RDI-IT Intuitive & CityVoIP Present

Enabling Callback & Dialout for Comedian Mail

Series: VoIP Engineering *Articles*

Topics: asterisk, voicemail, diaplan

Table of Contents

Introduction

Today's article is for the phone administrator running an Asterisk-based voip server that wants to enhance their voicemail feature set for their users. We assume you are familiar with Asterisk and dialplan basics. With this as the foundation, we show how you can provide the callback and dial out features that are among the most coveted features you can make available in the *Comedian Mail* voicemail system.

So why are users so keen to have access to the callback and dial out features? In short: it allows them to place calls without revealing their phone number to the people they are calling. This is most useful when people are checking their voicemail with a personal cell phone and do not wish to reveal their cell number to the world. Using the voicemail system's callback and dial out features, it is possible to make calls using your personal cell phone while those you are calling see only the caller ID of your company or other affiliated entity.

Let's review the two features we are discussing today:

The **Callback** Feature

When you select the callback option in the advanced menu, you are asking the voicemail system to dial the phone number of the person who left you the current voicemail message. This only makes sense if you are reviewing a voicemail message, so for this reason you only find the callback feature in the menu for *Advanced Options* when listening to messages.

Using the callback feature makes possible two key advantages:

⇒ **Automated Dialing** – The system knows what phone number left the voicemail message, and thus ensures that the number is dialed correctly to reach that person.

⇒ **Caller ID Substitution** – The voicemail system is initiating the call, and is thus able to set the Caller ID number to something other than your own phone number. Typically this is used to call people with your personal cell phone and ensure that the caller ID your callers see is an official company phone number and not your cell phone's number (as would otherwise be the case).

The Dialout Feature

The dialout feature is similar to a callback in that it allows the user to make a phone call using the voicemail system. However, the dialout feature requires the user to specify the number to dial. It is not tied to a particular voicemail message, and thus will appear when enabled in both the top-level Advanced Options menu, as well as the Advanced Options menu when reviewing messages.

The dialout feature is more flexible than the callback because the user is asked to enter the number to dial. This is made even more powerful because you can dial a '*' as part of the number, opening the door for supporting feature codes and other possibilities.

Typically, the system will have you specify a 10-digit phone number to dial, and then make that call with your caller ID set to that of your Company's main phone number, or similar. This is what we will be showing you in this article, as it is the most common implementation and usually what people expect.

However, keep in mind that dialout (and callback too, really) can in fact do just about anything. The dialout feature invokes ***custom*** dialplan code, which opens the door to advanced behaviors limited only by your expertise and imagination. If you wish to, and have the skills, you can instead present the user with a speed dial menu to choose from, or have them say the number to dial by hooking into a speech recognition API – or just about anything else you can dream up.

HowTo: Add Callback and Dialout Features To Your Phone System

Setting aside our dreams of how awesome we might one day make these features – let's start at the beginning. To make these features available, it all comes down to making two key changes to your phone system:

1. **Edit `voicemail.conf`** - Updating the `voicemail.conf` file to set the appropriate configuration values. We go into detail on this aspect in the ***Modifying Voicemail.conf*** section.
2. **Write dialplan handlers** - Writing the custom dialplan code that the voicemail system will invoke on behalf of your users when they select the advanced feature. We cover these edits in the ***Dialplan Edits: Callback Handler*** & ***Dialplan Edits: Dialout Handler*** sections.

We hope you enjoy this, the first of many technical articles to come.

-RDI Intuitive / CityVoIP

Modifying Voicemail.conf

The first task in supporting callbacks and dial out is to set the appropriate options in your `voicemail.conf` file. These options can be specified either in the global settings or on a per-user basis. Figure 1 shows how you would arrange for all users of your voicemail system to be able to use the callback and dialout features, by setting the values in the `[general]` section of the configuration file:

```
#
# This example voicemail.conf snippet shows how you would make callback
# amd dialout available to all users
#
[general]
…
callback = vm_callback
dialout = vm_dialout

[voicemail-group-1]
…
[voicemail-group-2]
…
```

Figure 1: General Settings for callback and dialout.

You can also specify these same options for specific users by setting them in the options section of their `voicemail.conf` entry. Figure 2 shows how this done at the user/extension level:

```
# Voicemail.conf config line for Example x103
# Note the use of callback= and dialout= options for the user
...
[voicemail-example-group]
...
102 => 102,Bob,,,callback=vm_bobsCB
103 => 103,E.g.,exemplar@example.com,,callback=vm_callback|dialout=vm_dialout
104 => 93104,E.G2,,,callback=vm_callback
105 => 105,NoDice,nodice@nocallback.nodialout.com
...
```

Figure 2 : Per-user control of callback and dialout features

Notice that we separate the options with a vertical bar '|' when more than one is specified for a given user/extension. Also, we don't have to give each user the same values. In the dialplan above, user 102 invokes `vm_bobsCB` when they use the callback feature, while user's 103 and 104 both invoke `vm_callback` when they activate that feature.

We also control who has what feature set by using the vertical bar to allow user 103 to have both dialout and callback capabilities. Thus, for the `voicemail-example-group` shown, only user 103 will be able to dial out, while everyone but 105 can use the callback option. Poor 105 has neither feature enabled, and must look on with envy while the other users make callbacks and dial out to their heart's content.

When you have the voicemail.conf file updated and ready to rock-and-roll, be sure to reload the voicemail system inside asterisk via :

```
asterisk -rx "voicemail reload"
```

Dialplan Edits: Callback Handler

Once you have enabled the callback and/or dialout options in the `voicemail.conf` file, your next task is to create the context that those options referenced. To set the stage properly, let us review the user's experience right up to the point where your dialplan context will be invoked:

Pre-requisite: *As mentioned, you need to be reviewing a message for this feature to be available.*

- ⇒ Press Option '3' to select Advanced Options while reviewing a message
- ⇒ Press Option '2' to request a callback for the current message.
- ⇒ The system will then read back to you the phone number for the person who left the message, one digit at a time.
- ⇒ You will then be asked if you wish to call that number or return to the main menu.
- ⇒ Press Option '1' to dial the number
- ⇒ The system then invokes your callback context like so:

Goto(vm_callback, $VMAIL_CALLBACK_NUM, 1)

where `$VMAIL_CALLBACK_NUM` is the caller id number of the voicemail message currently under review.

Code Review: `vm_callback` Context

Figure 3 shows an example of a callback context that does what is expected once the `Goto()` is invoked in the last step above. This code is designed to be placed in your general `extensions.conf` file, or similar equivalent. Let's review the context and make sure everything it does is clear:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Calllback context referenced for the answering service to support callbacks ;;
;; NOTE: Requires that CLIENT CID is set properly so the caller ID is correct. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1  [vm_callback]
2  exten => _1NXXXXXXXXX,1,NoOp(NOTICE: Voicemail Callback invoked for exten
.              ${EXTEN} in '${CONTEXT}' context. Outbound CID: ${CLIENT_CID}.)
3    same => n,Set(CALLERID(num)=${CLIENT_CID});
4    same => n,Dial(SIP/${EXTEN:1}@${OUTBOUND_GW}
5    same => n,Hangup();
6
7  exten => _NXXXXXXXXX,1,NoOp(NOTICE: VOicemail Callback invoked for exten
.              ${EXTEN} in '${CONTEXT}' context.)
8    same => n,Goto(vm_callback,1${EXTEN},1);
9    same => n,Hangup();
10
11  exten => i,1,NoOp(All other callbacks we will reject - this is only for
.              external numbers)
12    same => n,Playback(unidentified-no-callback);
13    same => n,Hangup()
```

Figure 3 : Simple vm_callback Example

⇒ **Line 1** : **Declaration of vm_callback context**.

⇒ **Line 2**: **Main number handler.** We use pattern matching to catch any 10 digit number that is dialed with a 1 prefix. Since we are pattern matching, we begin the extension with _. Then, we specify the 1 that we expect to see no matter what – followed by an 'N' to catch anything from 2-9. This is because area codes in the US/Canada do not start with 0 or 1. After the 'N' we have 9 more 'X' digits – to catch whatever the phone number might be.

⇒ **Line 3**: **Set Outbound CallerID number.** This is the magic line that makes this feature so attractive. We set the outbound caller ID to be the value of the variable "CLIENT_CID", allowing the caller to hide their own personal phone number and instead use this as their caller ID. If CLIENT_CID is set to the main number for your user's company, for example, this will allow them to make voicemail callbacks with their caller ID appearing as if the call originates from the company's phone system – regardless of the true phone that is being used.

⇒ **Line 4**: **Dial Command**. This is tailored to our particular provider, which requires outbound calls to have a starting '`1`' digit for the US & Canada, followed by the ten digit number to dial. We assume SIP is being used here – but the call could as easily be made with PJSIP as the default. The number is then followed by an '`@`' sign and then the trunk we are using to handle the outbound call. In this code, we assume you have this defined in a variable named `OUTBOUND_GW`.

- To make sure you get this right for your system, we recommend that you examine a `Dial()` command in one of your other dialplan files to determine the proper format to use for the command here.

⇒ **Line 5**: **Hangup** . There is no returning back to the voicemail system once the call is complete, so `Hangup()` is appropriate here.

Alternate Handler for 10 Digits Only:

⇒ **Line 7** : We add a handler for numbers that do not start with a 1 already when the vm_callback is invoked. It is the same as the main handler, but with the 1 removed. To promote code re-use, we simply invoke the _1NXXXXXXXXX handler by issuing:

```
Goto(vm_callback,1${EXTEN},1)
```

on **Line 8**.

Invalid Extension Handler

⇒ **Line 11**: We set up an invalid extension handler here to provide a short message informing the user that the callback number they wanted us to dial for them is not supported. You would probably want something better than the Playback message used on **Line 12** for a production environment. The primary purpose of the invalid handler is to catch the case where the voicemail caller ID is that of a local extension. Our handler doesn't support callbacks for these types of messages, and the message we play for the user here is intended to let them know that.

Why don't we support local extension callbacks? See the note below:

Note on Local Extension Callbacks

⇒ In the interest of space and clarity, we did not include in our `vm_callback` handler matches for local extensions. Part of the reason is that we don't know what extension range your local system might be using, nor what mechanism you might use to dial them. For example, RDI CityVoIP™ uses `PJSIP_DIAL_CONTACTS` to dial our extensions – allowing for multiple devices to be associated with a single extension handler. Trying to cover these various possibilities would not be clear and concise code, and rather than complicate what is otherwise a straightforward handler, we chose to leave this aspect as an *exercise for the reader.* ☺

Dialplan Edits: Dialout Handler

The dial out handler can be the same as the callback handler, or it can be a separate context. The primary difference between the callback and the dial out is that the user is prompted by *Comedian Mail* to enter the number they wish to dial, followed by the # key. This is the number that is then passed on to the handler, invoking the equivalent of:

⇒ `Goto(vm_dialout,${NUMBER_USER_ENTERED},1)`

This actually opens the door for a lot of flexibility – depending on how much effort you wish to put into the feature. We will discuss a few possibilities in a moment, but first let us review the example dialout handler that provides basic dialout support, shown in Figure 4:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Dialout context referenced for the answering service to support dial out    ;;
;; NOTE: Requires that CLIENT_CID is set properly so the caller ID is correct. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[vm_dialout]
exten => _1NXXXXXXXXX,1,NoOp(NOTICE: Voicemail Dialout invoked for exten
            ${EXTEN} in '${CONTEXT}' context. Outbound CID: ${CLIENT_CID}.)
  same => n,Set(CALLERID(num)=${CLIENT_CID});
  same => n,Dial(SIP/${EXTEN:1}@${OUTBOUND_GW}
  same => n,Hangup();

exten => _NXXXXXXXXX,1,NoOp(NOTICE: VOicemail Callback invoked for exten
            ${EXTEN} in '${CONTEXT}' context.)
  same => n,Goto(vm_callback,1${EXTEN},1);
  same => n,Hangup();

 exten => i,1,NoOp(All other callbacks we will reject - this is only for
            external numbers)
   same => n,Playback(CanOnlyDial10DigitNumbers)
   same => n,Hangup();
```

Figure 4 – Example Dialout Context Handler

This handler is essentially the same as the callback handler, so we won't go over it in detail like we did the callback version. Just keep in mind the major points from that previous review:

⇒ The handler only accepts 10 digit phone numbers, as well as a ten-digit phone number prefixed with a '1'.
⇒ CallerID is set to match the value of the dialplan variable `CLIENT_CID` – which must be set prior to the voicemail system being invoked by the caller.
⇒ Outbound calls are presumed to be SIP/ based, but could also be PJSIP with minor modification. The gateway to send the call out on is stored in the dialplan variable `OUTBOUND_GW` for our example.

The one part of our handler for dialout that is different from the callback handler is the invalid extension. Here we cover all non-10-digit numbers with a message that is customized for dial out. Specifically:

⇒ **Line 12** is where this announcement is made to the user. The `CanOnlyDial10DigitNumber` (highlighted in yellow) is presumed to be a pre-recorded sound file with that file name that effectively says:

> *"We're sorry, this feature only supports dialing out for 10 digit telephone numbers. Please try again using a phone number with area code"*

Supporting Feature Codes in the Dialout Handler

We want to offer a few ideas for how to improve the basic handler shown here. The dialout feature is the one that offers the most flexibility due to the user being able to enter the number to call back. Keep in mind that the user can dial the '*' (asterisk) character as part of the number they enter as long as it is not the first digit (because *Comedian Mail* uses '*' to return to the previous menu' (it just can't be the first digit) – which opens the door for a feature-code system if you wish to add it.

What do we mean by "a feature-code system"? Ah, I'm glad you asked. ☺

Simple Feature Code Example:

To give you an idea of what we mean when we say "feature-code system", imagine the following system for allowing the user to specify what caller ID number they want to use for the outgoing call:

⇒ System supports a prefix of 1* or 2*, followed by a ten digit number to dial, or dialing the number without a prefix.
  - Numbers dialed with the 1* prefix will use CallerID number (A)
  - Numbers dialed with the 2* prefix will use CallerID number (B)
  - Numbers dialed without a prefix will not set the caller ID, and will instead let the user's normal Caller ID be used for the outgoing call. (meaning that call will not hide their caller ID by using a substitute).

Using this simple system, the user can control what caller ID is presented to those they contact without needing to do more than dial two extra digits before the number.

This is only the start of what can be made available through a feature-code system. What else is possible? Well, let us see…

Other Feature Code Suggestions

The feature code example shown above could be used to trigger other features, such as **5*** to trigger call recording, or **8*** to play a pre-recorded message. The system can be made to support multiple feature-codes in a single call by simply chaining them together on the front of the call. Figure 5 shows a simple three-code system that does this:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Feature Code Chaining : Example with 1*, 2*, and 5* options
;; Invoked feature codes are tracked by setting the dialplan variables as each ;;
;; feature code is parsed:
;;
;;      CID_NUM     - tracks the caller ID number to use for the call
;;                    (if not set, call uses normal caller ID value)
;;      RECORD_CALL - If set, the call should be recorded when dialed
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[vm_dialout_extended]
exten => _1NXXXXXXXXX,1,NoOp(NOTICE: Voicemail Dialout invoked for exten
.              ${EXTEN} in '${CONTEXT}' context. Outbound CID: ${CLIENT_CID}.)
   same => n,ExecIf($["${CID_NUM}" != ""]?Set(CALLERID(num)=${CID_NUM}));
   same => n,ExecIf($["${RECORD_CALL}" = "1"]?Gosub(RecordCall,s,1))
   same => n,Dial(SIP/${EXTEN:1}@${OUTBOUND_GW}

exten => _NXXXXXXXXX,1,NoOp(NOTICE: VOicemail Callback invoked for exten
.              ${EXTEN} in '${CONTEXT}' context.)
   same => n,Goto(vm_callback,1${EXTEN},1);
   same => n,Hangup();

exten => _1*.,1,NoOp( Caller ID #1 Prefix Match )
   same => n,Set(__CID_NUM=6145551212)
   same => n,Goto(vm_dialout_extended,${EXTEN:2},1)    ; Continue Processing

exten => _2*.,1,NoOp( Caller ID #2 Prefix Match )
   same => n,Set(__CID_NUM=6145553434)
   same => n,Goto(vm_dialout_extended,${EXTEN:2},1)     ; Continue Processing

exten => _5*.,1,NoOp( Call Recording Request )
   same => n,Set(__RECORD_CALL=1)
   same => n,Goto(vm_dialout_extended,${EXTEN:2},1)    ; Continue Processing

exten => i,1,NoOp(All other callbacks we will reject - this is only for
.              external numbers)
    same => n,Playback(CanOnlyDial10DigitNumbers)
    same => n,Hangup();
```

Figure 5 : Feature Code Chaining System For Dial Out

Using wild-card matches allows "chaining" to occur by matching the first 2 digits that make up the feature code, setting a state variable for later recall, and then calling the handler recursively *with the remainder of the number* to process any further prefix codes. When the number is reduced to the bare 10-digit number, or '1' followed by ten digits, then the standard handler for dialing out is invoked.

The standard handler for dialing out is modified to check the state variables to add the special functionality as requested. Specifically:

- ⇒ **Line 3** - checks the `CID_NUM` variable. If this variable has been set to a value, then that value is used as the caller ID for the outbound call.
- ⇒ **Line 4** checks the `RECORD_CALL` state variable and, if set, invokes the routine "`RecordCall`", which is presumed to set up the call recording appropriately.
- ⇒ **Best Practices** - Note that these two lines are streamlined without error checking and best practices for coding your dialplan. In particular, production code should check the value of the `CID_NUM` variable to ensure it is correct. Further, the dialplan as a whole should have a match for invalid feature codes to give a good error message to the user.

Feature Code Chaining: An Example Call

Chaining allows the user to invoke multiple feature codes in a single dialout request using the method above. For example, the user could dial **1*5*6145554321** as the dialout number, and the system woud then:

- ⇒ ***[dialout handler] invoked with 1*5*6145554321***
- ⇒ Match 1* and set the `CID_NUM` variable to 6145551212
- ⇒ Strip off the 1* and invoke the handler with what remains: **5*6145554321**
  - ***[dialout_handler] invoked with 5*6145554321***
  - Match the 5* and set the `RECORD_CALL` variable to 1
  - Strip off the 5* and invoke the handler with what remains: **6145554321**
    - ***[dialout_handler] invoked with 6145554321***
    - Sets the `CALLERID(num)` to `CID_NUM`
    - Checks `RECORD_CALL` and sees it set, so calls `RecordCall()` via Gosub.
    - Dials outbound call to 6145554321

Using an IVR and other Ideas

As an alternative to feature codes, you could accept a standard number to dial and then provide the user with an IVR to select the features they want for the call. Use of a menu instead of feature codes has two advantages for the user to make up for the drawback of increased complexity:

- ⇒ **Works for callback users too** – because a menu does not rely on the user being able to dial the number ahead of time, it can be inserted into the callback handler to provide the features to those users as well as your dialout users.
- ⇒ **Self-documenting** – The message describing the menu allows you to inform your users as to what features are available and how to invoke them. Embedded feature codes are great, but they do rely on the user knowing of them ahead of time.

The following list is a starting point for considering what features you might want to provide to your users over time as you "flesh out" your callback and dialout handlers. Taking the time to add support for a feature in this list will often benefit your users across the board – thus justifying the time spent adding them.

- ⇒ **Functionality Ideas**
  - Allow user to select the desired caller ID from a preset list of valid numbers.
  - Allow the use to turn on call recording

- Bridge the call with one or more users
- Bring in a listener for a whisper session so they can observe the call and provide help if needed.
- Play a standard disclaimer before the call begins
- Allow the number to be blocked on your system going forward
- Opt to end the call with a survey or similar mechanism.

⇒ **System Enhancements**

- Announce info about the area code and/or NXX.
- Inform the caller the last time they called the number, assuming you have that information stored in a database that can be easily queried, etc..

Security Measures

We want to close out this article with a brief discussion about security. As everyone knows, the world is becoming more dangerous every day – and hackers are continually seeking ways to break into phone systems in order to make calls to high-toll phone numbers and instigate other nefarious activities.

The features we describe in this article *represent a potential security hole* in your phone system, as they offer the ability to make phone calls through the voicemail system alone. All it takes is someone with a weak passcode having an intruder guess their voicemail credentials. Once they have done this for an account where the callback or dialout options are available, they can make an outbound, arbitrary phone call.

For the dialout feature, this is as simple as dialing the number desired. The callback feature is also abusable in a similar way, with just slightly more setup work:

1. Hacker arranges to call in with their caller ID set to the number they wish to be able to call out to.
2. They then leave voicemail message for the voicemail account they have compromised.
3. Call in to the voicemail system, navigate to this message, and invoke the callback handler
4. Repeat Step 3 as many times as the system will allow.

We recommend the following security measures be a part of any use of the dialout or callback features in a production phone system:

⇒ Strong voicemail passwords with 5 digits or more – no patterns or
⇒ Limit access to callback & dialout via 7-10 digit PIN
⇒ Restrict access to internal phones if possible
⇒ Limit outbound calls by area code – the FTC provided a list back in 2014 that is still a good starting point for a ban list:

- https://www.consumer.ftc.gov/blog/2014/02/one-ring-cell-phone-scam-can-ding-your-wallet

⇒ Limit maximum number of dialouts or callbacks allowed in a given time period. The idea is that even if hacked, the evil-doers wont' be able to generate more than X calls per hour using the features.

⇒ Ensure the voicemail system has `maxlogins` set to a value of 3 or less in `voicemail.conf`.

Implementing these added security measures takes time and effort, and may make the system slightly harder to user for your phone customers. However, *and we cannot stress this enough*, in today's inter-connected world you cannot afford the risks of providing callback and dialout features to your users without having as many of these policies in place as possible.

Look for future articles in this tech series to cover, in further detail, some of the security measures mentioned above. Most are in use here at RDI CityVoIP and form a layer of protection within an overall security strategy we have in place, seeking to keep the phone system and its users as free from abuse as possible.