Example_Answering_Machine

(Doc in Progress NOT COMPLETE - Kyle) This is a simple answering application using javascript:

This application uses basically two scripts, which contains the general logic of the application and includes inc_*file*.js which contains the functions used:

answer.js (includes: inc_answer.js inc_logger.js)

vmListen.js (includes: inc_vmListen.js inc_logger.js)

All of these .js files should be added to the scripts directory.

answer.js

```
 include("inc_answer.js");
 include("inc_logger.js");

 use("TeleTone");
 use("etpan");

// * answermachine.js
// * Written by: Joshua Engelbrecht for use with FreeSwitch
// * Based on Mike B. Murdock script which was
// * Based on original samples by Anthony Minessale II
// * Revision: 02-20-2007
//
// * Version: MPL 1.1
// *
// * The contents of this file are subject to the Mozilla Public License Version
// * 1.1 (the "License"); you may not use this file except in compliance with
// * the License. You may obtain a copy of the License at
// * http://www.mozilla.org/MPL/
// *
// * Software distributed under the License is distributed on an "AS IS" basis,
// * WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License
// * for the specific language governing rights and limitations under the
// * License.
// *

// Note: this script uses mod_spidermonkey_etpan which must be compiled and loaded.
// Additionally, it requires sendmail be setup and configured on your server.

// uncomment next two lines if you want email notifications
// var eMailFrom = "you@domain.org"; // Where are your emails coming from
// var eMailTo = "joshe@coppercom.com"; // List email addresses to send message to. Separate mu

// Caller Name and Number will be attached
var eMailSubject = "Voicemail from ";
// Text of email body
var eMailBody = "New voicemail attached.\n";

// Uncomment next line to send a notification without a file attachment. Usually cell phone not
// var Pager_eMailTo = "3165551212@mobile.att.net";

// location of voicemail files. In a high performance app this might be a ram drive.
var vmDir = new File("/var/spool/freeswitch"); // should not end in slash

// find account number
var vmAccount = session.destination;
// test for account folder;
```

```
var dir = new File(vmDir + "/" + vmAccount);

// make sure that the user has a directory if not create
if(!dir.isDirectory){
    vmDir.mkdir(vmAccount);
}

// where the greeting is each user must have their own greeting
var vMailGreetingFile = dir + "/" + "greeting.wav";

//var BONG = "v=4000;>=0;+=2;#(60,0);v=2000;d(940,0)";
var BONG = "v=4000;>=0;+=2;#(400,0);";

// Maximum recording length in seconds (240 = 4 minutes should be enough)
var maxreclen = 240;

// Energy level audio must fall below to be considered silence (300-500 is good starting point)
var silencethreshold = 500;

// Amount of time in seconds caller must be silent to trigger detector
var silencehits = 5; // 3 seconds

var allDigits = "";

// If not answered answer the call
session.answer();

if (session.ready()) {
    // save the session properties for later use in case caller hangs up and we need the data.
    var sv_uuid = session.uuid;
    var sv_ani = session.ani;
    var sv_ani2 = session.ani2;
    var sv_caller_id_name = session.caller_id_name + " ";
    var sv_caller_id_num = session.caller_id_num + " ";
    var sv_destination = session.destination;
    var sv_dialplan = session.dialplan;
    var sv_name = session.name;
    var sv_network_addr = session.network_addr;
    var sv_state = session.state;

    fsLogger("Caller ID = " + sv_caller_id_name + "<" + sv_caller_id_num + ">\n");

    allDigits = "";

    // Play announcement
    var rtn;
    // session.streamFile(file, callback, callback_args, starting_sample_count);
    var vMGF = new File(vMailGreetingFile);
    if(vMGF.isFile) {
       rtn = session.streamFile(vMailGreetingFile, on_dtmf, "");
    } else {
       session.sayPhrase("speak", "Please leave a message after the beep", "en");
    }
    fsLogger("session.streamFile rtn=[" + rtn + "]");

    // verify that the caller is still here
    if (session.ready()) {

       // Pause for 1/10th second
       rtn = session.execute("sleep", "100");

       // Play bong
```

```
        var tts = new TeleTone(session);
        tts.addTone("d", 350.0, 440.0, 0.0);
        tts.generate(BONG);

        // Record message
        fsLogger("","Recording message");
        var tmp_Filename = dir + "/" + sv_uuid + ".wav";

        // Caller still here?
        if (session.ready()) {
           var toDate=new Date();
           rtn = session.recordFile(tmp_Filename, on_dtmf, "", maxreclen, silencethreshold, sile
           fsLogger("", "session.recordFile rtn=[" + rtn + "]");

           // create meta file
           var fd = new File(dir + "/" + sv_uuid + ".meta");
           fsLogger("Creating META file:","START\n");
           fd.open("write,create");
           fd.writeln("callerID="+session.caller_id_num);
           fd.writeln("callerName="+session.caller_id_name);
           fd.writeln("createdDay="+toDate.getDay());
           fd.writeln("createdDate="+toDate.getMonth() + " " + toDate.getDate() + ", " + toDate.
           fd.writeln("createdTime="+toDate.getHours() + ":" + toDate.getMinutes() + ":" + toDat
           fd.writeln("heard="+"false");
           fd.writeln("markForDeletion="+"false");
           fd.close;
           fsLogger("Creating META file:", "END\n");

           // Caller still here?
           if (session.ready()) {
              session.hangup // Hangup
            }
            else {
              fsLogger("","Caller Hungup during record");
            }

            // Send notifications
            // uncomment specific function(s) to send notifications
            // emailNotification();
            // send email pager notification
            // sendPageNotification();
        }
    }
}
```

inc_answer.js

```
function on_dtmf(type, digits, arg)
{
    if (digits == "#") {
        return allDigits;
    }

    if (digits == "*") {
        return "hangup";
    }

    console_log("digit: " + digits + "\n");
    allDigits += digits;
```

```
    return(allDigits);
}

function emailNotification() {
   if (eMailTo != "") {
      console_log(">>>>>>>>>>>>>>>>>>>>>>>>>>>>> Sending Email\n");
      var tmp_eMailSubject = eMailSubject + sv_caller_id_name + " (" + sv_caller_id_num + ")";
      email(eMailFrom, eMailTo, "Subject: " + tmp_eMailSubject, eMailBody, tmp_Filename);

      // construct the email notification to cell phone gateway - no file attached
      if (Pager_eMailTo != "") {
         email(eMailFrom, Pager_eMailTo, "Subject: " + tmp_eMailSubject, tmp_eMailSubject);
      }
   }
}

function sendPageNotification() {
   if (Pager_eMailTo != "") {
      console_log(">>>>>>>>>>>>>>>>>>>>>>>>>>>>> Sending Page\n");
      email(eMailFrom, Pager_eMailTo, "Subject: " + tmp_eMailSubject, tmp_eMailSubject);
   }
}
```

inc_logger.js

```
//  inc_logger.js
//  logging function function library

//  usage fsLogger("","");

function fsLogger(descriptor, message) {
    console_log(" >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> " + descriptor + " >>>>>>>>> " + message +"\
}
```

vmListen.js

```
include("inc_logger.js");           // the logging functions
include("inc_vmListen.js");         // loading functions
include("inc_settings_test.js");

var result = "";

if (session.ready()) {
    session.answer();

    session.execute("sleep","2000");
    session.sayPhrase("speak", "welcome to the  v m listen application", "en");

    var dir = new File(vmMessageDir + "/" + vmAccount);

    var numNewMessages = 0;
    var numOldMessages = 0;
    var currentNewNum = 0;
    var currentOldNum = 0;
    var newMessagesArray = new Array();
    var oldMessagesArray = new Array();

// Reads the voicemail directory and sorts messages

    if (dir.isDirectory) {
```

```
        var dirContent = dir.list();
        var numFiles = dirContent.length;

        for (i = 0; i < numFiles; i++) {
            var contentFile = dirContent[i];
            var strThisFile = contentFile.name;
            var fileName = strThisFile.substring(strThisFile.lastIndexOf("/"),strThisFile.last
            var fileSuffix = strThisFile.substring(strThisFile.lastIndexOf("."));
            fsLogger(i +" FILE ", fileName);
            fsLogger(i +" DATE ", contentFile.creationTime);
            fsLogger("fileSuffix", fileSuffix);
            if (fileSuffix == ".meta") {
                var infoObj = metaFileRead(vmAccount,fileName);
                fsLogger("infoObj.heard",infoObj.heard);
                if (infoObj.heard == "true") {
                    oldMessagesArray[numOldMessages] = fileName;
                    numOldMessages++;
                } else {
                    newMessagesArray[numNewMessages] = fileName;
                    numNewMessages++;
                }
            }
        }
    } else {
        fsLogger("", dir.name + " is NOT a directory");
        session.sayPhrase("speak", "No directory was found for this account ", "en");
    }

// Announcement of Message Summary
    announceMessageSummary(numNewMessages, numOldMessages);

// Playback for NEW Messages

    if (numNewMessages > 0) {
        while (currentNewNum < numNewMessages) {
            var fileName = newMessagesArray[currentNewNum];
            var vmMessageMeta = fileName + ".meta";
            var infoObj = metaFileRead(vmAccount,fileName);
            currentNewNum++;
            dtmf = session.sayPhrase("speak", "Message " + currentNewNum + " Recieved on " + i
               + " from " + validateNull(infoObj.callerName) + ", press 1 to play,  press 2 to
            session.execute("sleep","1000");
            if (dtmf == "1") {
                messagePlay(vmAccount,fileName);
                messageHeard(infoObj);
            }
            if (dtmf == "2") {
                session.sayPhrase("speak", "Skipping this message ", "en");
            }
            if (dtmf == "3") {
                messageDelete(vmAccount,fileName);
            }
        }
        session.sayPhrase("speak", "This is the end of your new messages ", "en");
    } else {
        session.sayPhrase("speak", "Skipping to old messages ", "en");
    }

// Playback for Old Messages

    if (numOldMessages > 0) {
        while (currentOldNum < numOldMessages) {
```

```
            var fileName = oldMessagesArray[currentOldNum];
            var vmMessageMeta = fileName + ".meta";
            var infoObj = metaFileRead(vmAccount,fileName);

            currentOldNum++;
            dtmf = session.sayPhrase("speak", "Message " + currentOldNum + " recieved on " + i
               + " from " + validateNull(infoObj.callerName) + " press 1 to play,  press 2 to
            session.execute("sleep","1000");
            if (dtmf == "1") {
                messagePlay(vmAccount,fileName);
            }
            if (dtmf == "2") {
                session.sayPhrase("speak", "Skipping this message ", "en");
            }
            if (dtmf == "3") {
                messageDelete(vmAccount,fileName);
            }
            session.execute("sleep","2000");
        }
        session.sayPhrase("speak", "This is the end of your old messages ", "en");
    } else {
        session.sayPhrase("speak", "No old messages ", "en");
    }

    session.execute("sleep","1000");
    session.sayPhrase("speak", "this is the end of the application", "en");
    session.execute("sleep","1000");
    session.sayPhrase("speak", "Good bye", "en");
    session.hangup;
}
```

inc_vmListen.js

```
function validateNull(thisString) {
    if ((thisString == null) || (thisString == "")) {
        return "No Data";
    }
    return thisString;
}

function menu_dtmf(type, digits, arg) {
   if (type == "dtmf") {
       if (digits == "3") {
           return digits;
       }
       if (digits == "1") {
           return digits;
       }
       if (digits == "2") {
           return digits;
       }
       if (digits == "*") {
           return "hangup";
       }
   }
   return(true);
}

function play_dtmf(type, digits, arg) {
```

```
   if (type == "dtmf") {
       if (digits == "3") {
           return false;
       }
       if (digits == "1") {
           return "seek:-5000";
       }
       if (digits == "5") {
           return "pause";
       }
       if (digits == "*") {
           return "hangup";
       }
   }
   return(true); // return false to cause playback to stop
}

function getNumNewMessages(dir) {
    if (dir.isDirectory) {
        var dirContent = dir.list();
        var numMessages = dirContent.length;

        fsLogger("",dir.name + " is a directory");
        fsLogger("","number of " + numMessages);

        for (i = 0; i < numMessages; i++) {

            var contentFile = dirContent[i];
            var wavFile = contentFile.name;
            var fileName = wavFile.substring(wavFile.lastIndexOf("/"));

            fsLogger(i +" wav File ", wavFile);
            fsLogger(i +" FILE ", fileName);
            fsLogger(i +" DATE ", contentFile.creationTime);
        }
        return numMessages;
    } else {
        fsLogger("", dir.name + " is NOT a directory");
        return 0;
    }
}

function messageDelete(vmAccount,fileName) {
    var delFile = File(vmMessageDir + "/" + vmAccount + "/" + fileName + ".wav");
    delFile.remove();
    var delFile = File(vmMessageDir + "/" + vmAccount + "/" + fileName + ".meta");
    delFile.remove();
    session.sayPhrase("speak", "this message has been deleted", "en");
}

function messageHeard(metaObj) {
    var metaFile = new File(vmMessageDir + "/" + vmAccount +"/" + fileName + ".meta");
    if (metaFile.exists) {
        metaFile.remove();
    }
    metaFile.open("write,create");
    metaFile.writeln("callerID="+metaObj.callerID);
    metaFile.writeln("callerName="+metaObj.callerName);
    metaFile.writeln("createdDay="+metaObj.createdDay);
    metaFile.writeln("createdDate="+metaObj.createdDate);
    metaFile.writeln("createdTime="+metaObj.createdTime);
    metaFile.writeln("heard="+"true");
```

```
    metaFile.writeln("markForDeletion="+metaObj.markForDeletion);
    metaFile.close;
}

function messagePlay(vmAccount,fileName) {
    var vmMessageWav = fileName + ".wav";
    session.sayPhrase("NowPlaying", i+1,"en");
    session.streamFile(vmMessageDir + "/" + vmAccount + "/" +  vmMessageWav, play_dtmf, "", 0)
    fsLogger(i +" FILE ", fileName);
}

function metaFileRead(vmAccount,fileName) {
    var metaFile = new File(vmMessageDir + "/" + vmAccount +"/" + fileName + ".meta");
    metaFile.open("read");
    var metaObj = new Object();
    var tmpString = metaFile.readln();
    metaObj.callerID = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.callerName = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.createdDay = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.createdDate = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.createdTime = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.heard = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.markForDeletion = tmpString.substring(tmpString.lastIndexOf("=") +1);
    metaFile.close();
    fsLogger("metaObj",metaObj);
    return metaObj;
}

function metaFileWrite(vmAccount,fileName,metaObj) {
    var metaFile = new File(vmMessageDir + "/" + vmAccount +"/" + fileName + ".meta");
    if (metaFile.exists) {
        metaFile.remove();
    }
    metaFile.open("write,create");
    metaFile.writeln("callerID="+session.caller_id_num);
    metaFile.writeln("callerName="+session.caller_id_name);
    metaFile.writeln("createdDay="+toDate.getDay());
    metaFile.writeln("createdDate="+toDate.getMonth() + " " + toDate.getDate() + ", " + toDate
    metaFile.writeln("createdTime="+toDate.getHours() + ":" + toDate.getMinutes() + ":" + toDa
    metaFile.writeln("heard="+"false");
    metaFile.writeln("markForDeletion="+"false");
    metaFile.close;
}

function messagePlay(vmAccount,fileName) {
    var vmMessageWav = fileName + ".wav";
    session.sayPhrase("NowPlaying", i+1,"en");
    session.streamFile(vmMessageDir + "/" + vmAccount + "/" +  vmMessageWav, play_dtmf, "", 0)
    fsLogger(i +" FILE ", fileName);
}

function metaFileRead(vmAccount,fileName) {
    var metaFile = new File(vmMessageDir + "/" + vmAccount +"/" + fileName + ".meta");
    metaFile.open("read");
    var metaObj = new Object();
    var tmpString = metaFile.readln();
```

```
    metaObj.callerID = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.callerName = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.createdDay = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.createdDate = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.createdTime = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.heard = tmpString.substring(tmpString.lastIndexOf("=") +1);
    tmpString = metaFile.readln();
    metaObj.markForDeletion = tmpString.substring(tmpString.lastIndexOf("=") +1);
    metaFile.close();
    fsLogger("metaObj",metaObj);
    return metaObj;
}

function metaFileWrite(vmAccount,fileName,metaObj) {
    var metaFile = new File(vmMessageDir + "/" + vmAccount +"/" + fileName + ".meta");
    if (metaFile.exists) {
        metaFile.remove();
    }
    metaFile.open("write,create");
    metaFile.writeln("callerID="+session.caller_id_num);
    metaFile.writeln("callerName="+session.caller_id_name);
    metaFile.writeln("createdDay="+toDate.getDay());
    metaFile.writeln("createdDate="+toDate.getMonth() + " " + toDate.getDate() + ", " + toDate
    metaFile.writeln("createdTime="+toDate.getHours() + ":" + toDate.getMinutes() + ":" + toDa
    metaFile.writeln("heard="+"false");
    metaFile.writeln("markForDeletion="+"false");
    metaFile.close;
}
```

inc_settings_test.js

```
console_log(" >>>>>>>>>>>>>>>>>>> USING INC TEST SETTINGS <<<<<<<<<<<<<<<<<<<<<<<< \n");

// Freeswitch Style
var vmMessageDir = "/var/spool/freeswitch";
var vmAccount = session.caller_id_num;

fsLogger("session.caller_ani",session.caller_ani);
fsLogger("session.caller_ani2",session.caller_ani2);
fsLogger("session.caller_id_name",session.caller_id_name);
fsLogger("session.caller_id_num",session.caller_id_num);
fsLogger("session.destination",session.destination);
fsLogger("session.name",session.name);
fsLogger("session.state",session.state);
```

lang_en.xml (for the speak phrase using in the js script)

```
<macro name="speak">
  <input pattern="(.*)">
    <match>
      <action function="speak-text" data="$1"/>
    </match>
  </input>
```

```
</macro>
```

The files structure starts at **/var/spool/freeswitch** then directories are created as messages are left. The directory will be give the same name as the registered phone extension. Within the directory, messages left will be given the id of the call session with a .wav extension along with a file with a .meta extension. The .meta file is a text file that will contain the callerID, callerName, createdDay, createdDate, createdTime, heard flag and a markForDeletion flag.

The **/var/spool/freeswitch** will be needed to be created manually but the directory underneath will be created as messages are left.

Example of .meta file:

```
callerID=2666
callerName=Kyle's PC
createdDay=4
createdDate=1 22, 2007
createdTime=8:12:39
heard=true
markForDeletion=false
```

See Also

- Javascript