Oct 22, 2020 21:39

Smart Dialplan

Smart Dialplan explains how to use Custom Applications in Wildix Dialplan to interact with external software and databases.

Updated: April 2020

Permalink: https://confluence.wildix.com/x/swBuAQ

AGI in “Custom Application”

Variables in Dialplan

Variables are needed to store the information necessary to execute the application.

Syntax used to set a variable: **set(variablename=value)**

Where:
variablename: is the name of the variable, it is insensitive to the variables defined by users
value: is the value attributed to the variable; value can be a function.

Syntax used to call a variable: **${variablename}**

Example:

Set a variable: **set(foo=123456789)**

Call a variable: **${foo:offset:length}**

Where:
foo: is the name of the variable
offset: optional offset in starting to read a variable
length: optional number of characters to read

${foo} 123456789
${foO:1} 23456789
${Foo:-4:3} 678

Dialplan example:

- **set(FOO=101)** - sets the name of the variable as FOO and the value as 101
- **Dial(SIP/${FOO})** - dials 101

Here are some useful variables:

${CALLERID(all)}: The current Caller ID name and number

${CALLERID(name)}: The current Caller ID name

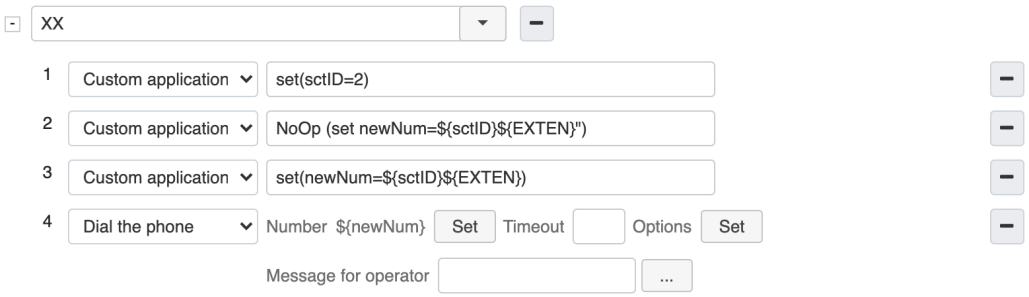
${CALLERID(num)}: The current Caller ID number ${ANSWEREDTIME}: Elapsed time since the call has been answered

${EXTEN}: Extension

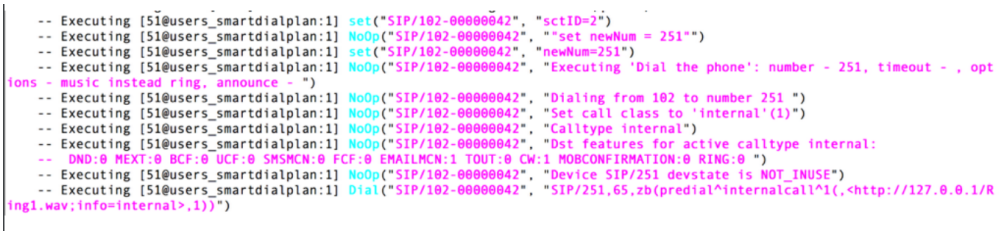
${CONTEXT}: Name of Dialplan procedure

${CHANNEL}: The channel (the called number present in the Dialplan procedure)

Example:



This Dialplan is assigned to user 102.
NoOp function serves to display the variable value in logs:



Conditions in Dialplan

You can use execute a conditional jump to another Dialplan procedure, based on the boolean result (true or false) of the **Gotoif** function.

Syntax to execute a jump: **GotoIf(condition?label1:label2)**

Where:

label1:label2 are the destinations of the jump, consisting of: **context**, **extension**, **priority**

- **context**: name of Dialplan procedure
- **extension**: extension or called number present in this Dialplan procedure
- **priority**: line number, usually 1

condition: the choice of Gotoif depends on the boolean result of this expression: true or false, in Dialplan True = 1 and False = 0.

The condition syntax is the following one: $[expr1 operator expr2], where

- **expr** can be a variable, a value, a function, an expression (expressions can be nested)
- **operator** as a rule it is relational (comparison) operator (=, <>..); for nested expressions it's possible to use logical operators AND, OR, NOT (explained below)

Logical operators:

expr1 | expr2 OR

expr1 & expr2 AND

!expr NOT

Relational operators:

expr1 = expr2 equal

expr1 != expr2 inequal

expr1 < expr2 less than

expr1 > expr2 greater than

expr1 <= expr2 less than or equal

expr1 >= expr2 greater than or equal

Example:

In this example:

set(FOO=1) - we introduce FOO variable with the value 1.

GotoIf($[${FOO} = 1]?users,101,1:users,102,1) - if FOO variable = 1, then jump to "users" procedure with extension 101; else - jump to "users" procedure with extension 102

GotoIf(condition?label1:label2) is introduced in our example in the following format:

GotoIf($[${FOO} = 1]?users,101,1:users,102,1), where:

- condition: $[${FOO} = 1]
- label1: users,101,1
- label2: users,102,1

Example 1: Check if a device is INUSE before sending a call

GotoIf($[${DEVICE_STATE(SIP/100)} == NOT_INUSE]?context,101,1)

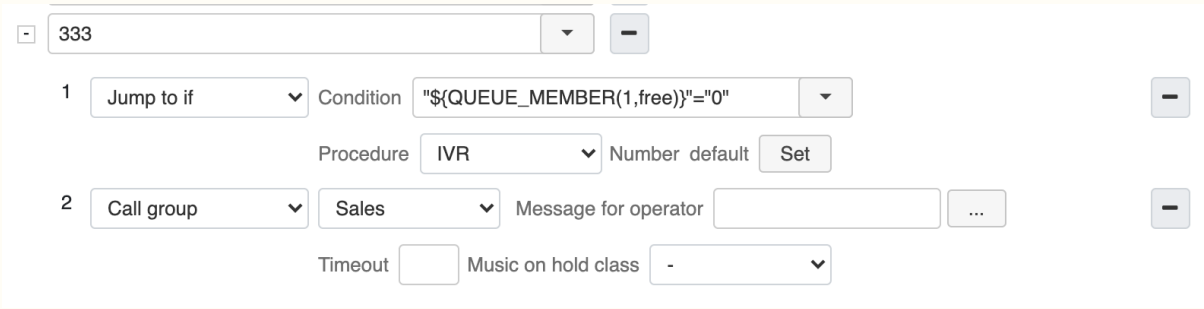
The function **DEVICE_STATE(SIP/100)** returns the state of the device, results can be:
UNKNOWN | NOT_INUSE | INUSE | BUSY | INVALID |UNAVAILABLE | RINGING | RINGINUSE | ONHOLD

Example 2: Check if there are available users in a call group before sending a call

GotoIf($[${QUEUE_MEMBER(2,ready)} == 0]?context,101,1)

The function QUEUE_MEMBER(2,ready) counts the number of ready members of a call group

With the help of QUEUE_MEMBER function it is possible to configure CG strategy to execute another Dialplan procedure. The function counts the number of free members of a Call group. When there are no available CG members, an incoming call is routed according to another Dialplan procedure.



- **Jump to if -> "${QUEUE_MEMBER(<call_group_id>,free)}"="0"**

where

"free" - returns the number of not paused / busy members for a specified queue that can answer calls or are currently paused for the duration of wrap up time after a previous call (free CG members).

Note: Wrap up time and ringing state are not considered when using the function.

For example: All Call group members are put on pause for the duration period of wrap up time -> the incoming call stays in a queue until wrap up time doesn't expire

Limitation: the system recognizes unregistered (offline) CG members as available for answering calls. In case unregistered members are present in CG -> QUEUE_MEMBER = 0 performance is false.

Example 3: Check if the CALLERID(number) matches a regular expression

GotoIf($[${REGEX("^0[6-7][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$" ${CALLERID(number)})}]?noanswer, 01,1)

The function REGEX("<regular expression>" <data>) returns true if the regular expression matches the data.

In the example above we check if the caller is a French mobile phone:
Start by 0 then a 6 or 7 then followed by 8 digits => ^0[6-7][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$

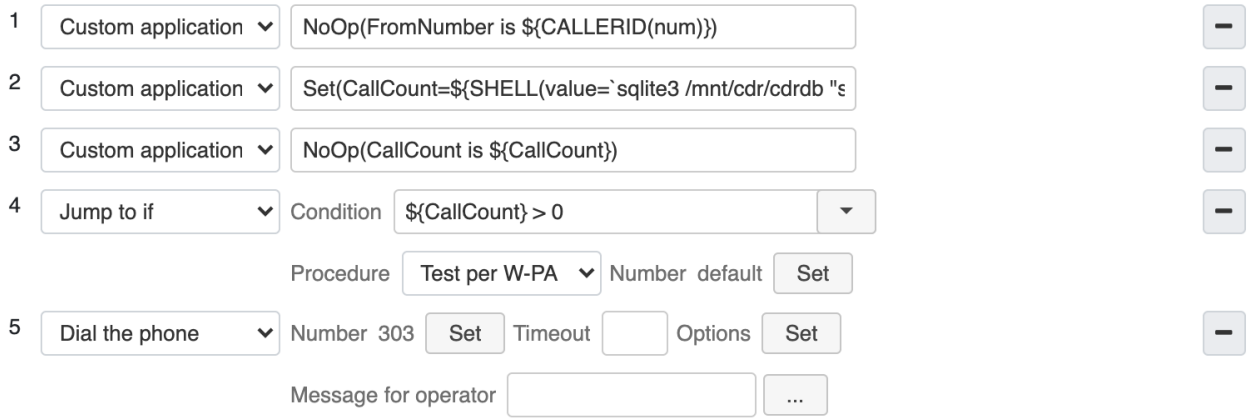
Example 4: Check if the call arriving in the system is from a caller that has called in recently and not been answered - example of sqlite query using SHELL COMMAND

The variable helps to recognize that a call entering the system was from a caller that had recently called in and not been answered. In this case, the call can be routed differently, for instance, to a Call Group.

What it does:
Gets a count of calls from the caller ID number of the current call that:

- had called in in the past 5 minutes
- was NOT answered

For this purpose, a query to CDR is made to get a count of unanswered calls from the Caller ID number over the previous X minutes.



NoOp(FromNumber is ${CALLERID(num)}) - detects the Caller ID.

Set(CallCount=${SHELL(value=`sqlite3 /mnt/cdr/cdrdb "select count() from cdr where c_from = '+1${CALLERID(num)}' and answer = '' and start > Datetime('now' ,'localtime','-5 minutes');"` && echo $value)}):

- **Set(CallCount=** - sets the value of a variable called "CallCount" (can then be used to check later with the 'jump to if' application
- **${SHELL(value=`sqlite3 /mnt/cdr/cdrdb** - executes a Linux SHELL command (sqlite query in this case)
- **"select count() from cdr where** - gets count of calls from cdrdb
- **c_from = '+1${CALLERID(num)}'** - checks cdrdb c_from field (contains caller ID numbers; note that +1 in our example is a prefix – change it with your country's prefix or remove it)
- **answer = ''** - checks cdrdb answer field (contains answer timestamps of previous calls; blank if call was not answered)
- **start > Datetime('now' ,'localtime','-5 minutes');"`** - checks start field in cdrdb for a date/timestamp within the past 5 minutes (notice the -5 minutes portion)
- **&& echo $value** - syntax required by the SHELL command to return the value of the linux shell command that is being executed

NoOp(CallCount is ${CallCount}) - detects a count of calls.

${CallCount} > 0 - If a count of calls is greater than 0, a call will be routed to the Dialplan procedure.

Note: The same scenario can be set up for checking outgoing calls, refer to the Article: How to check whether missed call has been called back.

In case of using MySQL or MSSQL DB, apply one of these custom applications:

MySQL:
Set(CallCount=${SHELL(value=`echo "select count(*) from <TABLE> where c_from = '${CALLERID(num)}' and start = answer and start > date_add(LOCALTIME(),INTERVAL - 5 MINUTE);" | mysql -s --raw -h <DB_HOST> -P 3306 <DATABASE> -u <USER> -p <PASSWORD>` && echo $value)})

MSSQL:
Set(CallCount=${SHELL(value=`echo -e "select count(*) from <TABLE> where c_from = '${CALLERID(num)}' and answer = '1900-01-01 00:00:00.000' and start > dateadd(MI,-5,GETDATE());\nGO\nEXIT\n" | tsql -S <DB_HOST> -p 1433 -U <USER> -P <PASSWORD> -D <DATABASE> | sed '6!d'` && echo $value)})

where:
<DATABASE> - database name
<TABLE> - table name
<DB_HOST> - hostname
<USER> - user
<PASSWORD> - password

Important: Time zone of the PBX and the DB server should match!

Example 5. Check if a user/ users are registered

- Jump to if -> **${REGUSERS(152)}=0** - if user 152 has a registered device (excluding push), the execution result will be "1", otherwise "0"
- Jump to if -> **${REGUSERS(200,201,202)}=0** - if at least one of the users has at least one registered device (excluding push), the execution result will be "1", otherwise "0"

Request to remote server: CURL

CURL allows sending a request to a web page. As a rule, a CURL(url) is used together with Set() in order to write a value received from a web page into a variable:

Set(variablename=${CURL(URL)})

or

Set(variablename=${FILTER(1234567890,${CURL(URL)})
It is recommended to use the second request so that you escape the unwanted characters.

Example:

Set(foo=${FILTER(1234567890,${CURL(http://myexample.wildix.com/smartdialplan/echo101.php)})})

In this example the value returned is 101.

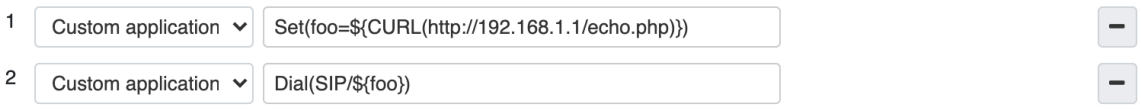
In this way we can pass the caller number to a web page using ${CALLERID(num)} as an URL parameter:
Set(foo=${CURL(http://192.168.1.1/callernum.php?callernum=${CALLERID(num)})})

To keep it simple, a remote script must return a string or a character. The url string that can integrate variables or functions.

Example: a simple PHP script that returns the value “101” from a remote server:

echo.php

```
<?php
echo 101;
?>
```



Set(foo=${CURL(http://192.168.1.1/echo.php)})

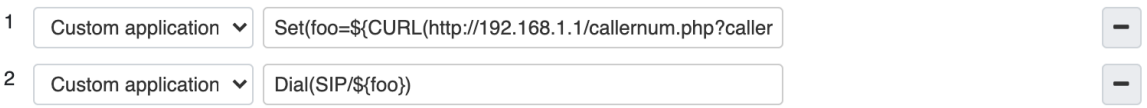
Dial(SIP/${foo})

Add parameters in CURL url

In the following example the PHP script returns “102” if “callernum” parameter is 0612345678, else the script returns “101”.

callernum.php

```
<?php
if (isset($_GET['callernum']))
{
     $callernum = $_GET['callernum'];
}
Else
{
     $callernum="UNKOWN ID";
}
if ($callernum === "0612345678") {
echo 102;
}
Else
{
echo 101;
}
?>
```



Set(foo=${CURL(http://192.168.1.1/callernum.php?callernum=${CALLERID(num)})})

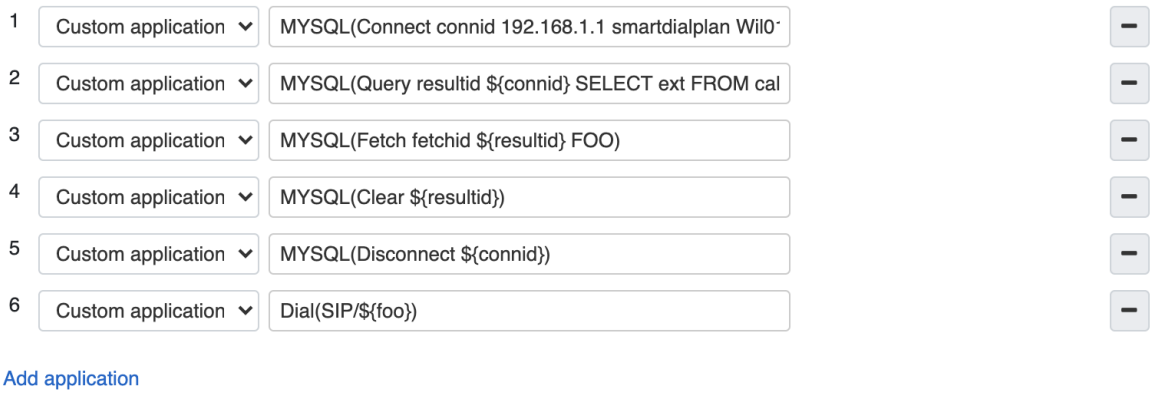
Dial(SIP/${foo})

We use the predefined variable **${CALLERID(num)}** in the url, but it can be also a function or a variable defined previously.

Request to MySQL server

Dialplan can interact with databases, e.g. MySQL. We can send a request to a MySQL server with **MYSQL()** function.

A database request requires several steps, here is an example:



MYSQL(Connect connid 192.168.1.1 smartdialplan Wil01diX@@ smartdialplan)
MYSQL(Query resultid ${connid} SELECT ext FROM callernum WHERE callernum=${CALLERID(num)})
MYSQL(Fetch fetchid ${resultid} FOO)
MYSQL(Clear ${resultid})
MYSQL(Disconnect ${connid})

Connect: Establishing connection to the BDD
Query: Request execution
Fetch: Reading the result
Clear: Memory cleaning
Disconnect: Disconnection from the BDD

Details of each step:

MYSQL(Connect connid dhhost dbuser dbpass dbname)
Connection to the database: at this stage the credentials, the name of the server and the database are provided. MySQL parameters are passed through the function mysql_real_connect. Connection identifier is returned in ${connid}. If the connection could not be established, then it returns ${connid} == "".

MYSQL(Query resultid ${connid} query-string)
Executes standard MySQL query contained in query string using established connection identified by ${connid}. Result of query is stored in ${resultid}.

MYSQL(Fetch fetchid ${resultid} var1\ var2\ ...\ varN)
If result is fetched, ${fetchid} is set to 1, and the single result is contained in ${resultid} and returned fields are written to ${var1}, ${var2} ... ${varN} respectively. In case of no result, ${fetchid} is set to 0.

MYSQL(Clear ${resultid})
Frees the memory associated to the connection and results.

MYSQL(Disconnect ${connid})
Disconnection from the database.

Example: the database is set as follows:

smartdialplan

Callernum	Ext
0123456789	101
0612345678	102

Set MySQL privileges for the PBX to be able to remotely access to this database.

Use DTMF in call

DTMF allows interaction of machines with humans and uses the following functions: Read(), SayNumber() and SayDigits().

Read(variable[,filename][,maxdigits][,option][,attempts][,timeout])
Writes a string of digits (terminated by # or containing a certain number of digits)

filename: file playback before reading the digits
maxdigits: max acceptable number of digits (in this case user is not required to enter #)
option: There are 3 options:

s(skip): continue the execution immediately if the line is not available.
i(indication): play an indication in case line is not available
n(noanswer): read digits even in case line is not available

attempts: (if greater than 1) number of attempts in the event no data is entered.
timeout: timeout in seconds; if greater than 0, that value overrides the default timeout.

SayDigits(digits)
Says the digits, one by one, digits can be a variable.

SayNumber(number, gender)
Says number
gender is "f" for female voice; "m" - for male ;"c" - for neutral

Example:

Read(FOO,00000/smartdialplan/numcommande,3,,,10)

SayNumber(${FOO},f)

Dial(SIP/${FOO})

Implementation of Smart Dialplan: limited access to the phone based on the amount of credits purchased

This service allows granting a user limited access to phone in a hotel or a camping, based on the number of credits he or she has purchased.

How it works:

Database Table: Users

ID	Current_units
1234	600
2345	234

- User calls the dedicated code (555) from the phone
- User enters a secret code for authentication
- Connection to the database is established to check if this code valid; if the code is wrong, playback “authentication failed”
- Check in the database the amount of the remaining credits; if there are no credits left, playback “no more credits”; else - ask to enter the number consisting of 10 digits
- The max duration of a call is defined based on the number of credits left
- The other party is called
- Once the call is terminated (by user or if reached the max duration of a call), the database values are updated

Implementation:

1. Create a dedicated Dialplan procedure “accounting”
2. Record the missing sounds using Sounds menu
3. Prepare the database

After these initial steps it's necessary to set up "accounting" procedure:

Context 555 (called number 555)

"Custom applications":

Read(CALLER,00000/accounting/code,,,,10)

(secret code for authentication)

MYSQL(Connect connid support-fr.wildix.com odessa2016-12 oDess@0912to18 odessa2016-12)

MYSQL(Query resultid ${connid} SELECT credit FROM accounting WHERE id=${CALLER})

MYSQL(Fetch fetchid ${resultid} Current_units)

MYSQL(Clear ${resultid})

NoOp(Current Units : ${Current_units})

(connection to the DB and check the number of credits)

GotoIf($[${Current_units} != NULL]?accounting,0,1:accounting,1,1)

(jump to context 1 if code is wrong, else - context 0)

Context 1 (called number 1)

Play sound "authentication failed"

Hang up

Context 0 (called number 0)

Custom application **GotoIf($[${Current_units} > 0]?accounting,00,1:accounting,01,1)**

Check remaining credits if = 0, go to context 01, if more than 0, go to context 00

Context 01 (called number 01)

Play sound "no more credits"

Hang up

Context 00 (called number 00)

Custom applications:

Read(CALLED,00000/accounting/numcorresp,10,,,10)

(requires user to enter 10 digit called number)

Goto(accounting,${CALLED},1)

Context 0XXXXXXXXX (called number entered by user)

Custom application **Set(TIMEOUT(absolute)=$[${Current_units} + 10])**

(set up max duration time of the call)

Dial the trunk

h context (called number h) - actions to be executed after hangup (update the max duration and the number of credits in the database)

Custom applications:

Set(New_units=$[${Current_units} - ${ANSWEREDTIME}])
MYSQL(Query resultid ${connid} UPDATE accounting SET credit=${New_units} WHERE id=${CALLER})
MYSQL(Disconnect ${connid})

Dialplan debug:

-- Executing [555@accounting:1] Read("SIP/252-0000000f", "CALLER,00000/accounting/code,,,,10")
-- <SIP/252-0000000f> Playing '00000/accounting/code.g729' (language 'fr')
-- User entered '1234'
-- Executing [555@accounting:1] MYSQL("SIP/252-0000000f", "Connect connid support-fr.wildix.com odessa2016-12
oDess@0912to18 odessa2016-12")
-- Executing [555@accounting:1] MYSQL("SIP/252-0000000f", "Query resultid 1 SELECT credit FROM accounting WHERE id=1234") -- Executing [555@accounting:1] MYSQL("SIP/252-0000000f", "Fetch fetchid 2 Current_units")
-- Executing [555@accounting:1] MYSQL("SIP/252-0000000f", "Clear 2")
-- Executing [555@accounting:1] NoOp("SIP/252-0000000f", "Current Units : 94")
-- Executing [555@accounting:1] GotoIf("SIP/252-0000000f", "1?accounting,0,1:accounting,1,1")
-- Goto (accounting,0,1)
-- Executing [0@accounting:1] GotoIf("SIP/252-0000000f", "1?accounting,00,1:accounting,01,1")
-- Goto (accounting,00,1)
-- Executing [00@accounting:1] Read("SIP/252-0000000f", "CALLED,00000/accounting/numcorresp,10,,,10")
-- Accepting a maximum of 10 digits.
-- <SIP/252-0000000f> Playing '00000/accounting/numcorresp.g729' (language 'fr')
-- User entered '0176747983'
-- Executing [00@accounting:1] Goto("SIP/252-0000000f", "accounting,0176747983,1")
-- Goto (accounting,0176747983,1)
-- Executing [0176747983@accounting:1] Set("SIP/252-0000000f", "TIMEOUT(absolute)=104")

-- Channel will hangup at 2016-08-18 13:00:21.241 CEST.
-- Executing [0176747983@accounting:1] NoOp("SIP/252-0000000f", "Executing 'Dial the trunk': number - 0176747983, tr_name
-0276510950, callclass - auto")
-- Executing [0176747983@accounting:1] Dial("SIP/252-0000000f", "SIP/ 0276510950/0176747983,,b (predial^extcall^1(+33176747983,,))")
[Aug 18 12:58:50] WARNING[3772][C-0000000d]: chan_sip.c:32923 handle_request_bye: Got request BYE from peer 252; From: "Trunk test2" <sip:252@frtest.wildixin.com:443>;tag=676021154; To: <sip:555@frtest.wildixin.com:443>;tag=as076b897a; callid - '562925853'. Reason cause: 16 (Normal Clearing)
-- Executing [h@accounting:1] Set("SIP/252-0000000f", "New_units=84")
-- Executing [h@accounting:1] MYSQL("SIP/252-0000000f", "Query resultid 1 UPDATE accounting SET credit=84 WHERE id=1234") -- Executing [h@accounting:1] MYSQL("SIP/252-0000000f", "Disconnect 1"

phpAGI in “Remote script”

Integrate PHP in dialplan

You need to:

- Upload PHP AGI (phpagi.sourceforge.net) to **/var/www/agi/phpagi/phpagi.php**
- Upload test script to **/var/www/scripts/test.php**

test.php

```
#!/usr/bin/php
<?php
require '/var/www/agi/phpagi/phpagi.php';
$agi = new AGI();
$agi->exec(Dial, "SIP/102");
?>
```

Then you can use “remote script” in dialplan, example: Remote script

Microsoft SQL Integration

Create **/rw2/var/www/scripts/sqllookup.php**

```
#!/usr/bin/php
<?php
require_once '/var/www/autoload.php';
$agi = new AGI();
$dsn="dblib:host=IP:PORT;dbname=DBNAMEK";
$conn = new PDO($dsn, "USER", "PASSWORD");
$cid = substr($agi->request[agi_callerid],1);
$sql = "Exec dbo.SPR_CRM_ValidateCustomer @Phonenumber = '".$agi->request[agi_callerid]."'";
fwrite(STDERR, "\n".$cid.$sql."\n");
$result = $conn->query($sql);
if ($result) {
 foreach ($result as $row) {
  $agi->exec('Set', "STOREID=". $row[StoreID]);
 }
} else {
 foreach ($conn->errorInfo() as $row) {
  fwrite(STDERR, print_r($row));
 }
```

```
 fwrite(STDERR, "\nPDO::errorCode(): ". $conn->errorCode(). "   ".$conn->errorInfo()."\n");
}
?>
```

Call Remote script it in Dialplan:

Call file directory to generate a call

The default path of this directory is /var/spool/callweaver/outgoing/ but it's possible to use an external shared directory: WMS->Settings->System->Storages->Add new Windows share->Service Call generation

To generate a call we just need to move a *.call file in this directory.

> Attention! A mv (move) command is an atomic operation (an operation which does not take effect until it is 100% complete) and as such it is perfectly suited to move .call files. With cp (copy) command, the file is copied line by line, which could lead to PBX engine processing an incomplete file.

Call file Syntax

A simple call file example which dials a number and plays a sound file.

Test.call

```
------------------------------------------------------
Channel: SIP/trunkname/18882223333
Application: Playback
Data: 0000/hello-world
------------------------------------------------------------------
```

Syntax of call files

Specify the call destination and the channel to use:

Channel: <channel>: Channel to use for the call.
CallerID: "name" <number> Caller ID, please note: it may not work if you do not respect the format: CallerID: "Some Name" <1234>
MaxRetries: <number> Number of retries before failing (not including the initial attempt, e.g. 0 = total of 1 attempt to make the call). Default is 0.
RetryTime: <number> Seconds between retries. Default is 300 (5 min).
WaitTime: <number> Seconds to wait for an answer. Default is 45.

If the call answers, connect it here:

Context: <context-name> Context in extensions.conf
Extension: <ext> Extension definition in extensions.conf
Priority: <priority> Priority of extension to start with
Set: Set a variable to be used in the extension logic (**example: file1=/tmp/to**)
Application: Application to run (use it instead of specifying context, extension and priority)
Data: The options to be passed to the application

Use Callfile in dialplan

Create a script file in **/var/www/scripts** dir.

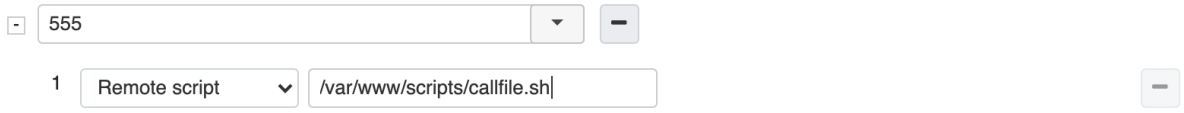
This script will be executed by dialplan.

Copy a template file and move the duplicated file to **/var/spool/callweaver/outgoing/**

callfile.sh

```
-----------------------------------------------------
#! /bin/sh
cp /var/www/scripts/test.call /var/www/scripts/temp.call
mv /var/www/scripts/temp.call /var/spool/callweaver/outgoing/
-----------------------------------------------------
Give permissions to execute the script.
#chmod +x /var/www/scripts/callfile.sh
Create a template file in /var/www/scripts dir.
Test.call
-----------------------------------------------------
Channel: SIP/0276510950/0970720101
Application: Playback
Data: 00000/callfile/message
-----------------------------------------------------
```

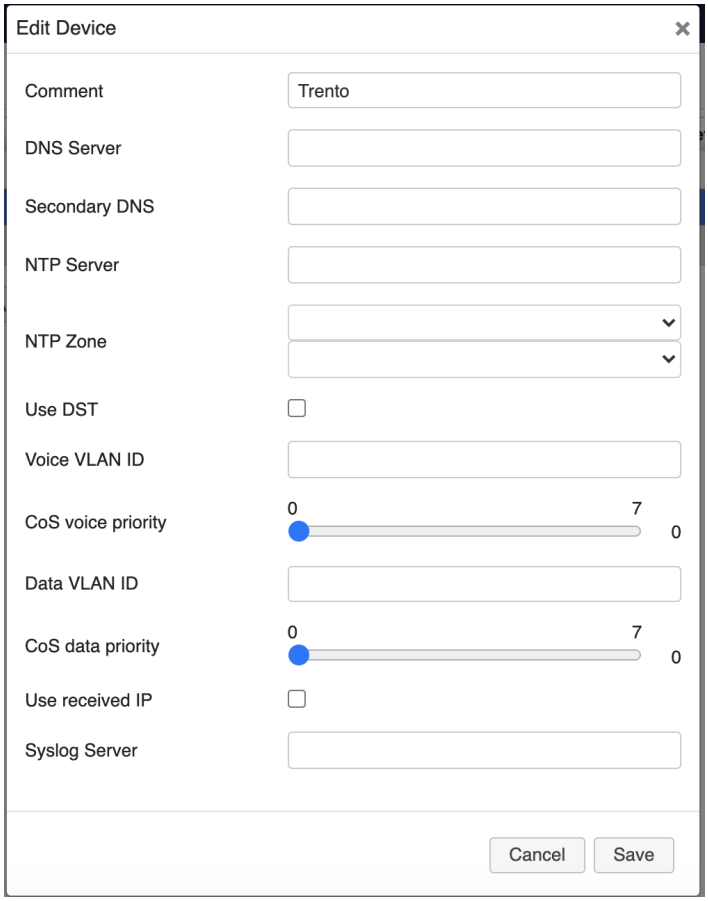
Call a remote script in dialplan.



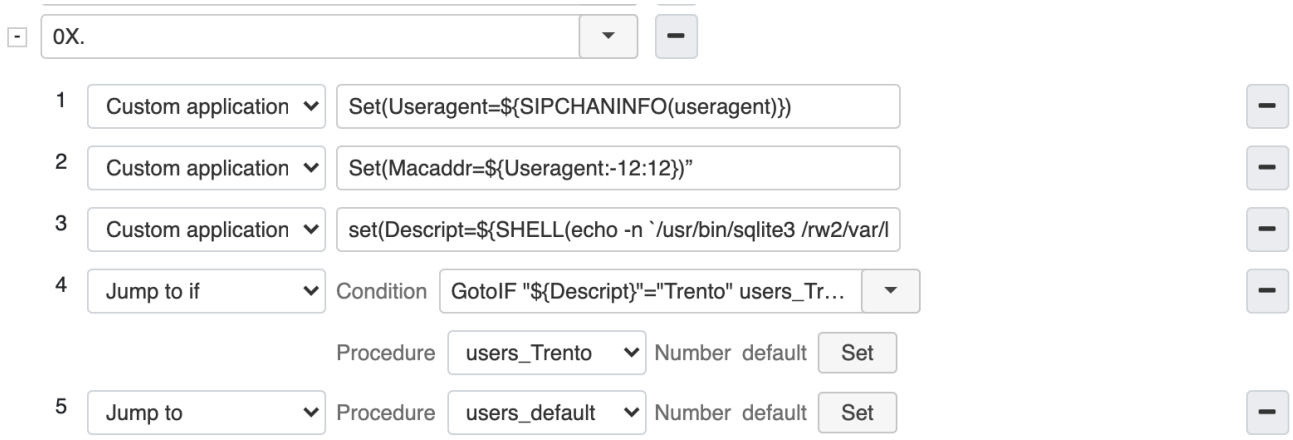
Use device comment field for customize CID of outgoing call

Set comment in *WMS -> Devices* for devices:

- Double click on a device to edit it
- Fill out the "*Comment*" field and click **Save**



In users dialplan add the following Custom apps:



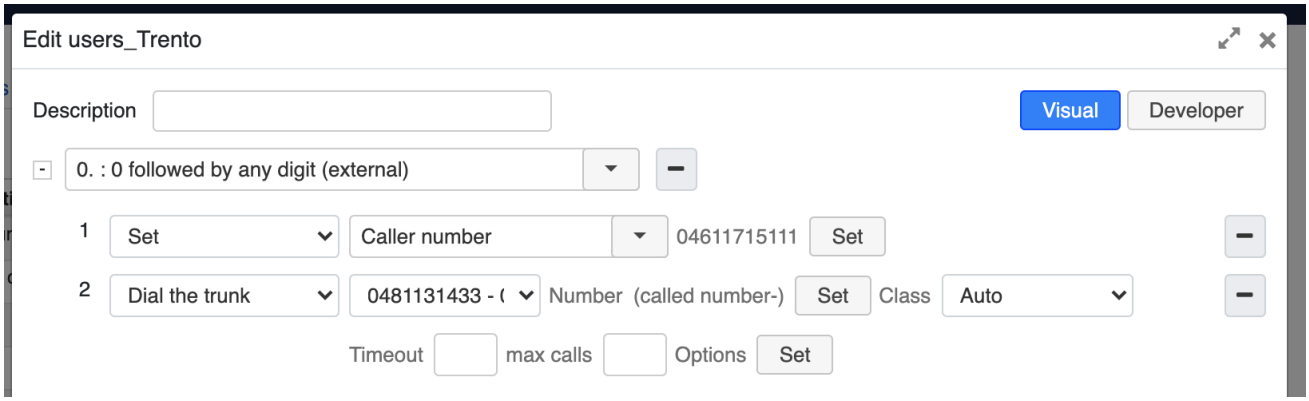
Set(Useragent=${SIPCHANINFO(useragent)})

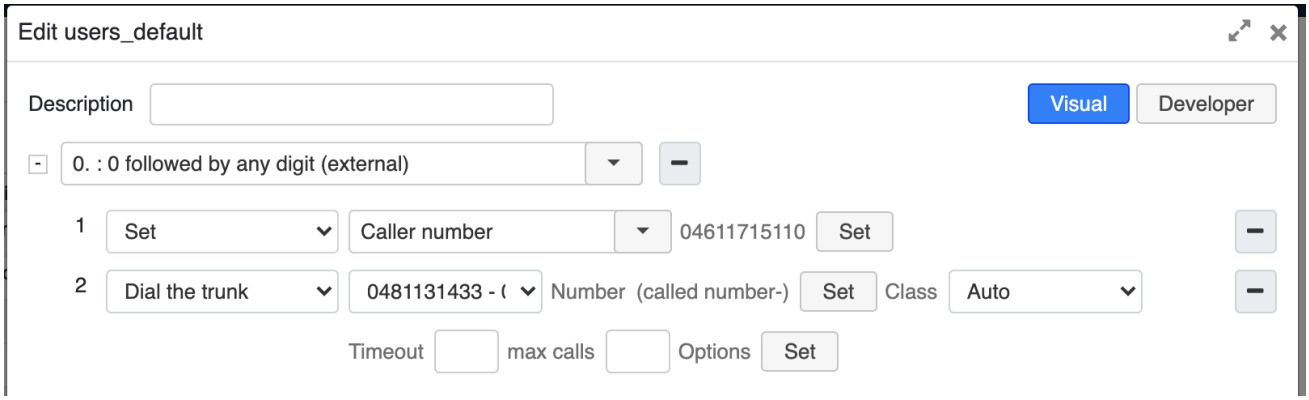
Set(Macaddr=${Useragent:-12:12})”
set(Descript=${SHELL(echo -n `/usr/bin/sqlite3 /rw2/var/lib/wmsdb.d/devdb "SELECT value FROM autoprov WHERE param = 'description' and mac = '${Macaddr}' COLLATE NOCASE"`)})

Then add these apps:

GotoIF "${Descript}"="Trento" users_Trento
Goto users_default

Prior to this, you should create dialplans named like this **users_${Descript}** and another one -- *users_default* in our example -- for calls don’t match to a device with comment field.





External outgoing call with audio file playback

Here is our scenario: outgoing call to urban line is placed; this call is being recorded, once it is answered, an audio file is played back automatically informing the called party that this call is being recorded.

This scenario can be carried out using Custom Application Dial:

Dial(SIP/pri1_0_0/${CalledID},30,A(00000/Messages/Recording) where pri1_0_0 is the channel used (PRI in this case), 00000 is the folder that contains the audio messages to be played back.

To remove digits from the called number, you can use function:

Dial(SIP/0123456789/${EXTEN:3},30,A(00000/Message/Recording) - in this example SIP trunk is used and ${EXTEN:3} removes 3 first digits from the called number.

What else can you do with Custom Dialplan Apps?

Delayed Paging

Read the doc: Delayed Paging

Notify callers of call recording

Read the doc: How to notify callers of call recording

Callback feature for Call groups

Read the doc: How to enable Callback feature for Call groups

Director-Secretary configuration

Read the doc: Director - Secretary configuration

Initiate an audio conference using API Originate

Read the doc: How to automatically initiate an audio conference using API Originate

Block outgoing calls from hotel rooms

Read the doc: How to block outgoing calls from hotel rooms

Change Voicemail PIN from a phone

Read the doc: How to change Voicemail PIN from a phone

Automatically generate a call and play a voice message

Read the doc: Automatically Generate a Call and Play a Recording

Record multiple answers from a caller and combine them into one file

Read the doc: Recording multiple answers from a caller and combining them into one file