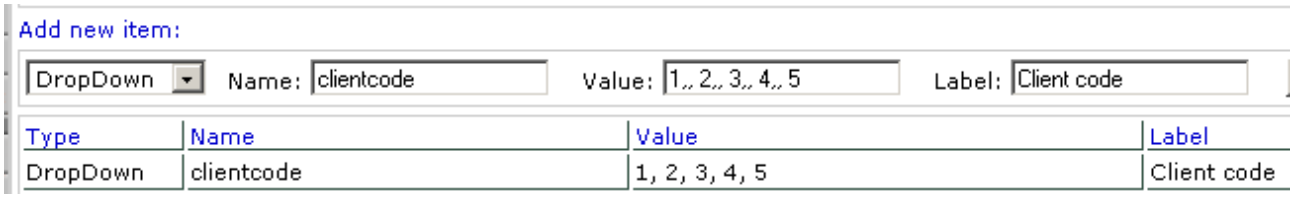
FormSubmit edition page allows to change the destination URL for the outgoing data, to choose a specific XSL to generate the html form (if not specified here, default XSL, set in Channel Management/XSL Skins page, is used), and also to add input items that will generate the values to send to the server.

Available inputs of FormSubmits are CheckBox, Dropdown Select, Text fields and Hidden inputs. Static text can also be added as items.
To add an input, choose the type from the dropdown list and write the values required.



Here, a dropdown select will be added to the form. Remember that dropdown options are set in the "Value" field, and they are separated with two commas ",," between them.

By clicking the "add" button, the item will be added to the current FormSubmit.

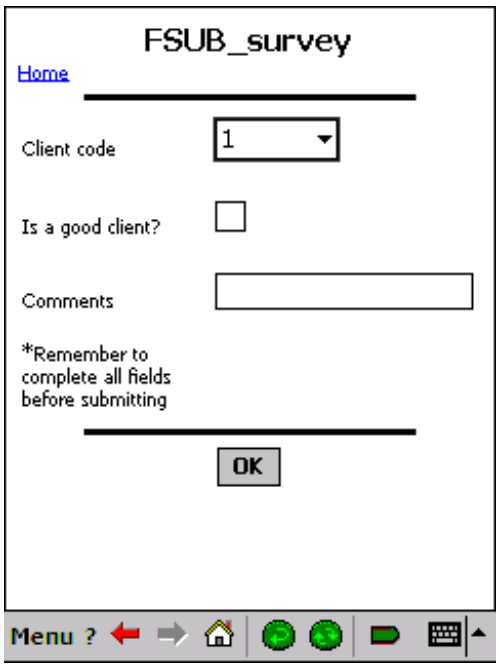


In this case, the values, coming from a dropdown item, are shown separated by commas. This is only representative. In the resulting html will be shown as different options of a select tag.

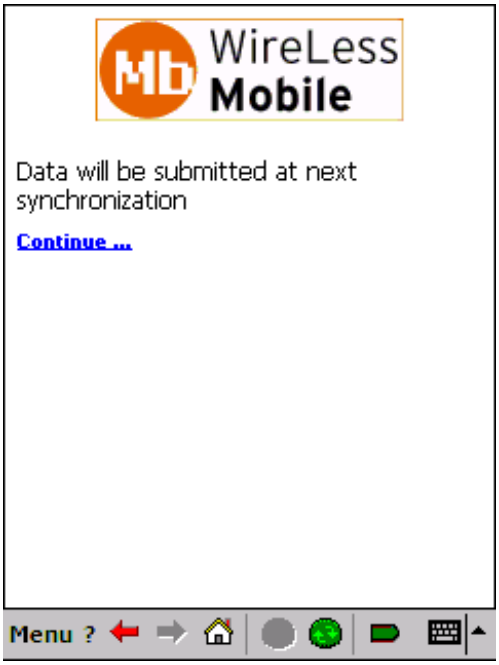
More items can be added following the same steps. The item list will look similar to:

Type	Name	Value	Label
DropDown	clientcode	1, 2, 3, 4, 5	Client code
CheckBox	goodclient	--------	Is a good client?
TextArea	comment	--------	Comments
StaticText	--------	--------	*Remember to complete all fields before submitting

The resulting page will look in the client as follows:



This html form contains the input fields specified in the FormSubmit edition page, and some fields specific to WireLess Mobile. Values sent by the POST action will be delivered to the specified URL for this form. After submitting a form with FormSubmit structure, the following page will be displayed in the client's window



DYNAMIC CHANNELS

Sometimes we want to generate different dbl´s for different users, or even work with different qry´s or tpl´s in the same channel according to the user.
Wireless Mobile can do this through the implementation of dynamic channels, that is, channels that update dynamically.

EXAMPLE

We are going to generate a dynamic channel (which content is the same as that of `MYCHANNEL`) that generates the .dbl file along with the pre, post and final files through an .asp script. This .asp will also include this files into the channel corpaliassource.xml.

1. Create a new channel called `MYCHANNELDYN` in the channel management screen.

2. Go to the user management screen and check `MYCHANNELDYN` checkbox. In the url textbox write the addres where you downloaded the file GetClients.asp that came with the example files. This file can be located anywhere, including different computers. You should edit this file to see how it does the job.

3. Add all the tpl and qry files you added to `MYCHANNEL` example (this files could also be easily added to the channel through a script)

4. Synchronize the client PDA with the new channel.

You can see that the database is similar to that of `MYCHANNEL` with some new rows (just to differenciate from the other one).

ADVANCED TOPICS

MULTIPLE SQL QUERIES

Sometimes we need to have two or more queries in the same tpl. To do this we will need an addition to the structures we have seen so far. In the qry files we can give a name to each SQL tag that appears. And then we can refer to that query by the <line:….> pre,do and post statements with the same name.

Structure of the .qry file

```
.#._EVAL
(EXITIF PATTERN EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT ...
.#.SQL <Name1>
SELECT ...
.#.SQL <Name2>
SELECT ...
.
.
.
.#.SQL <NameN>
 SELECT ...
```

There can only be one .#.SQL without a name, just for compatibility with the simple queries. But you can use a name in all queries if you want.

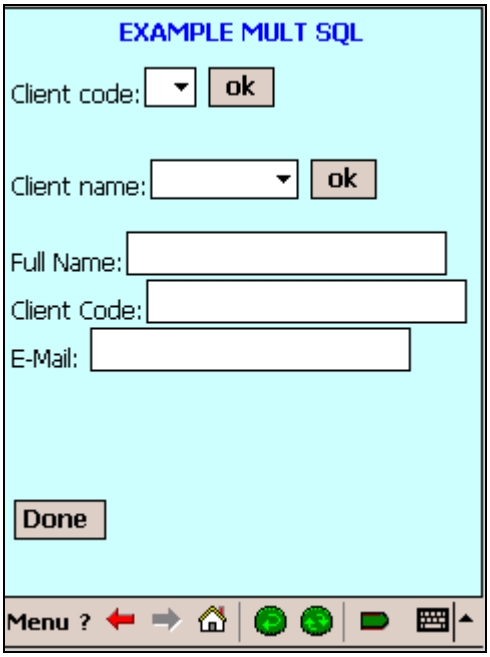
The tpl file also change to reflect this structure.

Structure of the .tpl file

```
<!--header-->
<html><XBODY><FONT >
<!--title-->

...

<!--line:pre-->
Statement
<!--line:do-->
Statements
<!--line:post-->
Statements
<!--line(<Name1>):pre-->
Statements
<!--line(<Name1>):do-->
Statements
<!--line(<Name1>):post-->
Statements
<!--line(<Name2>):pre-->
Statements
<!--line(<Name2>):do-->
Statements
<!--line(<Name2>2):post-->
Statements
.
.
.
<!--line(<NameN>):pre-->
Statements
<!--line(<NameN>):do-->
Statements
<!--line(<NameN>2):post-->
Statements


</font></XBODY>< html>
```

EXAMPLE

In this example we will have two queries of the client table in the same tpl. Both queries will show all the data of the CLIENTS table as a result, the first one searching by code and the other one searching by name.



Files Involved

To try this example create a new channel called for example MYCHANNELADV with the example files in the AdvExamples folder that you downloaded of softogo website along with MYCHANNEL example:

MultQueriesExample.tpl
MultQueriesExample.qry
MultQueriesExample2.qry
MYCHANNELADVhp.html
MYCHANNELADV.dbl

How it works

Edit the .tpl and the .qry´s files. If you take a look at them you will see that they are very simple to understand. We have two triplets of <line:> statements, named 1 and 2 in the .tpl and the corresponding SQL statements.

BIBLIOGRAPHY

You can download all the bibliography from softogo site at http://www.softogo.com .You can also see an online html version of the manuals.

[1] : **WMSynchSpecs.pdf**. Synchronization specificifications. This manual explains how to create the pre, post and final synchronization files. It also explains how the result files are generated.

[2]: **WMSpecsDbl.pdf**. DBL specifications. This manual explains the structure of .dbl files .

[3]: **WMAppTrans.pdf**. This manual explains the structure of the .tpl and .qry files.