Mitel MiContact Center Enterprise

SIMPLE VOICE MAIL APPLICATIONS SCRIPT MANAGER USER GUIDE

Release 9.5

NOTICE

The information contained in this document is believed to be accurate in all respects but is not warranted by Mitel Networks™ Corporation (MITEL®). The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes.

TRADEMARKS

Simple Voice Mail Applications Script Manager
User Guide
Release 9.5 – September 2020

INTRODUCTION

This document contains the sample scripts using components provided in the Call and Media Control and the Database Component Libraries. The tutorial provides a set of sample scripts and useful hints when building the script.

WHAT YOU WILL LEARN

In this document, the following topics are discussed in detail:

- How to manage the recorded voice files.
- Sample voice recording application.
- Sample voice mail application.

BEFORE USING TUTORIALS

Make sure the sample scripts are installed on your development PC. Refer to Installing Sample Scripts.

Script Manager Installation installs a database called smdb to help maintain the recorded messages. The tables are used when new messages are recorded using application messages, and when recorded messages are deleted using DeleteRecMsg block. Script Manager does not automatically remove any entry in the table. The user must write a script to manage the table and the voice files. The following tutorials provided are sample scripts to maintain the voice messages, as well as clean up old messages in the database table and from the OAS Server. The smdb database has two tables:

APP_SRV_INFO

App_srv_info consists of two fields. One entry is automatically created during installation. The default application code is “DEF” and the PIN is 1234. This is to store the application code and the associated PIN for message retrieval. Application users can create new entries to the table for each application.

FIELD NAME	DATA TYPE	USAGE
code	Varchar	Stores the application code.
Pin	Int	Password to access the application messages.

APP_SRV_MSG

One row will be created for each recorded message that records as Application Message. App_srv_msg has the following fields:

FIELD NAME	DATA TYPE	USAGE
Message_id	Int	System generated unique message identifier
Media_object_id	Varchar	File name consists of the recorded voice message.
Message_status	Varchar	The status of the recorded voice message. “N” new message “R” Read message “D” Discarded message “A” recorded message has been deleted and this entry is available for re-use.
Num_of_read	int	Number of read on this recorded message.
Srv_access_num	Varchar	The service access ID used to record the message.
App_srv_code	Varchar	The application code used during the recording. The default is “DEF”
Message_timestamp	Datetime	The timestamp when the recorded file was last accessed.
Firstread_timestamp	Datetime	The timestamp when the first time the recorded file was read.
Media_repository	varchar	The name of the media repository.

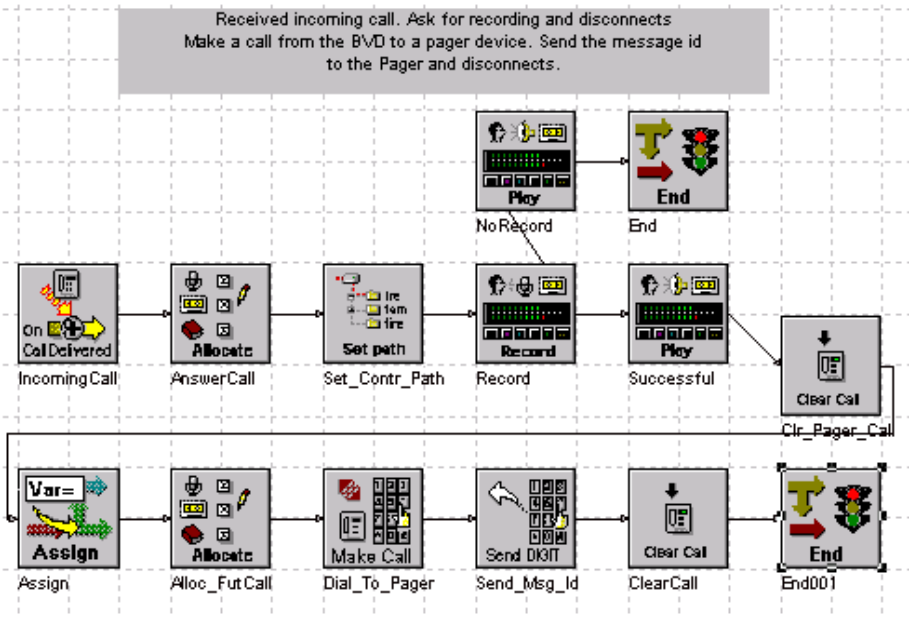
TUTORIALS

RECORD MESSAGES

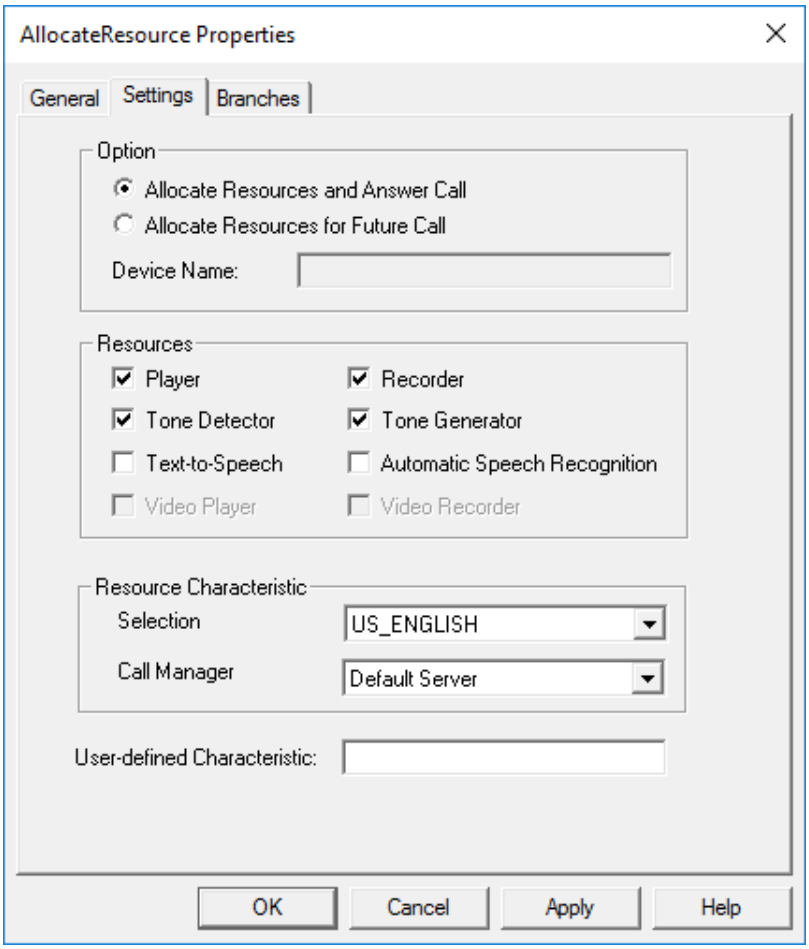
RECORD.MFD

The record messages tutorial demonstrates the usage of recording and managing the recorded messages. The Record script allows the caller to record a message and then pages an agent given the message identifier. This sample script demonstrates the use of the Allocate Resource, SetDefaultContainerPath, Record, MakeCall, Assign, SendDigits and ClearCall components.

From Script Designer, open the project called tutorials.fdp from the SampleScripts directory. Open the **Record** script.

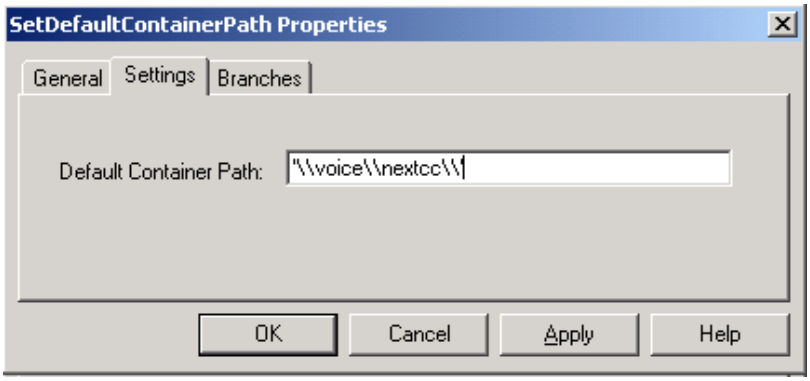


ALLOCATERESOURCE



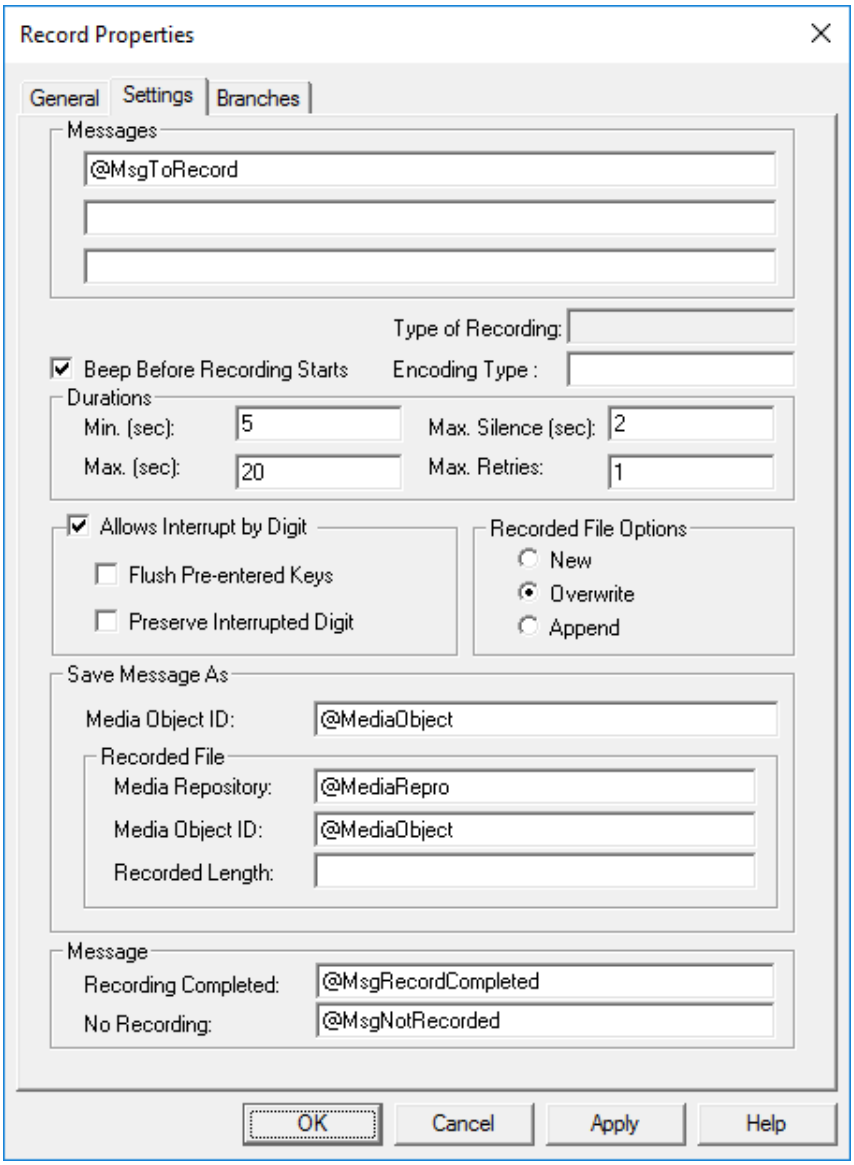
The AllocateResource block allocates resources for play messages and recording.

SET_CONTR_PATH



The SetDefaultContainerPath block sets the directory path where the play messages and the recorded voice prompts will be located.

RECORD



The Record block records a voice message from the caller. The messages provided in the Messages text boxes will be played before recording starts. The “Beep Before Recording Starts” option should be enabled if the caller is to hear a beep before recording starts.

The Durations options define the recording parameters. The minimum and the maximum duration of the recording can be set. If less than the minimum duration was recorded, the recorded voice file will not be created.

The Maximum silence indicates there is no speech for recording. If the system detects that the silence period is longer than the maximum silence period, the recording will be stopped.

When saving the recorded message, there are two options. Application Message uses the Script Manager pre-defined database to maintain the recorded file. An entry will be added to the app_srv_msg table when a successful recording is completed. A message ID is returned to the

script to identify the added entry. The default application code is “DEF”. The user can change the application to a different one depending on the usage. The application code can be used to determine who has access to which recorded message.

If the recording is saved as an individual message, the script must manually maintain the recorded file with its own database. There will not be any entry saved in the app_srv_msg. The recorded file name will be provided in the media object ID.

There are also options to overwrite or create new recorded files. If user interrupt is allowed, then the option “Allow Interrupt by Digit” should be set.

ASSIGN

The Assign block converts the message ID from an integer to a string. The message ID will be used to send to a pager of a designated number.

ALLOC_FUTR_CALL

After a successful recording, the call is disconnected. The script continues and will make an outgoing call to a pager. This is to notify the called party of the arrival of a voice message. The AllocateResource block allocates the resources for a future call. The tone generator resource is also required to generate tones to the pager.

MAKECALL

The MakeCall block makes an outgoing call to a paging device. It waits for the answer before continuing to the next block.

SEND_MSG_ID

The SendDigit block sends the recorded message ID in a string format. This message ID will be used later on to call back to the system to listen to the recorded message. Once the paging is successful, the call is cleared from the system.

VOICE PROMPTS

The following voice prompts need to be recorded and configured as play messages in the OAS Server. The default message IDs are provided in the following table. If a different message ID is used in the OAS Server, re-configure the variables with the new message ID when creating the Service Access.

SCRIPT MANAGER VARIABLE	MESSAGE ID	SAMPLE VOICE PROMPT DESCRIPTION
MsgNotRecorded	1100	“Recording failed”
MsgRecordCompleted	1101	“Recording completed”
MsgThkGoodbye	1102	“Goodbye”

MsgToRecord	1103	“Start recording after the beep”

TESTING THE APPLICATION

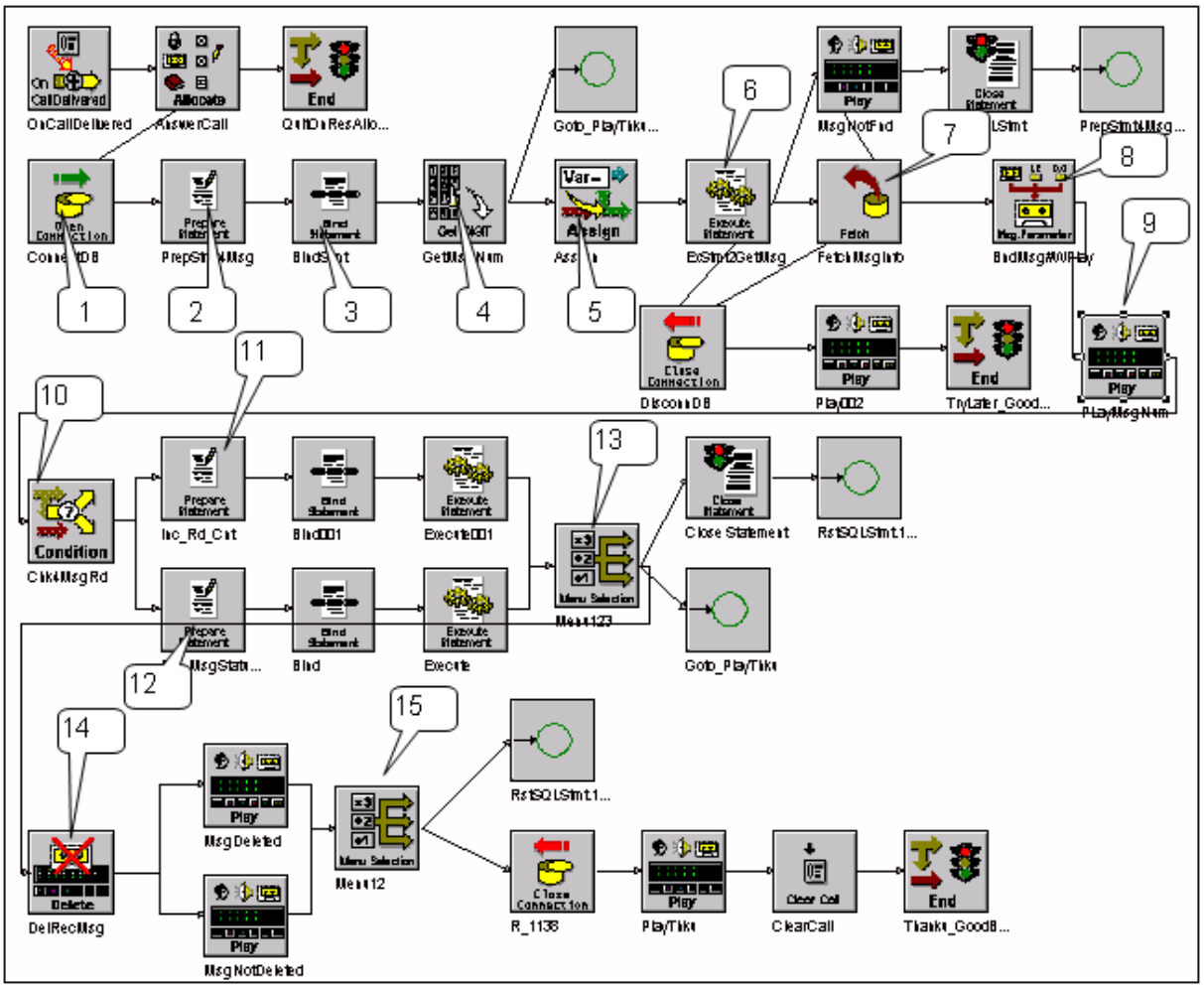
Compile and create a Service Access using the compiled script. Make sure the variables, voice prompts and play messages are installed and configured.

LISTEN TO RECORDED MESSAGES

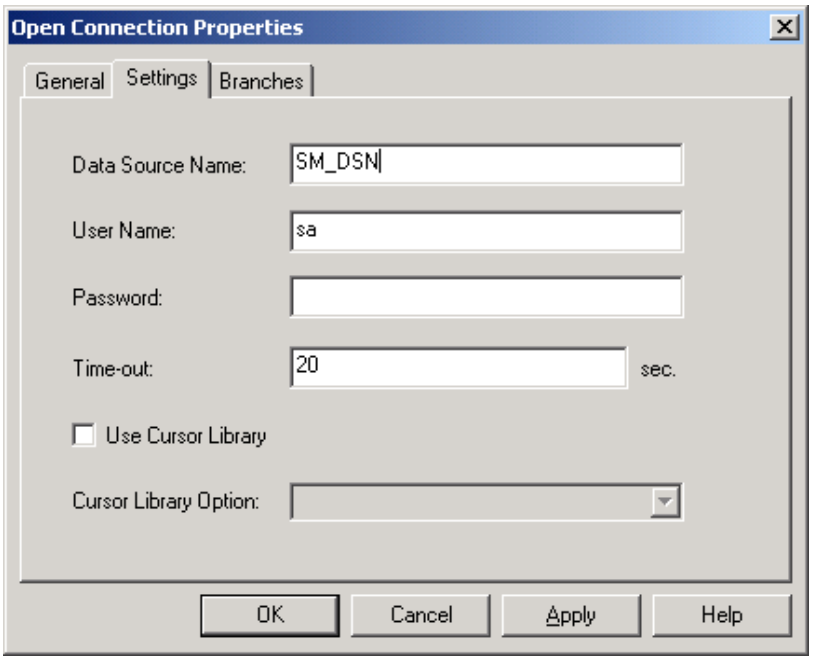
LISTEN.MFD

The scripts designed to listen to the recorded messages continue from the previous Record script. After the user has recorded the voice message, the script notifies the user to call back to the system to listen to the specified message.

The Listen script allows the caller to listen to a message based on a message ID. This sample script demonstrates the use of ODBC components as well as the DeleteRecMsg component.

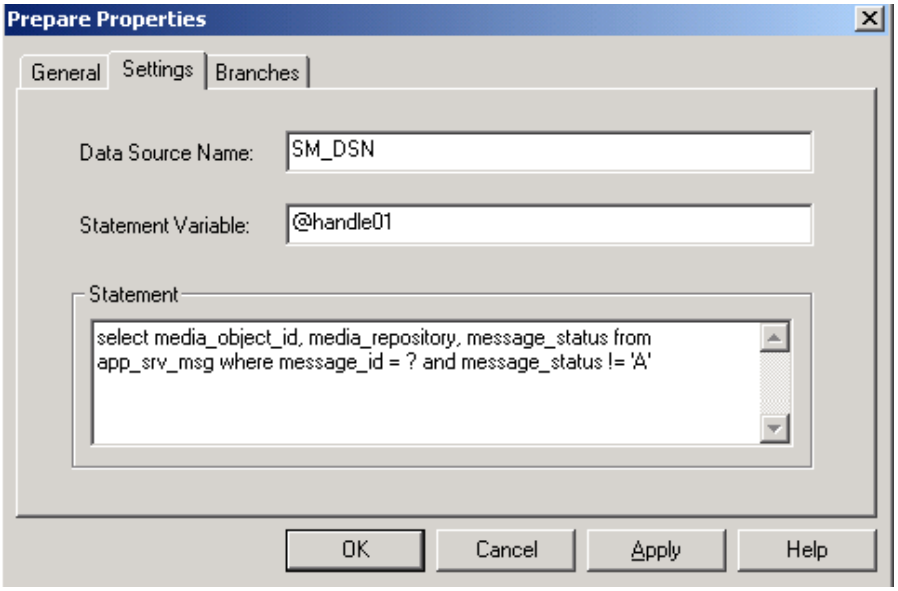


OpenConnection(1)



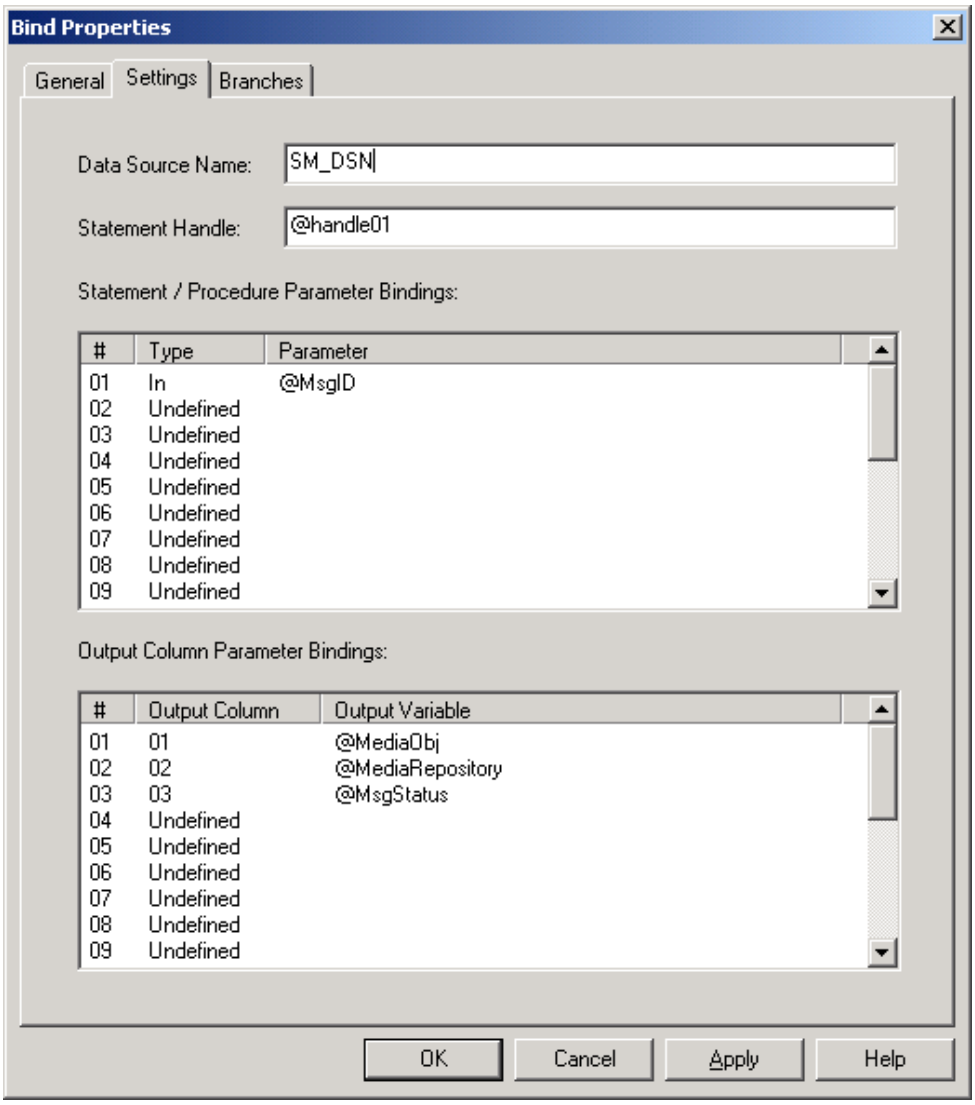
After receiving an incoming call and allocating resources for play messages, the OpenConnection block connects to the **smdb** database. The data source called SM_DSN was created when Script Manager was installed.

Prepare (2)



The Prepare block prepares the SQL statement to get the requested message ID. If the message status is deleted (equal to 'A'), the message ID will not be returned.

Bind(3)



The Bind block binds the input to the @MsgID variable. This @MsgID content will be obtained from the user's DTMF input. It also binds the output to the variables @MediaObj, @MediaRepository and @MsgStatus. This is to obtain the specific voice message that will be requested from the caller.

GetDigit (4)

The GetDigit block prompts the user to enter the message ID that was received from the pager (Refer to the Record.mfd sample script). The DTMF digits are then saved into the @MsgID variable.

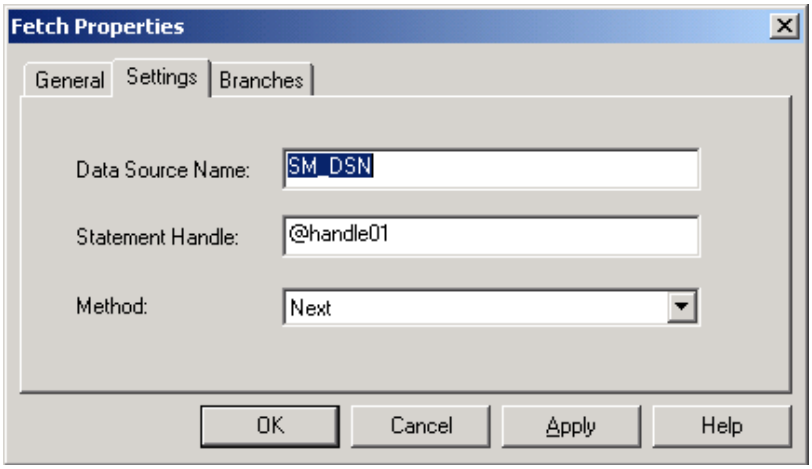
Assign (5)

The Assign block converts the string @MsgID to an integer and stores it to @SavedMsgID.

Execute (6)

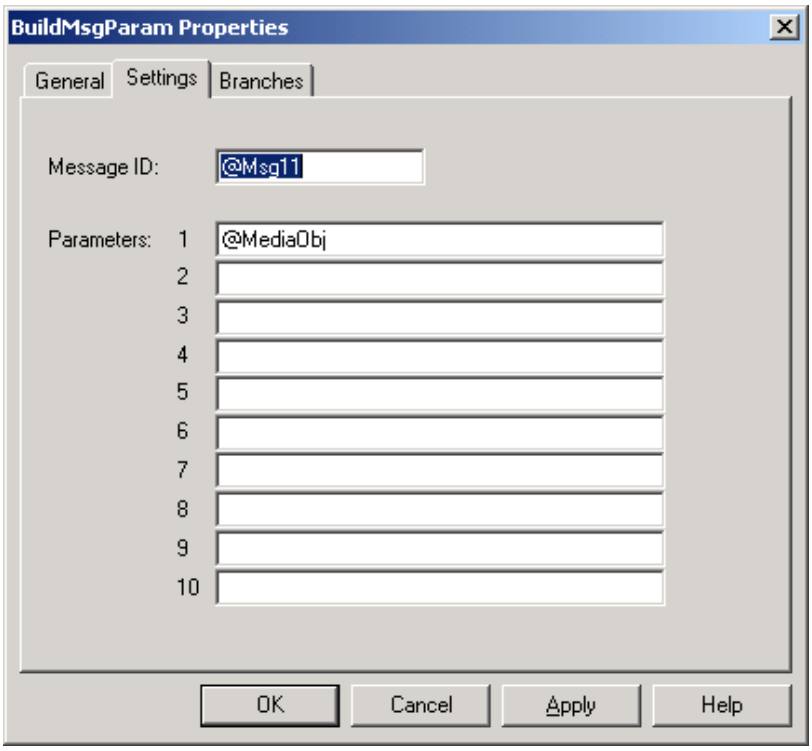
The Execute block executes the previous prepared statement with the handle @handle01.

Fetch (7)



The Fetch block is executed on the successful branch from the Execute block. This fetches the data from the select statement. If the Fetch block takes the success branch, the variables @MediaObj, @MediaRepository and @MsgStatus will contain the recorded voice file information.

BuildMsgParam (8)

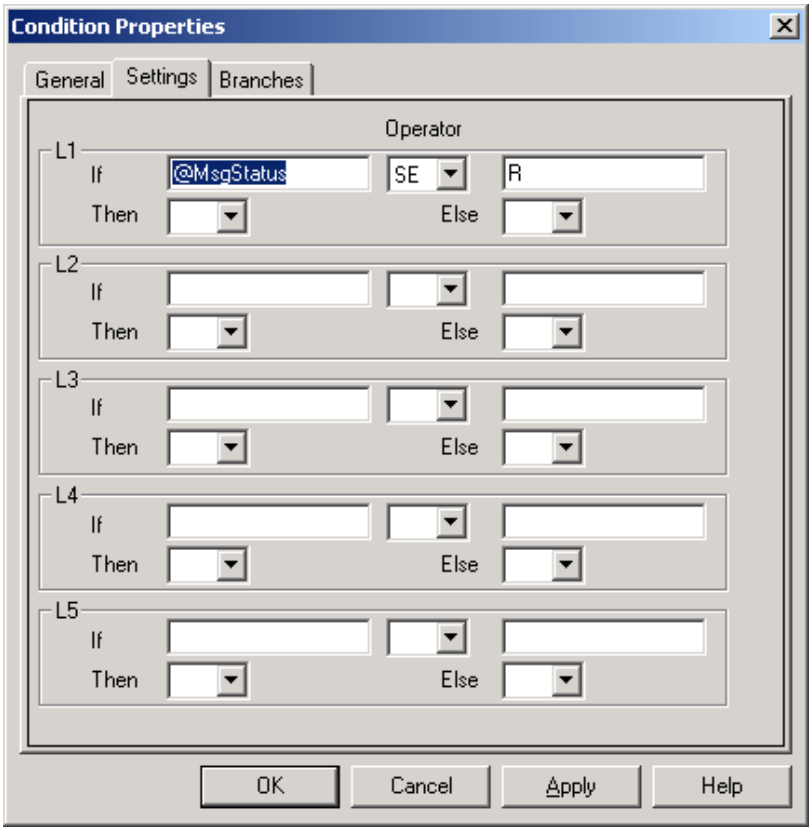


To play back the recorded message, the BuildMsgParam block uses the @MediaObj as the input parameter to the play message ID defined in the @Msg11 variable. Refer to the **Play Message IDs for Listen and ListenAll Scripts** section for the definition of the play message.

Play (9)

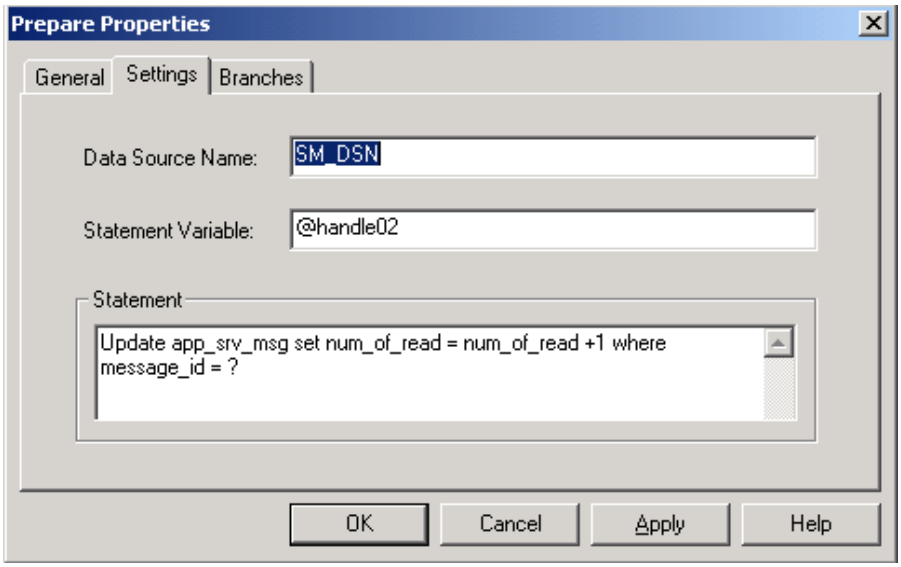
The Play block here does not provide a play message ID. It plays what has been built before it reaches this block. In this example, it was built from the BuildMsgParameter (8) block. The interrupt option allows the user to interrupt the play.

Condition(10)



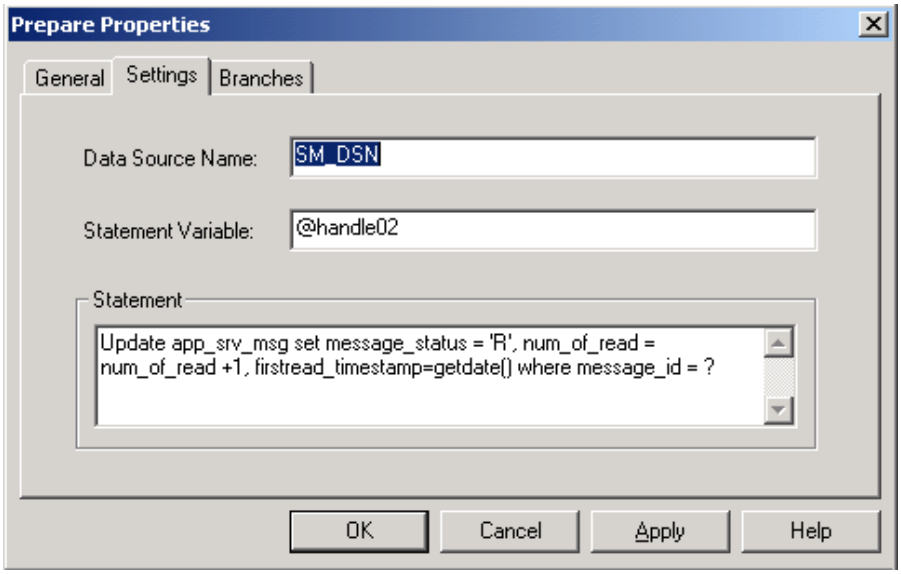
After the Play is completed, it goes to the Condition block. The Condition block checks if the voice message has a status of “R” (read). It does a string comparison to the character R. If the message was listened to previously, it will increment the read counter (11). Otherwise, it will update the message status to “R” and set the read counter to 1 (12).

Prepare (11)



In the case where the voice message was already listened to at least once, this Prepare block increments the read counter by 1. It is followed by the Bind block with the @MsgID as the input parameter, and then the Execute block.

Prepare (12)

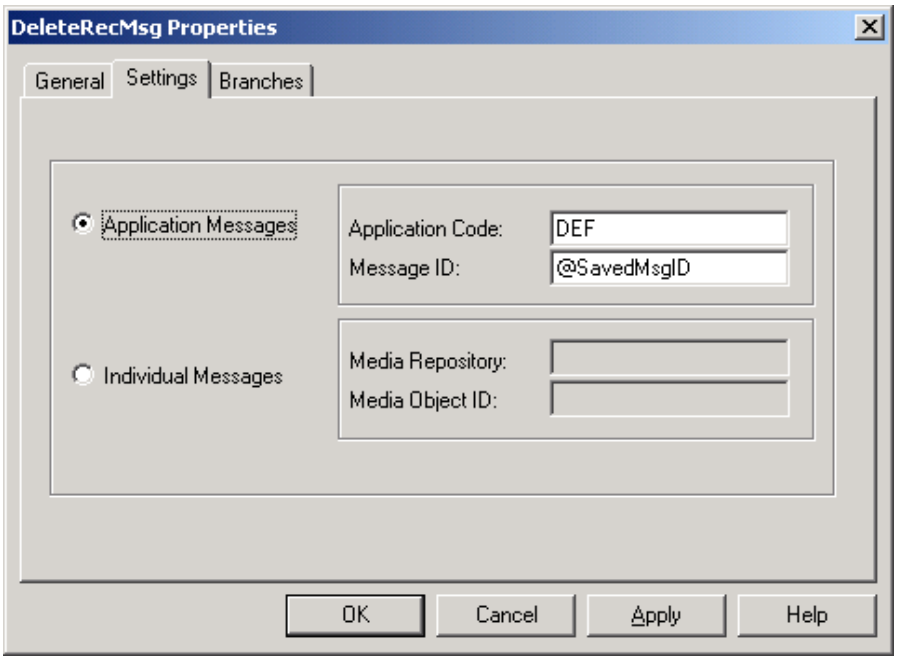


If this is a new message, the message status is updated to be ‘R’. The read counter is incremented, and the time stamp for this read is set. It is followed by the Bind block with the @MsgID as the input parameter, and then the Execute block.

MenuSelection (13)

After the caller has listened to the voice message and the database table has been updated, the user is given a choice to listen to a new message, delete the current message or exit the system. The @Msg12 has the play message ID containing the prompts. See **Message IDs for Listen and ListenAll scripts** section for the instruction to construct the play message. If the user selects the digit to delete the message, the script continues on the DeleteRecMsg block.

DeleteRecMsg (14)



The DeleteRecMsg block removes the @MediaObject file from the OAS container. If Application Message was used during recording, the same option must be used in the DeleteRecMsg block. In this example, the same application code (DEF) and the message ID are used in the block. It will delete the voice file from the OAS container. Then it will mark the message status to 'A' for the entry that matches with the @SavedMsgID in the app_srv_msg table. This enables the next record application message to re-use the table entry.

MenuSelection (15)

After the voice message is removed, a confirmation message will play. It prompts the user to choose either play another message or exit the system.

TESTING THE APPLICATION

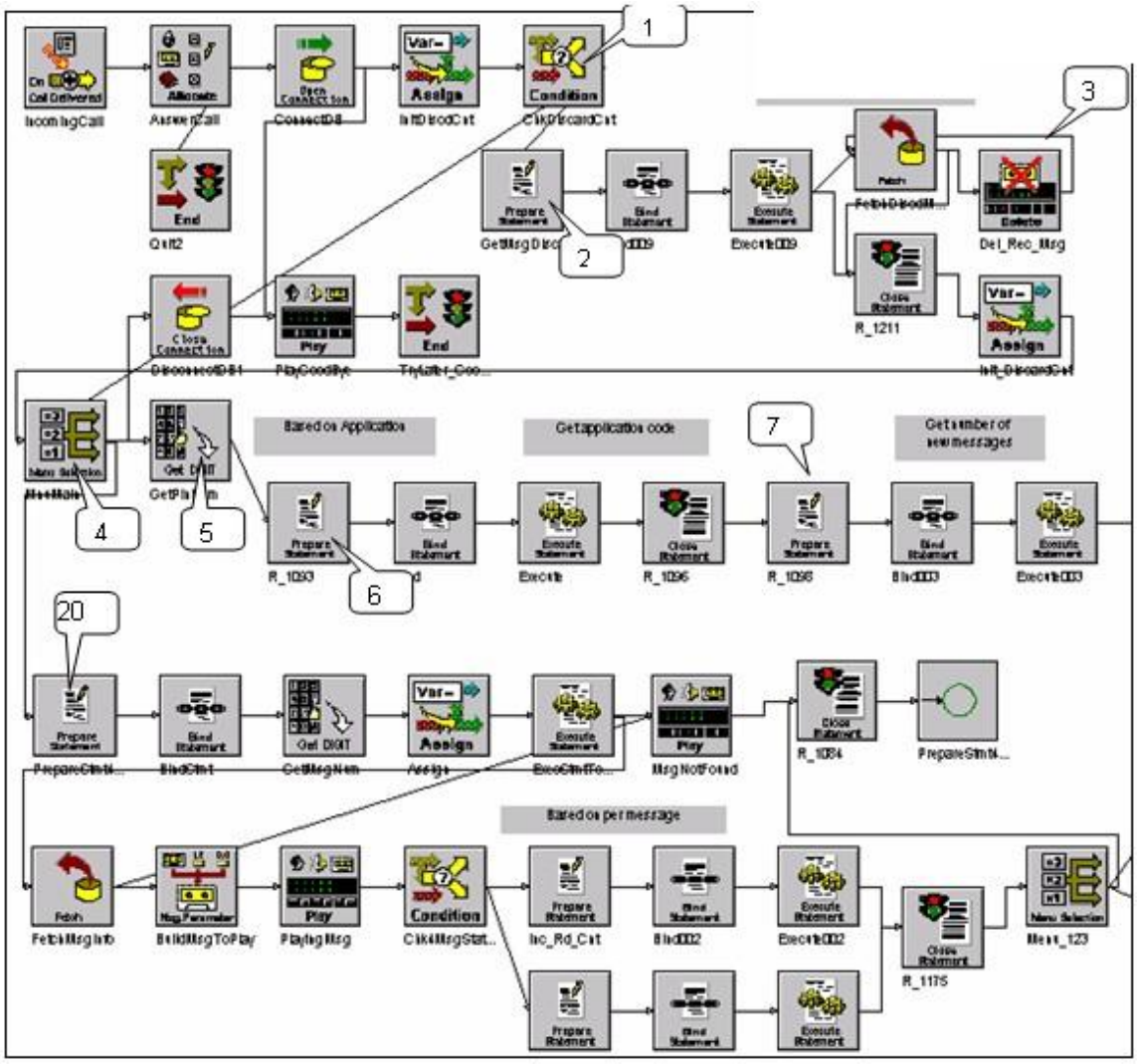
After all the necessary changes in the script, compile the script and make sure a binary script is generated. Create a Service Access using the binary script. Make sure the play messages are configured in OAS as indicated at the end of this document. Modify the play message numbers if they are configured with a different value. Configure the MonitorDev and you are ready to test the script.

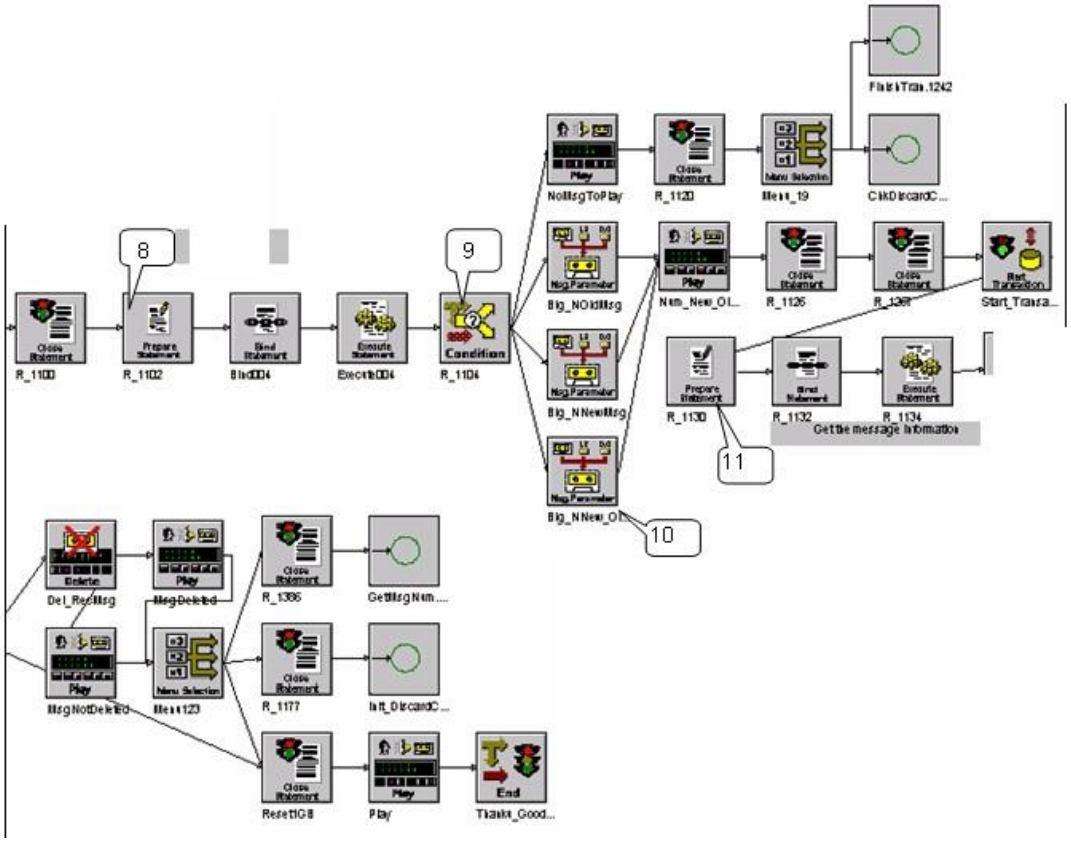
LISTEN TO ALL THE RECORDED MESSAGES

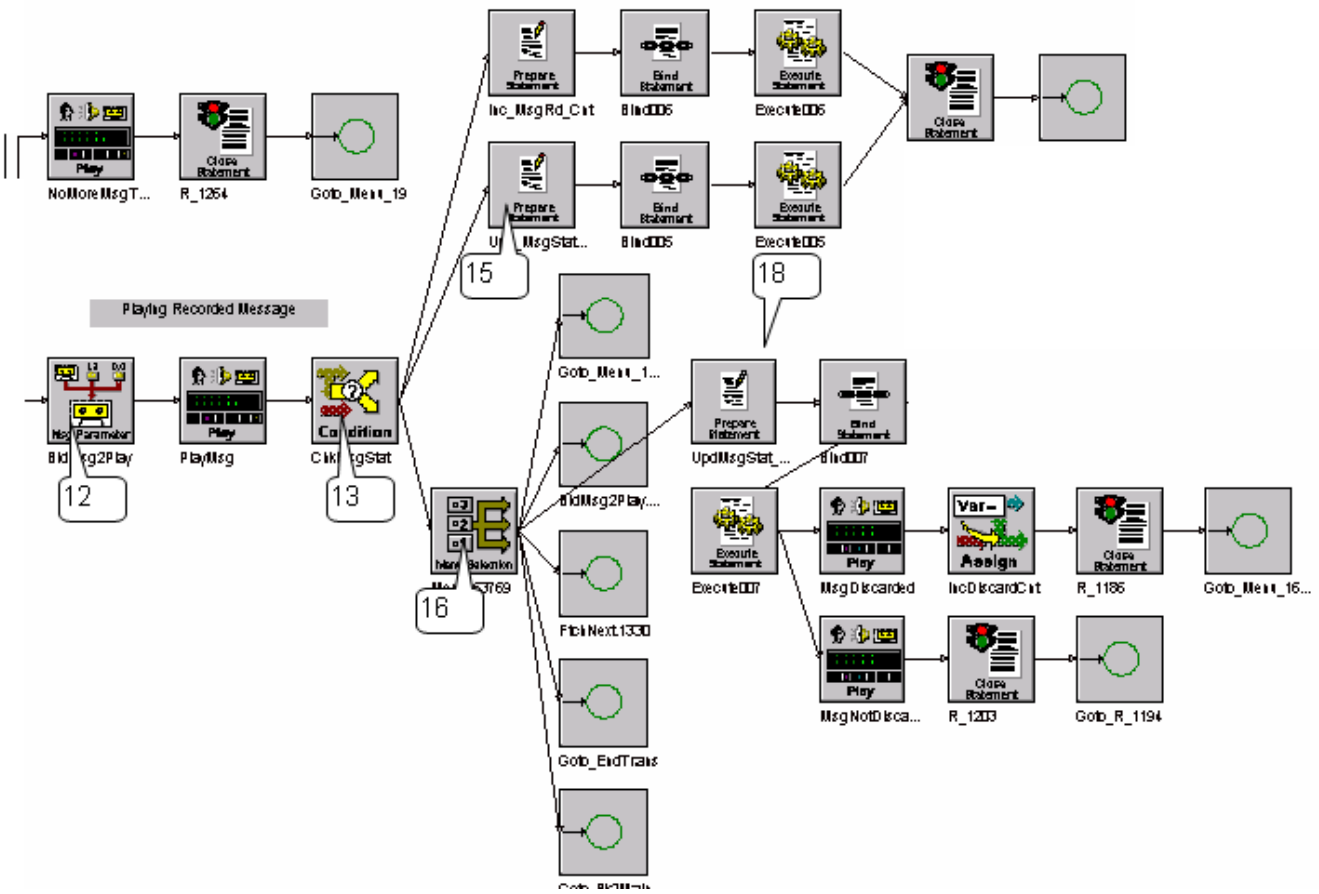
LISTENALL.MFD

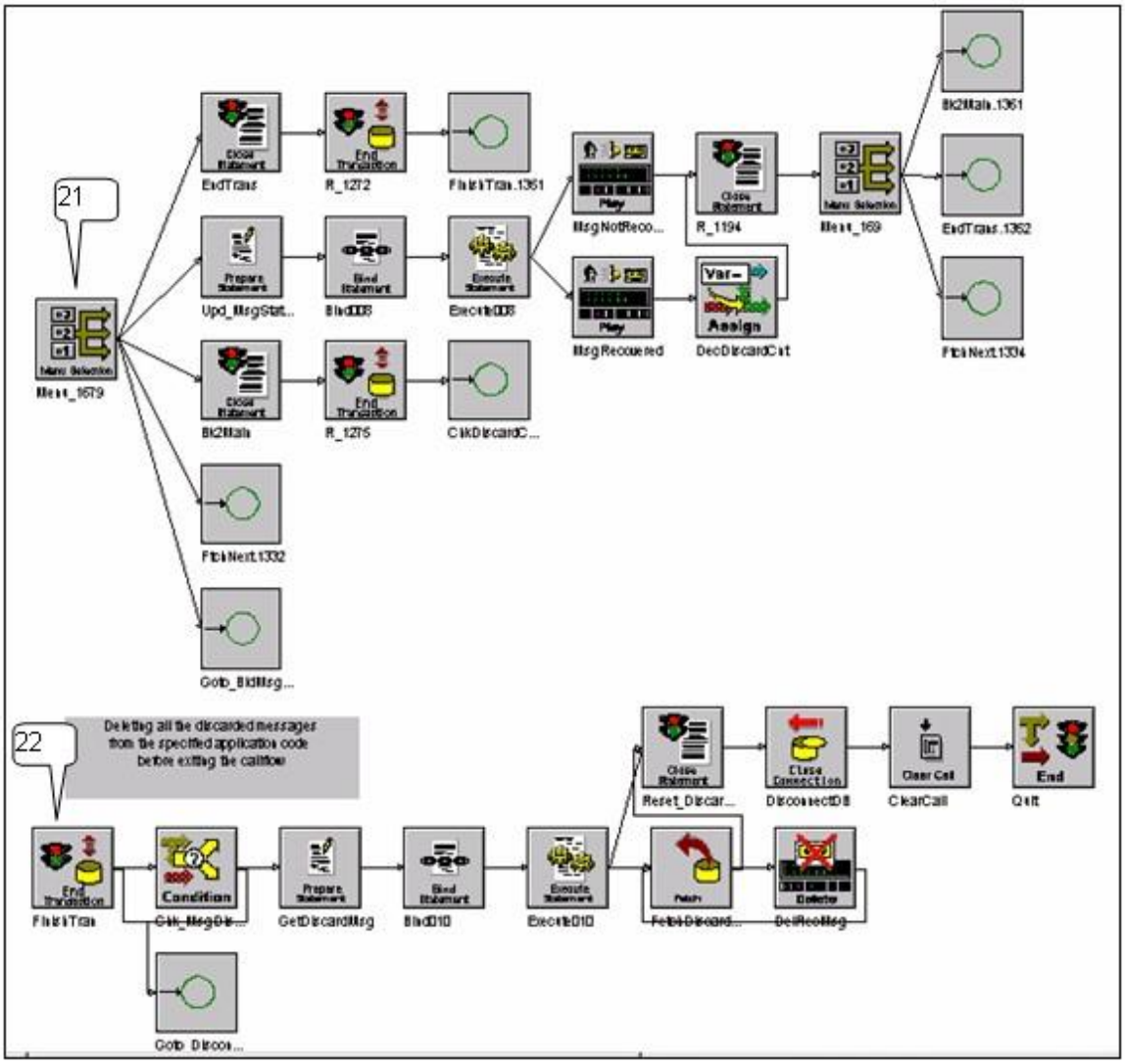
The ListenAll script allows the caller to listen to any one or all of the messages that are recorded for a specific application. This sample script demonstrates the use of most of the media components and ODBC components. This provides a base to build a simple voice mail application.

Note: Due to the size of the script, the entire workspace has been divided into four depictions starting from the upper-left portion of the workspace.




8
9
10
11
Condition
Get the message information
End

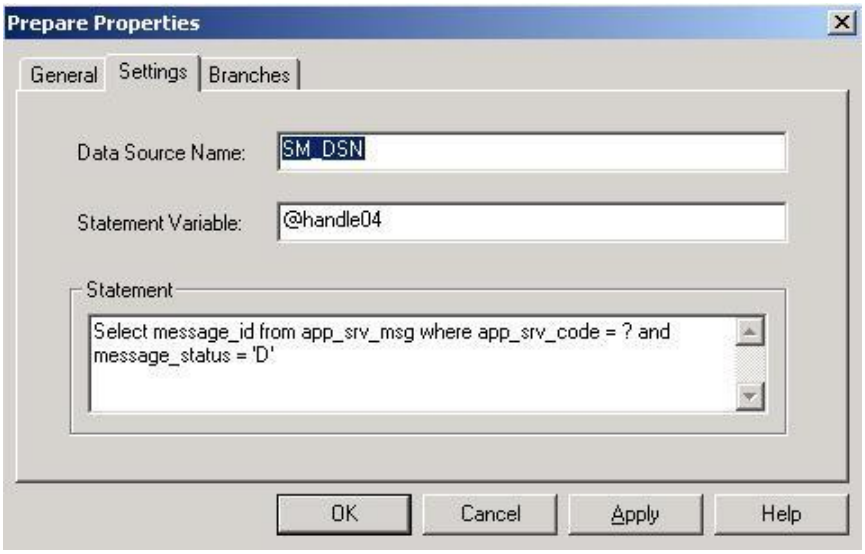

Playing Recorded Message
Play
Close Statement
NoMoreMsgT...
R_1264
Goto_Menu_19
Prepare Statement
Bind Statement
Execute Statement
Inc_MsgRd_Cnt
Bind005
Execute005
MsgStat...
15
18
Goto_Menu_1...
Prepare Statement
Bind Statement
UpdMsgStat_...
Bind007
Execute Statement
Execute007
Play
MsgDiscarded
Var=
Assign
IncDiscardCnt
R_1186
Goto_Menu_16...
MsgNotDisca...
R_1203
Goto_R_1194
Play
Play/Msg
12
13
16
Goto_EndTrans
PlayNext1330



Condition (1)

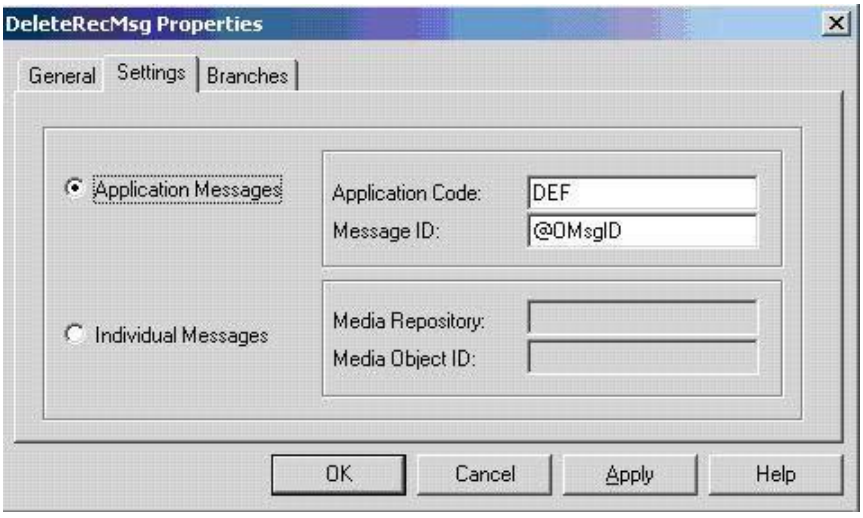
The Condition block checks for the number of discarded messages in variable @NumOfDisCardMsg. If the discarded message count is greater than zero, then the script updates the app_srv_table with a new status and deletes the voice file. Blocks (2) to (3) are for maintenance purposes. It is checking to make sure the voice messages that are marked for deletion are in fact removed from the system.

Prepare (2)



The Prepare block prepares the SQL statement to search for any message where message_status is ‘D’ with the matching application code. The application code is provided via the PIN validation. It binds to the application code, executes and then does a Fetch.

DeleteRecMsg (3)



For each successful Fetch, the voice message is removed from the OAS Server using the DeleteRecMsg block. It uses the @OMsgID retrieved from the Fetch block to delete. The script repeats the DeleteRecMsg block until all records marked for deletion have been removed. The DeleteRecMsg block automatic marks the message_status to ‘A’ to make it available for re-use.

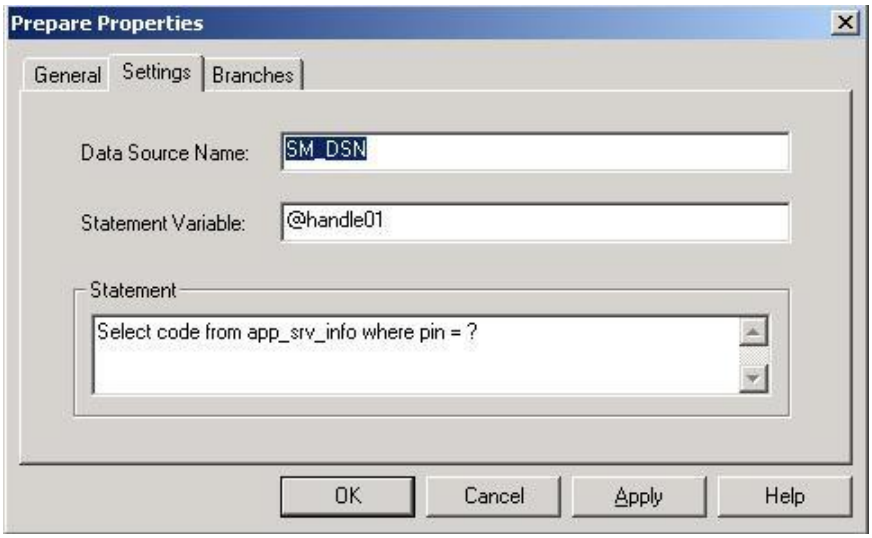
MenuSelection (4)

The MenuSelection allows the user to choose to retrieve the message by the message ID or by the application code. This is the first message that the user will hear when calling into the script.

GetDigit (5)

If the user selected to retrieve the message by application code, the user will be prompted to enter the PIN code.

Prepare (6)



The Prepare block retrieves the application code that matches with the PIN entered by the user. The application code is used to determine the voice messages that are available to the caller.

Prepare (7)

Select count(message_id) from app_srv_msg where app_srv_code = ? and message_status = 'N'

The application code is then used in the next Prepare block to get the number of new voice messages, that is message_status = 'N'.

Prepare (8)

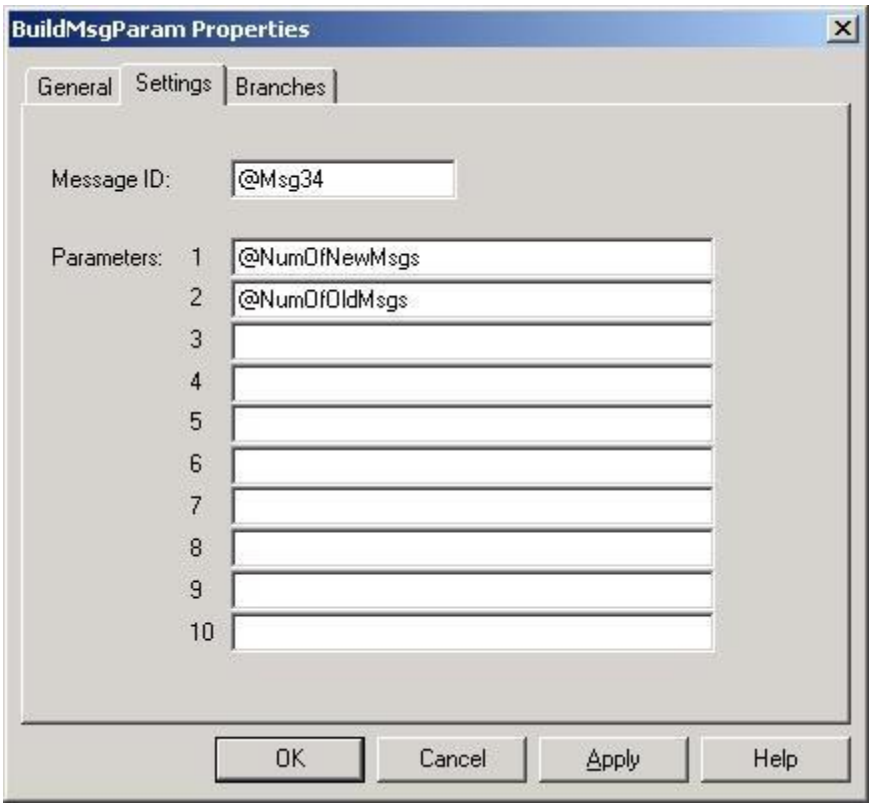
Select count(message_id) from app_srv_msg where app_srv_code = ? and message_status = 'R'

Then the same application code is used to get the number of old voice messages, that is message_status = 'R'.

Condition (9)

The Condition block checks the number of old messages and new messages. It branches to different BuildMsgParam blocks and prepares the prompts to play back to the caller.

BuildMsgParam (10)



The BuildMsgParam uses different play message IDs to prepare the message to be played. In this BuildMsgPararm block, @Msg34 expects two number parameters as defined in the OAS play messages. The number of new messages and the number of old messages will be inserted into the prompts. The next Play block will play the build messages.

Prepare (11)

Select message_id, media_repository, media_object_id, num_of_read, message_status, convert(varchar(10),message_timestamp,103), substring(convert(varchar(8),message_timestamp,8),1,5) from app_srv_msg where app_srv_code = ? and (message_status = 'N' or message_status = 'R') order by message_status

If there are any voice messages available, this Prepare block will retrieve the voice file and other information from the database.

BuildMsgParam (12)

The BuildMsgParam prepares the parameter for the voice messages, they are the @MediaObjID, @MsgTimestamp, and the @MsgDate. These three parameters are defined in the play message in @Msg36. The following Play block plays the prepared voice messages. It will play back the time and date when the message was received, and the recorded voice message.

Condition (13)

The Condition checks if it needs to update the message_status in the database. It branches to the Prepare (14) or (15) to update the message_status, and then it returns to the MenuSelection (16).

MenuSelection (16)

The MenuSelection block plays the selection options as provided in @Msg37. The user can choose to save the message, discard the message, repeat the message, goto the next message or exit the system. If the user chooses to listen to the next message, it goes back to the Fetch (17) block and retrieves the next message.

Prepare (18)

For other cases, the app_srv_msg table will be updated with the correct message_status.

MenuSelection (20)

The flow will take this branch if the user chooses to retrieve a specific message based on a message ID. This part of the script is similar to the Listen.mfd script.

MenuSelection (21)

The flow takes on this branch when the user selected to save the message. It will update the message_status and increment the read counter.

End Transaction (22)

This sequence of the script will clean up the discarded message before ending the flow.

TESTING THE APPLICATION

Make sure the pre-recorded voice prompts are copied to the OAS root container directory. The play message ID should also be defined as indicated in the following section. If any of the message IDs are defined differently, then the variable must be updated to the correct message ID after the service access is created.

VOICE PROMPTS

The following voice prompts are shipped with your system and are available in U.S. English only. After you have installed the MiCC Enterprise system, the voice prompts are located in your <InstallDir>\MiCC Enterprise\Localization\prompts directory. You can record new voice prompts for these message files; however, be sure to keep the same file names. These voice prompts are used in the Listen and ListenAll sample scripts to demonstrate the simple voice mail application.

	FILE NAME	CONTENT
1	ListenMg	...to listen to the new message
2	DelMsg	...to delete the message
3	ExitSys	...to exit the system
4	InvalOp	You have entered an invalid option. Please try again.
5	TryAgain	Please try again
6	EnterMgN	Please enter the message number
7	InvalidMg	Invalid message number.
8	SysNAvail	The system is not available at this time. Please call back later.
9	ThkCall	Thank you for calling.
10	MsgNDel	Message is not deleted.
11	MsgDel	...deleted
12	PleaseEn	Please enter ...
13	PINNum	...PIN number
14	MsgNum	...message number
15	ListenOp	Please enter PIN, or message number
16	NoMsg	You have no recorded message.
17	Youhave	You have ...
18	NewMsg	...new messages
19	OldMsg	...old messages
20	SavedMsg	...saved messages
21	MainMenu	...to go to the main menu
22	MsgRece	Message received on...

	FILE NAME	CONTENT
23	At	...at...
24	SaveMsg	...to save the message
25	DiscMsg	...to discard the message
26	RepMsg	...to listen to the same message again
27	NextMsg	...to listen to the next message
28	And	...and...
29	MsgNot	We are sorry that the message cannot be...
30	Recvered	...recovered
31	SysUnava	...system unavailable
32	DueTo	...due to...
33	PrevMsg	...to listen to the previous message.
34	Discard	...discarded
35	Msghasb	Message has been...