Contents

- 1 Credit
- 2 Summary
- 3 Features
- 4 Issues
- 5 ODBC
- 6 Dial Plan
- 7 Conference Configuration File
- 8 SQL Table
- 9 JavaScript IVR Code
- 10 See Also

Credit

The background for this code came from confcall.js. Thanks to the person who wrote that because it gave me a good template to work from.

Summary

This code was written to implement a publicly available conferencing system that provided "secure" conference rooms that prevent abuse from outside parties. The security comes from the fact that while the conference room is always available, it's unusable until the owner (a.k.a moderator) enters. The system also pulls its conference information from a database and some extra features were added at the request of the users.

The FreeSWITCH conference application (mod_conference), while clean and robust, does not support a number of the bells and whistles that the Asterisk MeetMe application does. This code takes advantage of the FreeSWITCH API, console commands, and the new wait-mod conference-flag (added in build 13442) to mimic some of MeetMe's functionality.

Features

- Provides "secure" conference rooms which are provisioned in a database table.
- Prompts for a conference "room" number, then prompts for a password to mark the moderator. Anyone who does not provide the proper password is considered a "user". Any user who enters the room before the moderator will hear "the conference will begin when the leader arrives", then music until the moderator enters.

A custom prompt is useful to say something like "If you are the conference moderator, enter your password now, otherwise just press the pound key to be joined to the conference."

- Voices the count of other participants to the caller when they enter.
- Marks the moderator with a UUID variable for internal use and so other tools might leverage it later.
- Executes a dialplan transfer to the conference which is more efficient than calling the conference application directly from JavaScript (according to a mailing list post).

- Quite a bit of debug output so you can watch the action or troubleshoot problems.

Issues

- Some of the voice prompt WAV files referenced within the code were custom-made, so you may have to substitute the names for something you have installed.
- Currently, the code ONLY supports secure conference rooms i.e. they ALL utilize the wait-mod feature and require the use of a password. It would probably be easy to add support for unsecured rooms within the same code, but I didn't have a need for that. (To handle this, you could: Check for an empty password in the database, if it's empty, don't prompt for a password, and enter ALL callers as moderators.)
- The wait-mod conference flag DOES NOT put the users back to music if the moderator drops from the conference. Other conferencing systems do this, but FreeSWITCH's implementation does not. You could possibly use the 'endconf' flag for the moderators which would tear down the conference if all moderators left, but I preferred not to do this.
- If a user enters before a moderator and is listening to music, they can hear the ding sounds of other users entering the conference over the music. This is neither good or bad in my opinion, just a little different.

ODBC

Getting ODBC to work was somewhat tricky - refer to Mod spidermonkey odbc.

Dial Plan

These are the only two extensions that are required for this implementation.

```
<!-- This is where you should drop callers destined for the conferencing IVR -->
<extension name="Conference JavaScript IVR">
   <condition field="destination_number" expression="^9999$">
      <action application="javascript" data="conf-ivr.js"/>
   </condition>
</extension>

<!-- This is used within conf-ivr.js once a caller successfully gets through the IVR -->
<extension name="Conference JavaScript Transfer Destination">
   <condition field="destination_number" expression="^conf-room--(.*)$">
      <action application="conference" data="$1"/>
   </condition>
</extension>
```

Conference Configuration File

Make absolutely sure the following line is included in the conference profile within conference.conf.xml:

```
<param name="conference-flags"  value="wait-mod"/>
```

(That line is what enables the "moderator wait" feature.)
In the JavaScript code, the transfer sends callers into the conference using the "default" profile. If you wish to use other conferences which do not utilize this feature you'll need to create another profile and change the names accordingly.

Also, comment out the "alone-sound" parameter.

```
<!--<param name="alone-sound" value="conference/conf-alone.wav"/>-->
```

The reason for commenting this feature is due to the behavior of the wait-mod conference flag and how I wanted the voicing of user counts to work. With wait-mod enabled, the first user enters the conference and gets music, but hears nothing about whether they are first or not. I wanted the first person to KNOW they were first (it can help if you punched in the wrong conference number), so I implemented that in the Say_Other_Participant_Count() function.

SQL Table

I was using MySQL for my database, although anything will work via ODBC. Create a table using the SQL code below...

```
CREATE TABLE  `freeswitch`.`conferences` (
  `autonumber`   int(11) NOT NULL auto_increment,
  `conf_room`    varchar(10) NOT NULL,
  `mod_password` varchar(10) NOT NULL,
  `conf_name`    varchar(30) NOT NULL,
  PRIMARY KEY  USING BTREE (`autonumber`,`conf_room`)
) DEFAULT CHARSET=latin1;
```

JavaScript IVR Code

Create the file: /freeswitch-dir/scripts/conf-ivr.js
Copy the code below and paste it into that file.

```
//==================================================================================
// Environment
//==================================================================================
use("ODBC");
var USERTYPE__INVALID   = 0;
var USERTYPE__USER      = 1;
var USERTYPE__MODERATOR = 2;


//==================================================================================
// Function :: Debug output
//==================================================================================
function Debug(text)
{
   var uuid_last5 = session.uuid.substr(-5)
   console_log("info", "["+uuid_last5+"] "+text+"\n");
}
```

```
//==========================================================================================
// Function :: Voices the count of other participants in the conference
//==========================================================================================
function Say_Other_Participant_Count(participant_count)
{
   if(participant_count == 0)
   {
      Debug("This is the first person in the conference");
      session.streamFile("conference/conf-onlyperson.wav");
   }
   else
   {
      Debug("There are ["+participant_count+"] other participants in the conference");
      session.streamFile("conference/conf-thereare.wav");
      session.execute("say","en number pronounced "+ participant_count);
      session.streamFile("conference/conf-otherinparty.wav");
   }
}


//==========================================================================================
// Function :: Determine how many other participants are already in the conference
//==========================================================================================
function Get_Participant_Count(confroom)
{
   var output = apiExecute("conference", confroom+" list");

   //-------------------------------------------------------
   // Is the conference active?
   //-------------------------------------------------------
   if(output.indexOf("not found") == -1)
   {
      //-------------------------------------------------------
      // Count the number of line-feed chars and subtract 1.
      // This is tacky, but couldn't find a better way...
      //-------------------------------------------------------
      var line_count = output.split("\n").length - 1;
      return line_count;
   }

   return 0;
}


//==========================================================================================
// Function :: Scans the users in the conference to see if a moderator is present
//==========================================================================================
function Is_Moderator_In_Room(confroom)
{
   var output = apiExecute("conference", confroom+" list");

   //-------------------------------------------------------
   // If the conference is active, scan the UUIDs in
   // the conference for the presence of a moderator
   //-------------------------------------------------------
   if(output.indexOf("not found") == -1)
   {
      var lines = output.split(/\n/);

      for(var x = 0; x < lines.length-1; x++)
```

```
      {
         var line   = lines[x];
         var fields = line.split(";");
         var uuid   = fields[2];
         var result = apiExecute("uuid_getvar", uuid+" ConferenceIVR_IsModerator");

         if(result == "1")
         {
            //Debug("Moderator is present in the room");
            return true;
         }
      }
   }

   //Debug("Moderator is NOT present in the room");
   return false;
}


//==================================================================================
// Function :: Iterates over all the users in the conference and sets a flag
//             indicating that the conference has been opened by the moderator.
//             This is a crappy way of handling this but there is no way to set a
//             "flag" on the conference itself, so, we set it on all users and when
//             we check for that flag only ONE user needs to have it set.
//==================================================================================
function Set_User_Conference_Room_Opened_Flags(value)
{
   Debug("Setting RoomOpen flag on all users");

   var output = apiExecute("conference", confroom+" list");

   //-----------------------------------------------------
   // If the conference is active, iterate over
   // the UUIDs and set a flag variable
   //-----------------------------------------------------
   if(output.indexOf("not found") == -1)
   {
      var lines = output.split(/\n/);

      for(var x = 0; x < lines.length-1; x++)
      {
         var line   = lines[x];
         var fields = line.split(";");
         var uuid   = fields[2];

         apiExecute("uuid_setvar", uuid+" ConferenceIVR_RoomHasBeenOpened "+value);
      }
   }
}


//==================================================================================
// Function :: Scans the users in the conference to see if a flag has been set.
//             If a single user in the conference has the ConferenceIVR_RoomHasBeenOpened
//             flag set, it means that the moderator entered this conference at some
//             point in time.  We need this information because if the moderator
//             leaves at some point, we have no status information from the conference
//             to let us know if wait_mod got cleared or not.
//==================================================================================
function Has_Conference_Room_Been_Opened(confroom)
{
```

```
   Debug("Checking all users for RoomOpen flag");

   var output = apiExecute("conference", confroom+" list");

   //-----------------------------------------------------
   // If the conference is active, scan the UUIDs in
   // the conference for the presence of a moderator
   //-----------------------------------------------------
   if(output.indexOf("not found") == -1)
   {
      var lines = output.split(/\n/);

      for(var x = 0; x < lines.length-1; x++)
      {
         var line   = lines[x];
         var fields = line.split(";");
         var uuid   = fields[2];
         var result = apiExecute("uuid_getvar", uuid+" ConferenceIVR_RoomHasBeenOpened");

         if(result == "1")
         {
            Debug("Conference has been opened by moderator"); return true;
         }
      }
   }

   Debug("Conference has NOT YET been opened by moderator");
   return false;
}


//==================================================================================
// Function :: Replace all occurrences of a char in a string
//             Why doesn't JavaScript provide this natively?  Duh.
//==================================================================================
function ReplaceAll(text, str1, str2)
{
   while(text.indexOf(str1) != -1) { text = text.replace(str1, str2); }
   return text;
}


//==================================================================================
// Function :: Callback handlers for collectInput
//             Returning false from these handlers causes the digit collection
//             to "unblock".  Depending on what behavior you want to achieve
//             you might want to unblock on any input or on a specific digit.
//==================================================================================
function Get_DTMF_Stop_On_Any(session, type, data, arg)
{
   //Debug("DTMF: "+data.digit+"");
   arg.digits += data.digit;
   return false;
}

function Get_DTMF_Stop_On_Pound(session, type, data, arg)
{
   //Debug("DTMF: "+data.digit+"");
   arg.digits += data.digit;
   if(data.digit == '#') { return false; }
   else                  { return true;  }
}
```

```
//==================================================================================
// Function :: Database lookup to check conference rooms
//==================================================================================
function DB_Check_Conference_Room(confroom)
{
   db.connect();
   var returnval = false;

        sql = "select conf_name from conferences where conf_room = '"+confroom+"'";
        db.exec(sql);

   if(db.numRows() == 1)
   {
      db.nextRow();
      var row      = db.getData();
      var confname = row["conf_name"];

      Debug("Room ["+confroom+"] is valid, it is called ["+confname+"]");
      returnval = true;
   }
   else if(db.numRows() == 0)
   {
      Debug("Room ["+confroom+"] was not found in the database");
      returnval = false;
   }
   else
   {
      Debug("Room ["+confroom+"] returned a weird amount of rows, strange things are afoot");
      returnval = false;
   }

   db.disconnect();
   return returnval;
}


//==================================================================================
// Function :: Database lookup to check moderator passwords
//==================================================================================
function DB_Check_Moderator_Password(confroom, password)
{
   db.connect();
   var returnval = USERTYPE__INVALID;

   if(password == "")
   {
      returnval = USERTYPE__USER;
   }
   else
   {
      sql = "select conf_room, mod_password from conferences where conf_room = '"+confroom+"' and
      db.exec(sql);

      if(db.numRows() == 1)
      {
         returnval = USERTYPE__MODERATOR;
      }
      else if(db.numRows() == 0)
      {
         Debug("Password ["+password+"] was not valid for Room ["+confroom+"]");
```

```
         returnval = USERTYPE__INVALID;
      }
      else
      {
         Debug("Password ["+password+"], Room ["+confroom+"] returned a weird amount of rows, str
         returnval = USERTYPE__INVALID;
      }
   }

   db.disconnect();
   return returnval;
}



//==========================================================================================
//==========================================================================================
//==========================================================================================
//==========================================================================================


//==============================================================================
// Main Section
//==============================================================================
var dsn     = "freeswitch-conferences";
var db_user = "user";
var db_pass = "password";
var db      = new ODBC(dsn, db_user, db_pass);

session.answer();

//-----------------------------------------------------
// Setup vars and enter processing loop
//-----------------------------------------------------
var confroom_dtmf          = new Object();
var confroom_prompt_count  = 0;
var confroom_prompt_limit  = 3;
var password_dtmf          = new Object();
var password_prompt_count  = 0;
var password_prompt_limit  = 3;

while(session.ready())
{
   //-----------------------------------------------------
   // Are we under the confroom prompt limit?
   //-----------------------------------------------------
   if(++confroom_prompt_count <= confroom_prompt_limit)
   {
      //-------------------------------------------------------
      // Flush digits, ask for Conference Room
      //-------------------------------------------------------
      confroom_dtmf.digits = "";
      session.flushDigits();

      Debug("Prompting for Conference Room...");
      session.streamFile("conference/conf-getconfno.wav", Get_DTMF_Stop_On_Any, confroom_dtmf);
      if(confroom_dtmf.digits != "#") { session.collectInput(Get_DTMF_Stop_On_Pound, confroom_dtm
      Debug("Conference Room Digits Collected = ["+confroom_dtmf.digits+"]");

      //-------------------------------------------------------
      // Is that a good conference room?
      //-------------------------------------------------------
```

```
var confroom        = ReplaceAll(confroom_dtmf.digits, "#", "");
var confroom_result = DB_Check_Conference_Room(confroom);

if(confroom_result)
{
   session.streamFile("conference/conf-thank-you.wav");

   while(session.ready())
   {
      //-----------------------------------------------------
      // Are we under the password prompt limit?
      //-----------------------------------------------------
      if(++password_prompt_count <= password_prompt_limit)
      {
         //-------------------------------------------------------
         // Flush digits, ask for Password
         //-------------------------------------------------------
         password_dtmf.digits = "";
         session.flushDigits();

         Debug("Prompting for Moderator Password...");
         session.streamFile("conference/conf-getconfpass.wav", Get_DTMF_Stop_On_Any, passwo
         if(password_dtmf.digits != "#") { session.collectInput(Get_DTMF_Stop_On_Pound, pas
         Debug("Password Digits Collected = ["+password_dtmf.digits+"]");

         //-------------------------------------------------------
         // Is that a good password?
         //-------------------------------------------------------
         var password        = ReplaceAll(password_dtmf.digits, "#", "");
         var password_result = DB_Check_Moderator_Password(confroom, password);

         switch(password_result)
         {
            //-------------------------------------------------------
            case USERTYPE__MODERATOR:
            //-------------------------------------------------------
               Debug("Putting caller into room ["+confroom+"] as a [MODERATOR]");

               var participant_count = Get_Participant_Count(confroom);
               Say_Other_Participant_Count(participant_count);

               Set_User_Conference_Room_Opened_Flags(1);
               session.setVariable("ConferenceIVR_RoomHasBeenOpened", "1");
               session.setVariable("ConferenceIVR_IsModerator",       "1");

               session.flushDigits();
               session.execute("transfer", "conf-room--"+confroom+"@default+flags{moderator
               break;

            //-------------------------------------------------------
            case USERTYPE__USER:
            //-------------------------------------------------------
               Debug("Putting caller into room ["+confroom+"] as a [USER]");

               var participant_count = Get_Participant_Count(confroom);
               Say_Other_Participant_Count(participant_count);

               if(Is_Moderator_In_Room(confroom))
               {
                  session.setVariable("ConferenceIVR_RoomHasBeenOpened", "1");
                  Debug("Moderator is in the room, going in");
               }
```

```
                    else
                    {
                       if(Has_Conference_Room_Been_Opened(confroom))
                       {
                          Debug("Moderator is NOT in the room, but room is open, going in");
                       }
                       else
                       {
                          Debug("Moderator has NOT opened the conference, waiting");
                          session.streamFile("conference/conf-waitforleader.wav");
                       }
                    }

                    session.setVariable("ConferenceIVR_IsModerator", "0");
                    session.flushDigits();
                    session.execute("transfer", "conf-room--"+confroom+"@default");
                    break;

                 //-------------------------------------------------------
                 case USERTYPE__INVALID:
                 //-------------------------------------------------------
                 default:
                    // User entered a bad moderator password
                    session.streamFile("conference/conf-password-incorrect.wav");
                    break;
              }
           }
           else
           {
              // User exceeded the password prompt limit, say bye-bye
              session.streamFile("conference/conf-goodbye.wav");
              session.hangup();
           }
        }
     }
     else
     {
        // User entered a bad conference room number
        session.streamFile("conference/conf-invalid.wav");
     }
  }
  else
  {
     // User exceeded the confroom prompt limit, say bye-bye
              session.streamFile("conference/conf-goodbye.wav");
              session.hangup();
  }
}
```

See Also

- Javascript
- confcall_js
- Javascript Examples
- Conference_Control