Pre , Post And Final Synchronization

Specifications

WMSyncSpecs.doc

20051018 Version 2.0.0	Karina Spak
20040921 Version 1.0.0	Néstor Masnatta

INDEX

INTRODUCTION

This document describes the required procedures to allow executing some queries and create result files that can be sent to the server, and describe the associated structures of both: the new .mbq query files and the .mbr files to be submitted.
These features are designed in order to speed up the synchronization time. Especially when it involves a large amount of data submission from the client to the server.

MBQ FILES

There exist three files intended for each channel

```
<ChannelName>_pre.mbq
<ChannelName>_post.mbq
<ChannelName>_final.mbq
```

with the structure shown below. The post file won't be executed if the synchronization failed. But it is recommended not to include updates or deletes into the pre file, because its statements will not be reversed, once evaluated.

The tag with name precedes each statement within the file .#.SQL, by tag with name .#.EVAL or by the new tag: .#.QRYTOSEND. It is possible to access a collection of variables, which are set by calling routine or within the .mbq and can be modified there. These variables are used to declare other SQL queries. SQL queries are both in read and write mode (Select, Insert, Update, Delete, Create Table, etc.)

Script file format is as follow:

```
_____________________________
.#.<TagName1>
<statement content1>

.#.<TagName2>
<statement content2>

...
_____________________________
```

where <*statement content1*> is an SQL sentence or a set of evaluation clauses.

.#.EVAL can be one or more of

(SET <*varname*> $<*fieldpos*>$)
(INCREMENT <*varname*>)
(DBUPDATE $<*fieldpos*>$ <*varname*>)
(EXITIF <*varname*> <*operatortag*> <*testvalue*>)
for <*operatortag*>
EQ stands for ==
NE stands for !=
GE stands for >=
GT stands for >
LE stands for <=
LT stands for <
<*testvalue*> can be an integer or a (case insensitive) string.
(DBOPEN <*DbId*>)
or
(DBOPEN $<*varname*>$)

.#SQL. tag can contain any valid SQLCE/POCKET ACCESS statement.

.mbq files have the same structure of normal .qry files, but they add

.#.QRYTOSEND
FILENAME = <file_name>
SECTION=<n>
SQL=<sql_statement>

When the tag QRYTOSEND is found, a FILENAME.mbr (WireLess Mobile FormSubmit) will be created (if it was not created yet). It will be open and result data from the statement SQL will be added to section SECTION.

It is left to the application to understand the structure of each section and which data goes in each one. The same applies to the server side scripts. They will parse the files and update backend (server-side) data.

MBR FILES

This are the result files that Wireless Mobile generates processing the .mbq files. The structure is identical to the one of normal .afd files, but they have two changes. There is not the field ALAD_URL, which is replaced by the tag MB_URL, containing a page not necessarily the same as the page used to submit normal forms. This change requires a modification in WireLess Mobile Server, in order to allow the user to set this information. Besides, FORMDATA is replaced by BINARYDATA followed by CR/LF (0x0D 0x0A) characters. BINARYDATA consists of a series of SECTION's, identified as 1, 2 and so on. SECTION's are separated by the string |n|CR/FL, where n is the ID of the SECTION that follows.

Each SECTION contains data that is the result of some query SQL. CR/LF characters separate records. Commas separate fields. If a pipe ('|'), comma (',') or CR/LF appear within the data, they will be escaped as '\|' '\,' and '\n' respectively. If a backward slash '\' appears in data it will be escaped as '\\'.

WireLess Mobile Server will parse the first part of the file, and will perform a submit to the script indicated by MB_URL, sending the BINARYDATA part of the file. This script will have to get the file, parse it and update the correspondent tables.

An example of this script follows.

SCRIPT EXAMPLE IN ASP

```
<%@ 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("e:\getorders\Newclients.txt",ForWriting,True)

  oFSFile.Write( PostSize )

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

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

PRE AND POST EXAMPLE

Obs.: NBRESULT is the quantity of records returned in the last SELECT statement.

File MyChannel_pre.mbq and

```
.#.EVAL
(EXITIF CID EQ "")
(EXITIF ORDERID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT Det FROM Orders_Hdr WHERE ID = '$ORDERID$'
.#.EVAL
(EXITIF NBRESULT EQ 0)
(SET DETID $1$)
.#.QRYTOSEND
FILENAME = OrdersToSend
SECTION = 1
SQL = SELECT Det,Nam,Adr,Qty FROM Orders_Hdr WHERE ID = '$ORDERID$'
.#.QRYTOSEND
FILENAME = OrdersToSend
SECTION = 2
SQL = SELECT ID,Pro,Qty,Pri FROM Orders_Det WHERE ID = '$DETID$'
```

File MyChannel_post.mbq

```
.#.EVAL
(EXITIF CID EQ "")
(EXITIF ORDERID EQ "")
(DBOPEN $CID$)
.#.SQL
SELECT Det FROM Orders_Hdr WHERE ID = '$ORDERID$'
.#.EVAL
(SET DETID $1$)
.#.SQL
DELETE FROM Orders_Hdr WHERE ID = '$ORDERID$'
.#.SQL
DELETE FROM Orders_Det WHERE ID = '$DETID$'
```

This .mbq file can generate a .mbr like this:

```
MB_DAT_CRE  2004-09-20 12:13:14
MB_AFF_CID  MyChannel
MB_REF      FSUB_AR_LST_Client
MB_URL      http://abc.fr/getorders.asp
MB_PRV_REF  NoPreviousRef
BINARYDATA
|1|
1,City Lights Bookstore\,261 Columbus Av.\, San Francisco\nCalifornia,3
2,Shakespeare and Company\, 37 rue de la Bûcherie\n75005 Paris, 4
|2|
1,The Gold Bug,30,12\,34
1,Les Miserables,12,12.45
1,Folders for Dummies: \\ or /,7,33.44
2,El Ingenioso Hidalgo Don Quijote de la Mancha,4,23.43
2,The Purloined Letter,3,3.12
2,Du Contrat Social,14,125.99
2,Mécanique analytique,12,15.23

That means two orders:

one for
“City Lights Bookstore,261 Columbus Av., San Francisco
California”
with 3 articles:
30 copies of “The Gold Bug” (€12.34)
12 copies of “Les Miserables” (€12.45)
7 copies of “Folders for Dummies: \ or /” (€33.44)

and one for
“Shakespeare and Company, 37 rue de la Bûcherie
75005 Paris”
with 4 articles.
4 copies of “El Ingenioso Hidalgo Don Quijote de la Mancha” (€23.43)
3 copies of “The Purloined Letter” (€3.12)
14 copies of “Du Contrat Social” (€125.99)
12 copies of “Mécanique analytique” (€15.23)
```

ACTIONS IN POST SYNCHRONIZATION

A tag is provided to make some given actions when reached, such as displaying a new button at the taskbar, or playing some sort of sound. In the first case, after the button is pressed, a new `qry/tpl` could be launched to see the results. After that, the button disappears. In the second case, no additional button is shown

This tag is named
`.#.ACTION`

It can be included in any part of the post or final file (i.e.: before or after any of the other SQL or EVAL statements). It contains two different sections, named "`DO`" and "`URL_FILE`". `DO` section can contain either "`show flag`" or "`play sound`" (case insensitive. Also "`showflag`", "`show`", "`flag`", "`playsound`", "`play`" or "`sound`" are possible configurations, either between single or double quotation marks). In case `DO` is "`show flag`", `URL_FILE` must be a `URL` in the form known by *WireLess Mobile*. This one will be the URL to be displayed when the user select the "flag" button. In case `DO` is "`play sound`", `URL_FILE` must contain an absolute path for the file to be played.

EXAMPLE

File `MyChannel_post.mbq`

```
.#.SQL
SELECT Det FROM Orders_Hdr
.#.EVAL
(EXITIF NBRESULT EQ 0)
(SET DETID $1$)
.#.ACTION
DO "show flag"
URL_FILE "call://DisplayList?qry=clients.qry&tpl=clients.tpl"
```

It is assumed that `clients.qry` and `clients.tpl` both exist in the current channel folder.

File `MyChannel_final.mbq`

```
.#.ACTION
DO "play sound"
URL_FILE "/windows/Alarm1.wav"
```

REMARKS

As the only way to send parameters to the URL given is by the query string, the programmer must remember to send every required parameter there, i.e.: if `clients.qry` contains:

```
.#.EVAL
(EXITIF PATTERN EQ "")
(DBOPEN $CID$)
```

the URL must contain the parameter CID as in:

```
URL_FILE "call://DisplayList?qry=clients.qry&tpl=clients.tpl&CID=CHNDB"
```

If `URL_FILE` must be an absolute path, and folders can be specified by \\ or \ or /.