Asterisk 11 Reference

Asterisk Development Team <asteriskteam@digium.com>

New in 11

Please read the upgrade notes at Upgrading to Asterisk 11 or in UPGRADE.txt before migrating an existing installation to Asterisk 11.

In Brief

Asterisk 11 introduces several new features that build on the features in Asterisk 10. Highlights include:

- Call Identifier Logging, which makes it easier for system admins to debug problems in their deployments
- Named Callgroups and Pickupgroups, which effectively removes the 64 callgroup/pickupgroup limitation
- General and Technology Specific Hangup Cause querying from the dialplan
- Hangup Handlers, which simplify hangup routine implementations in the dialplan
- Pre-dial routines, which let a dialplan writer execute a routine on a channel during construction
- Initial support for WebRTC, including support for ICE, DTLS-SRTP, and SIP over Websockets
- Named ACLs, which let ACLs be defined once and referred to by name from any module
- A brand new XMPP channel driver `chan_motif`, which has revamped support for the various Google Jingle protocols

And much more. A full listing is below.

Build System

- The Asterisk build system will now build and install a shared library (*libasteriskssl.so*) used to wrap various initialization and shutdown functions from the *libssl* and *libcrypto* libraries provided by OpenSSL. This is done so that Asterisk can ensure that these functions do **not** get called by any modules that are loaded into Asterisk, since they should only be called once in any single process. If desired, this feature can be disabled by supplying the `--disable-asteriskssl` option to the configure script.
- A new make target, `full`, has been added to the Makefile. This performs the same compilation actions as `make all`, but will also scan the entirety of each source file for documentation. This option is needed to generate AMI event documentation. Note that your system must have Python in order for this make target to succeed.
- The optimization portion of the build system has been reworked to avoid broken builds on certain architectures. All architecture-specific optimization has been removed in favor of using `-march=native` to allow gcc to detect the environment in which it is running when possible. This can be toggled as `BUILD_NATIVE` under "**Compiler Flags**" in menuselect.
- `BUILD_CFLAGS` and `BUILD_LDFLAGS` can now be passed to menuselect, e.g., `make BUILD_CFLAGS="whatever"` or `BUILD_LDFLAGS="whatever"`
- Remove *asterisk/version.h* in favor of *asterisk/ast_version.h*. If you previously parsed the header file to obtain the version of Asterisk, you will now have to go through Asterisk to get the version information.

Applications

Bridge

- Added '**F()**' option. Similar to the Dial option, this can be supplied with arguments indicating where the callee should go after the caller is hung up, or without options specified, the priority after the call to Bridge will be used.

ConfBridge

- Added menu action `admin_toggle_mute_participants`. This will mute / unmute all non-admin participants on a conference. The confbridge configuration file also allows for the default sounds played to all conference users when this occurs to be overriden using `sound_participants_unmuted` and `sound_participants_muted`.
- Added menu action `participant_count`. This will playback the number of current participants in a conference.
- Added announcement configuration option to user profile. If set the sound file will be played to the user, and only the user, upon joining the conference bridge.

Dial

- Added '**b**' and '**B**' options to Dial that execute a Gosub on callee and caller channels respectively before the callee channels are called. See pre-dial handlers for more information.

ExternalIVR

- Added support for IPv6.
- Add interrupt ('**I**') command to ExternalIVR. Sending this command from an external process will cause the current playlist to be cleared, including stopping any audio file that is currently playing. This is useful when you want to interrupt audio playback only when specific DTMF is entered by the caller.

FollowMe

- A new option, '**I**' has been added to FollowMe. By setting this option, Asterisk will not update the caller with connected line changes when they occur. This is similar to options in Dial and Queue.
- The '**N**' option is now ignored if the call is already answered.
- Added '**b**' and '**B**' options to FollowMe that execute a Gosub on callee and caller channels respectively before the callee channels are called. For more information, see pre-dial handlers.
- The winning FollowMe outgoing call is now put on hold if the caller put it on hold.

MixMonitor

- MixMonitor hooks now have IDs associated with them which can be used to assign a target to StopMixMonitor. Use of MixMonitor's **i(variable)** option will allow storage of the MixMontior ID in a channel variable. StopMixMonitor now accepts that ID as an argument.
- Added '**m**' option, which stores a copy of the recording as a voicemail in the indicated mailboxes.

MySQL

- The connect action in app_mysql now allows you to specify a port number to connect to. This is useful if you run a MySQL server on a non-standard port number.

OSP Applications

- Increased the default number of allowed destinations from 5 to 12.

Page

- The app_page application now no longer depends on DAHDI or app_meetme. It has been re-architected to use app_confbridge internally.

Queue

- Added queue options `autopausebusy` and `autopauseunavail` for automatically pausing a queue member when their device reports busy or congestion.
- The `ignorebusy` option for queue members has been deprecated in favor of the option `ringinuse`. Also a `queue set ringinuse` CLI command has been added as well as an AMI action QueueMemberRingInUse to set this variable on a per interface basis. Individual `ringinuse` values can now be set in *queues.conf* via an argument to member definitions. Lastly, the queue `ringinuse` setting now only determines defaults for the per member `ringinuse` setting and does not override per member settings like it does in earlier versions.
- Added '**F()**' option. Similar to the option in Dial, this can be supplied with arguments indicating where the callee should go after the caller is hung up, or without options specified, the priority after the Queue will be used.
- Added new option `log_member_name_as_agent`, which will cause the membername to be logged in the agent field for `ADDMEMBER` and `REMOVEMEMBER` queue events if a `state_interface` has been set.

SayUnixTime

- Added '**j**' option to SayUnixTime. SayUnixTime no longer auto jumps to extension when receiving DTMF. Use the '**j**' option to enable extension jumping. Also changed arguments to SayUnixTime so that every option is truly optional even when using multiple options (so that **j** option could be used without having to manually specify timezone and format) There are other benefits, e.g., format can now be used without specifying time zone as well.

Voicemail

- Addition of the VM_INFO function - see Function changes.
- The `imapserver`, `imapport`, and `imapflags` configuration options can now be overriden on a user by user basis.
- When voicemail plays a message's envelope with `saycid` set to yes, when reaching the caller id field it will play a recording of a file with the same base name as the sender's callerid if there is a similarly named file in *<astspooldir>/recordings/callerids/*.
- Voicemails now contains a unique message identifier `msg_id`, which is stored in the message envelope with the sound files. IMAP backends will now store the message identifiers with a header of "`X-Asterisk-VM-Message-ID`". ODBC backends will store the message identifier in a "`msg_id`" column. See UPGRADE.txt or Upgrading to Asterisk 11 for more information.
- Added VoiceMailPlayMsg application. This application will play a single voicemail message from a mailbox. The result of the application, `SUCCESS` or `FAILED`, is stored in the channel variable `VOICEMAIL_PLAYBACKSTATUS`.

Functions

- Hangup handlers can be attached to channels using the CHANNEL function. Hangup handlers will run when the channel is hung up similar to the **h** extension. The `hangup_handler_push` option will push a Gosub compatible location in the dialplan onto the channel's hangup handler stack. The `hangup_handler_pop` option will remove the last added location, and optionally replace it with a new Gosub compatible location. The `hangup_handler_wipe` option will remove all locations on the stack, and optionally add a new location.
- The expression parser now recognizes the `ABS()` absolute value function, which will convert negative floating point values to positive values.
- FAXOPT(faxdetect) will enable a generic fax detect framehook for dialplan control of faxdetect.
- Addition of the VM_INFO function that can be used to retrieve voicemail user information, such as the email address and full name. The MAILBOX_EXISTS dialplan function has been deprecated in favour of VM_INFO.
- The REDIRECTING function now supports the redirecting original party id and reason.
- Two new functions have been added: FEATURE and FEATUREMAP. FEATURE lets you set some of the configuration options from the `[general]` section of *features.conf* on a per-channel basis. FEATUREMAP lets you customize the key sequence used to activate built-in features, such as `blindxfer`, and `automon`.
- MESSAGE(from) for incoming SIP messages now returns `"display-name" <uri>` instead of simply the uri. This is the format that MessageSend can use in the from parameter for outgoing SIP messages.
- Added the PRESENCE_STATE function. This allows retrieving presence state information from any presence state provider. It also allows setting presence state information from a CustomPresence presence state provider. See AMI/CLI changes for related commands.
- Added the AMI_CLIENT function to make manager account attributes available to the dialplan. It currently supports returning the current number of active sessions for a given account.

Channel Drivers

chan_local

- Added a manager event LocalBridge for local channel call bridges between the two pseudo-channels created.

chan_dahdi

- Added `dialtone_detect` option for analog ports to disconnect incoming calls when dialtone is detected.
- Added option `colp_send` to send ISDN connected line information. Allowed settings are `block`, to not send any connected line information; `connect`, to send connected line information on initial connect; and `update`, to send information on any update during a call. Default is `update`.
- Add options `namedcallgroup` and `namedpickupgroup` to support installations where a higher number of groups (>64) is required.

chan_motif

- A new channel driver named chan_motif has been added which provides support for Google Talk and Jingle in a single channel driver. This new channel driver includes support for both audio and video, RFC2833 DTMF, all codecs supported by Asterisk, hold, unhold, and ringing notification. It is also compliant with the current Jingle specification, current Google Jingle specification, and the original Google Talk protocol.

> For more information on chan_motif, please see the updated calling using Google page.

chan_ooh323

- Added NAT support for RTP. Setting in config is `nat`, which can be set globally and overriden on a peer by peer basis.
- Direct media functionality has been added. Options in config are: `directmedia` (`directrtp`) and `directrtpsetup` (`earlydirect`)
- ChannelUpdate events now contain a `CallRef` header.

chan_sip

- Asterisk will no longer substitute CID number for CID name in the display name field if CID number exists without a CID name. This change improves compatibility with certain device features such as Avaya IP500's directory lookup service.
- A new setting for `autocreatepeer` (`autocreatepeer=persistent`) allows peers created using that setting to not be removed during SIP reload.
- Added settings `recordonfeature` and `recordofffeature`. When receiving an INFO request with a "`Record:`" header, this will turn the requested feature on/off. Allowed values are '`automon`', '`automixmon`', and blank to disable. Note that dynamic features must be enabled and configured properly on the requesting channel for this to function properly.
- Add support to realtime for the '`callbackextension`' option.
- When multiple peers exist with the same address, but differing `callbackextension` options, incoming requests that are matched by address will be matched to the peer with the matching `callbackextension` if it is available.
- Two new NAT options, `auto_force_rport` and `auto_comedia`, have been added which set the `force_rport` and `comedia` options automatically if Asterisk detects that an incoming SIP request crossed a NAT after being sent by the remote endpoint.
- NAT settings are now a combinable list of options. The equivalent of the deprecated `nat=yes` is `nat=force_rport,comedia`. `nat=no` behaves as before.
- Add an option `send_diversion` which can be disabled to prevent diversion headers from automatically being added to INVITE requests.
- Add support for lightweight NAT keepalive. If enabled a blank packet will be sent to the remote host at a given interval to keep the NAT mapping open. This can be enabled using the `keepalive` configuration option.
- Add option '`tonezone`' to specify country code for indications. This option can be set both globally and overridden for specific peers.
- The SIP Security Events Framework now supports IPv6.
- Add a new setting for directmedia, '`outgoing`', to alleviate INVITE glares between multiple user agents. When set, for directmedia reinvites, Asterisk will not send an immediate reinvite on an incoming call leg. This option is useful when peered with another SIP user agent that is known to send immediate direct media reinvites upon call establishment.
- Add support for WebSocket transport. This can be configured using '`ws`' or '`wss`' as the transport.
- Add options `subminexpiry` and `submaxexpiry` to set limits of subscription timer independently from registration timer settings. The setting of the registration timer limits still is done by options `minexpiry`, `maxexpiry` and `defaultexpiry`. For backwards compatibility the setting of `minexpiry` and `maxexpiry` also is used to configure the subscription timer limits if `subminexpiry` and `submaxexpiry` are not set in sip.conf.
- Set registration timer limits to default values when reloading sip configuration and values are not set by configuration.
- Add options `namedcallgroup` and `namedpickupgroup` to support installations where a higher number of groups (>64) is required.
- When a MESSAGE request is received, the address the request was received from is now saved in the `SIP_RECVADDR` variable.
- Add ANI2/OLI parsing for SIP. The "From" header in INVITE requests is now parsed for the presence of "isup-oli", "ss7-oli", or "oli" tags. If present, the ANI2/OLI information is set on the channel, which can be retrieved using the CALLERID function.
- Peers can now be configured to support negotiation of ICE candidates using the setting `icesupport`. See res_rtp_asterisk changes for more information.
- Added support for format attribute negotiation. See the Codecs changes for more information.
- Extra headers specified with SIPAddHeader are sent with the REFER message when using Transfer application. See `refer_addheaders` in *sip.conf.sample*.

chan_skinny

- Added skinny version 17 protocol support.

chan_unistim

- Added ability to use multiple lines for a single phone. This allows multiple calls to occur on a single phone, using callwaiting and switching between calls.
- Added option '`sharpdial`' allowing end dialing by pressing # key
- Added option '`interdigit_timer`' to control phone dial timeout
- Added options '`cwstyle`', '`cwvolume`' controlling `callwaiting` appearance
- Added global '`debug`' option, that enables debug in channel driver
- Added ability to translate on-screen menu in multiple languages. Tested on Russian languages. Supported encodings: ISO 8859-1, ISO 8859-2, ISO 8859-4, ISO 8859-5, ISO 2022-JP. Language controlled by '`language`' and on-screen menu of phone
- In addition to English added French and Russian languages for on-screen menus
- Reworked dialing number input: added dialing by timeout, immediate dial on on dialplan compare, phone number length now not limited by screen size
- Added ability to pickup a call using *features.conf* defined value and on-screen key

> For more information on the chan_unistim changes, please see Unistim channel improvements

chan_mISDN:

- Add options `namedcallgroup` and `namedpickupgroup` to support installations where a higher number of groups (>64) is required.

Core

- The minimum DTMF duration can now be configured in *asterisk.conf* as `mindtmfduration`. The default value is (as before) set to 80 ms. Previously this option was set to a hard coded value in the source code.
- Named ACLs can now be specified in *acl.conf* and used in configurations that use ACLs. As a general rule, if some derivative of '`permit`' or '`deny`' is used to specify an ACL, a similar form of '`acl`' will add a named ACL to the working ACL. In addition, some CLI commands have been added to provide show information and allow for module reloading - see CLI Changes.
- Rules in ACLs (specified using '`permit`' and '`deny`') can now contain multiple items (separated by commas), and items in the rule can be negated by prefixing them with '`!`'. This simplifies Asterisk Realtime configurations, since it is no longer necessray to control the order that the '`permit`' and '`deny`' columns are returned from queries.
- DUNDi now allows the built in variables ${NUMBER}, ${IPADDR} and ${SECRET} to be used within the dynamic weight attribute when specifying a mapping.
- CEL backends can now be configured to show "`USER_DEFINED`" in the `EventName` header, instead of putting the user defined event name there. When enabled the `UserDefType` header is added for user defined events. This feature is enabled with the setting `show_user_defined`.
- Macro has been deprecated in favor of GoSub. For redirecting and connected line purposes use the following variables instead of their macro equivalents:
    - `REDIRECTING_SEND_SUB`
    - `REDIRECTING_SEND_SUB_ARGS`
    - `CONNECTED_LINE_SEND_SUB`
    - `CONNECTED_LINE_SEND_SUB_ARGS`

    For CCSS, use `cc_callback_sub` instead of `cc_callback_macro` in channel configurations.
- Asterisk can now use a system-provided NetBSD editline library (libedit) if it is available.
- Call files now support the `early_media` option to connect with an outgoing extension when early media is received.

AGI

- A new channel variable, `AGIEXITONHANGUP`, has been added which allows Asterisk to behave like it did in Asterisk 1.4 and earlier where the AGI application would exit immediately after a channel hangup is detected.
- IPv6 addresses are now supported when using FastAGI (agi://). Hostnames are resolved and each address is attempted in turn until one

succeeds or all fail.

AMI (Asterisk Manager Interface)

- The Originate action now has an option EarlyMedia that enables the call to bridge when we get early media in the call. Previously, early media was disregarded always when originating calls using AMI.
- Added setvar= option to manager accounts (much like *sip.conf*)
- Originate now generates an error response if the extension given is not found in the dialplan.
- MixMonitor will now show IDs associated with the MixMonitor upon creating them if the **i(variable)** option is used. StopMixMonitor will accept MixMonitorID as an option to close specific MixMonitors.
- The SIPshowpeer manager action response field SIP-Forcerport has been updated to include information about peers configured with nat=auto_force_rport by returning "**A**" if auto_force_rport is set and nat is detected, and "**a**" if it is set and nat is not detected. "**Y**" and "**N**" are still returned if auto_force_rport is not enabled.
- Added SIPpeerstatus manager command which will generate PeerStatus events similar to the existing PeerStatus events found in chan_sip on demand.
- Hangup can now take a regular expression as the Channel option. If you want to hangup multiple channels, use /regex/ as the Channel option. Existing behavior to hanging up a single channel is unchanged, but if you pass a regex, the manager will send you a list of channels back that were hung up.
- Support for IPv6 addresses has been added.
- AMI Events can now be documented in the Asterisk source. Note that AMI event documentation is only generated when Asterisk is compiled using make full. See the CLI section for commands to display AMI event information.
- The AMI Hangup event now includes the AccountCode header so you can easily correlate with AMI Newchannel events.
- The QueueMemberStatus, QueueMemberAdded, and QueueMember events now include the StateInterface of the queue member.
- Added AMI event SessionTimeout in the Call category that is issued when a call is terminated due to either RTP stream inactivity or SIP session timer expiration.
- CEL events can now contain a user defined header UserDefType. See core changes for more information.
- OOH323 ChannelUpdate events now contain a CallRef header.
- Added PresenceState command. This command will report the presence state for the given presence provider.
- Added Parkinglots command. This will list all parking lots as a series of AMI Parkinglot events.
- Added MessageSend command. This behaves in the same manner as the MessageSend application, and is a technolgoy agnostic mechanism to send out of call text messages.
- Added "message" class authorization. This grants an account permission to send out of call messages. Write-only. See *manager.conf.sample*.

CLI

- The dialplan add include command has been modified to create context a context if one does not already exist. For instance, dialplan add include foo into bar will create context bar if it does not already exist.
- A dialplan remove context command has been added to remove a context from the dialplan
- The mixmonitor list <channel> command will now show MixMonitor ID, and the filenames of all running mixmonitors on a channel.
- The debug level of pri set debug is now a bitmask ranging from 0 to 15 if numeric instead of 0, 1, or 2.
- stun show status command will show a table describing how the STUN client is behaving.
- acl show [named acl] will show information regarding a Named ACL. The acl module can be reloaded with reload acl.
- Added CLI command to display AMI event information - manager show events, which shows a list of all known and documented AMI events, and manager show event [event name], which shows detailed information about a specific AMI event.
- The result of the CLI command queue show now includes the state interface information of the queue member.
- The command core set verbose will now set a separate level of logging for each remote console without affecting any other console.
- Added command cdr show pgsql status to check connection status
- sip show channel will now display the complete route set.
- Added presencestate list command. This command will list all custom presence states that have been set by using the PRESENCE_STATE dialplan function.
- Added presencestate change <entity> <state>[,<subtype>[,message[,options]]] command. This changes a custom presence to a new state.

Codecs

- Codec lists may now be modified by the `'!'` character, to allow succinct specification of a list of codecs allowed and disallowed, without the requirement to use two different keywords. For example, to specify all codecs except g729 and g723, one need only specify `allow=all,!g729,!g723`.
- Add support for parsing SDP attributes, generating SDP attributes, and passing it through. This support includes codecs such as H.263, H.264, SILK, and CELT. You are able to set up a call and have attribute information pass. This should help considerably with video calls.
- The iLBC codec can now use a system-provided iLBC library if one is installed, just like the GSM codec.

DUNDi changes

- Added CLI commands `dundi show hints` and `dundi show cache` which will list DUNDi '`DONTASK`' hints in the cache and list all DUNDi cache entires respectively.

Logging

- Asterisk version and build information is now logged at the beginning of a log file.
- Threads belonging to a particular call are now linked with callids which get added to any log messages produced by those threads. Log messages can now be easily identified as involved with a certain call by looking at their call id. Call ids may also be attached to log messages for just about any case where it can be determined to be related to a particular call.
- Each logging destination and console now have an independent notion of the current verbosity level. *logger.conf* now allows an optional argument to the '`verbose`' specifier, indicating the level of verbosity sent to that particular logging destination. Additionally, remote consoles now each have their own verbosity level. The command `core set verbose` will now set a separate level for each remote console without affecting any other console.

Music On Hold

- Added `announcement` option which will play at the start of MOH and between songs in modes of MOH that can detect transitions between songs, e.g., files, mp3, etc.

Parking

- New per parking lot options: `comebackcontext` and `comebackdialtime`. See *features.conf.sample* for more details.
- Channel variable `PARKER` is now set when `comebacktoorigin` is disabled in a parking lot.
- Channel variable `PARKEDCALL` is now set with the name of the parking lot when a timeout occurs.

CDRs

CDR Postgresql Driver

- Added command `cdr show pgsql status` to check connection status

CDR Adaptive ODBC Driver

- Added schema option for databases that support specifying a schema.

Resource Modules

Calendars

- A `CALENDAR_SUCCESS=1/0` channel variable is now set to show whether or not CALENDAR_WRITE has completed successfully.

res_rtp_asterisk

- A new option, `probation`, has been added to *rtp.conf*. RTP processing in `strictrtp` mode can now require more than 1 packet to exit

learning mode with a new source (and by default requires 4). The probation option allows the user to change the required number of packets in sequence to any desired value. Use a value of 1 to essentially restore the old behavior. Also, with `strictrtp` enabled, Asterisk will now drop all packets until learning mode has successfully exited. These changes are based on how pjmedia handles media sources and source changes.
- Add support for ICE/STUN/TURN in res_rtp_asterisk. This option can be enabled using the `icesupport` setting; it is disabled by default. A variety of other settings have been introduced to configure STUN/TURN connections.

res_corosync

- A new module, res_corosync, has been introduced. This module uses the Corosync cluster enginer to allow a local cluster of Asterisk servers to both Message Waiting Indication (MWI) and/or Device State (presence) information. This module is very similar to, and is a replacement for the res_ais module that was in previous releases of Asterisk.

res_xmpp

- This module adds a cleaned up, drop-in replacement for res_jabber called res_xmpp. This provides the same externally facing functionality but is implemented differently internally. res_jabber has been deprecated in favor of res_xmpp; please see Upgrading to Asterisk 11 or the UPGRADE.txt file for more information.

Scripts

- The *safe_asterisk* script has been updated to allow several of its parameters to be set from environment variables. This also enables a custom run directory of Asterisk to be specified, instead of defaulting to /tmp.
- The *live_ast* script will now look for the `LIVE_AST_BASE_DIR` variable and use its value to determine the directory to assume is the top-level directory of the source tree. If the variable is not set, it defaults to the current behavior and uses the current working directory.

Upgrading to Asterisk 11

The following are changes made in Asterisk 11 that may affect your configuration when upgrading.

Applications

Voicemail

All voicemails now have a `msg_id` included in their metadata which uniquely identifies a message. For users of file system and IMAP storage of voicemail, this should be transparent. For users of ODBC, you will need to add a `msg_id` column to your voice mail messages table. This should be a string capable of holding at least 32 characters.

All messages created in old Asterisk installations will have a `msg_id` added to them when required. This operation should be transparent as well.

MeetMe

The '**c**' option (announce user count) will now work even if the '**q**' (quiet) option is enabled.

FollowMe

Answered outgoing calls no longer get cut off when the next step is started. You now have until the last step times out to decide if you want to accept the call or not before being disconnected.

Channel Drivers

chan_gtalk

chan_gtalk has been deprecated in favour of the chan_motif channel driver. It is recommended that users switch to using it as it is a core supported module.

Please see Calling Using Google for more information.

chan_jingle

chan_jingle has been deprecated in favour of the chan_motif channel driver. It is recommended that users switch to using it as it is a core supported module.

Please see Calling Using Google for more information.

chan_sip

- SIP_CAUSE is now deprecated. It has been modified to use the same mechanism as the HANGUPCAUSE function. Behaviour should not change, but performance should be vastly improved. The HANGUPCAUSE function should now be used instead of SIP_CAUSE. Because of this, the *storesipcause* option in sip.conf is also deprecated. Please see Hangup Cause for more information.
- ICE support has been added and can be enabled using the `icesupport` configuration option. Some endpoints may have problems with the ICE candidates within the SDP. Symptoms of this include one way media or no media flow.

chan_unistim

Due to massive updates in chan_unistim phone keys functions and on-screen information has changed.

Please see the Unistim Channel Improvements for more information.

Resource Modules

res_ais

Users of res_ais in versions of Asterisk prior to Asterisk 11 must change to use the res_corosync module, instead. OpenAIS is deprecated, but Corosync is still actively developed and maintained. Corosync came out of the OpenAIS project.

res_jabber

This module has been deprecated in favor of the res_xmpp module. The res_xmpp module is backwards compatible with the res_jabber configuration file, dialplan functions, and AMI actions. The old CLI commands can also be made available using the res_clialiases template for Asterisk 11.

Dialplan Functions

- MAILBOX_EXISTS has been deprecated. Use VM_INFO with the '`exists`' parameter instead.
- Macro has been deprecated in favour of Gosub. While Macro has been deprecated for some time, in Asterisk 11 all internal functions that relied on Macros have been transitioned to use GoSub. For redirecting and connected line purposes use the following variables instead of their macro equivalents:
    - `REDIRECTING_SEND_SUB`
    - `REDIRECTING_SEND_SUB_ARGS`
    - `CONNECTED_LINE_SEND_SUB`
    - `CONNECTED_LINE_SEND_SUB_ARGS`
- The HANGUPCAUSE and HANGUPCAUSE_KEYS functions have been introduced to provide a replacement for the SIP_CAUSE hash. The HangupCauseClear application has also been introduced to remove this data from the channel when necessary. Please see
- ENUM query functions now return a count of -1 on lookup error to differentiate between a failed query and a successful query with 0 results matching the specified type.

Core

Logging

The `verbose` setting in *logger.conf* now takes an optional argument specifying the verbosity level for each logging destination. The default, if not otherwise specified, is a verbosity of 3.

Verbose/Debug setting changes

Asterisk 11 splits verbose logging levels from Asterisk and individual remote console sessions. This change was put in place due to the potential awkwardness of having verbosity changes take place in multiple remote console sessions on account of just one of those sessions requesting a verbosity change. Under the new system, core set verbosity is intercepted by the remote console and sets the remote CLI verbosity without ever actually contacting the Asterisk service involved.

There have been some complications introduced with this feature. The command override performed by the remote console is performed over the command string and is naive to the fact that Asterisk can call the verbosity setting CLI command by other means, notably by setting an alias. If the command is aliased via cli_aliases.conf, the alias will be created at the startup of the Asterisk service and since the aliased command can be formatted in any manner, the core set verbose command that is internal to Asterisk will still be accessible to remote consoles. This behavior is confusing and quite likely to trip up users.

Example of how it functions normally:

1. Asterisk service is started
2. user connects to Asterisk via remote console
3. Enters command: core set verbose 5

```
*CLI> core set verbose 5
Set remote console verbosity to 5
```

1. At this point, verbosity for the remote console session is at 5.

If an alias is made in cli_aliases.conf: foo bar=core set verbose

1. Asterisk service is started
2. user connects to Asterisk via remote console
3. Enters command: foo bar 5

```
*CLI> foo bar 5
Verbosity was 3 and is now 5
```

1. The user has been told verbosity is now 5, but in reality what he will see is still whatever the remote console's initial verbose level was and the internal verbosity is what has been changed. This is because foo bar 5 executed the internal version of 'core set verbose 5'

A simple workaround for this right now is to just not alias 'core set verbose'.

AMI

- DBdeltree now correctly returns an error when 0 rows are deleted just as the DBdel action does.
- The IAX2 PeerStatus event now sends a `Port` header. In Asterisk 10, this was erroneously being sent as a `Post` header.

CCSS

Macro is deprecated. Use `cc_callback_sub` instead of `cc_callback_macro` in channel configurations.

Parking

- The `comebacktoorigin` setting must now be set per parking lot. The setting in the general section will not be applied automatically to each parking lot.
- The `BLINDTRANSFER` channel variable is deleted from a channel when it is bridged to prevent subtle bugs in the parking feature. The channel variable is used by Asterisk internally for the Park application to work properly. If you were using it for your own purposes, copy it to your own channel variable before the channel is bridged.

users.conf

A defined user with `hasvoicemail=yes` now finally uses a Gosub to `stdexten` as documented in *extensions.conf.sample* since v1.6.0 instead of a Macro as documented in v1.4. Set the asterisk.conf `stdexten=macro` parameter to invoke the `stdexten` the old way.

Asterisk WebRTC Support

> ⚠ If you would like to test Asterisk with WebRTC you can now use the latest shipping Chrome. As VP8 is not presently a supported codec for passthrough video will not work, but audio will!

Background

WebRTC/rtcweb is an effort to bring a defined API to JavaScript developers that allows them to venture into the world of real time communications. This may be a click-to-call system or a "softphone" with both delivered as a webpage. No plug-ins are required and as this is a defined specification it can be used across different browsers where supported.

Asterisk has had support for WebRTC since version 11. A res_http_websocket module has been created which allows the JavaScript developers to interact and communicate with Asterisk. Support for WebSocket as a transport has been added to chan_sip to allow SIP to be used as the signaling protocol. ICE, STUN, and TURN support has been added to res_rtp_asterisk to allow clients behind NAT to better communicate with Asterisk. SRTP support was added in a previous version but it is also a requirement of WebRTC.

Browser Support

The latest information about browser support is available at http://en.wikipedia.org/wiki/WebRTC

SRTP

Secure media is a requirement of WebRTC and as a result SRTP must be available. In order for Asterisk to build SRTP support the libsrtp library and development headers must be available. This can be installed using the distribution's package management system or from source. Failure to do this will result in the media offers being rejected.

Configuring res_http_websocket

The built-in Asterisk HTTP server is used to provide the WebSocket support. This can be enabled using the following in the general section of the http.conf configuration file.

```
enabled=yes
```

If you would like to change the port from the default value of 8088 this can also be done in the general section.

```
bindport=8088
```

The res_http_websocket must also be built and loaded by Asterisk. For most individuals this is done by default.

> ⚠ Ensure that res_http_websocket.so is selected in menuselect prior to building Asterisk. Also ensure that res_http_websocket.so is loaded prior to chan_sip.so if you are not using autoload in modules.conf

The secure calling tutorial viewable at https://wiki.asterisk.org/wiki/display/AST/Secure+Calling+Tutorial can be used as a basis to configure the built-in HTTP server with HTTPS (and secure WebSocket) support.

Configuring chan_sip

All configuration occurs in sip.conf, or a configuration file included by it.

To allow a peer, user, or friend access using the WebSocket transport it must be added to their transport options like the following.

```
transport=ws,wss
```

To restrict access to clients using only an HTTPS connection allow the 'wss' transport only.

The WebRTC standard has selected AVPF as the audio video profile to use for media streams. This is **not** the default profile in use by chan_sip. As a result the following must be added to the peer, user, or friend.

```
avpf=yes
```

This will cause AVPF and SAVPF to be used and the media streams to be accepted.

> Asterisk 11.0.0-beta1 has an issue in it where registering over WebSocket may not work properly. The work around is to use a newer version of Asterisk that has been released, or check out the Asterisk 11 branch from SVN. You can also set
>
> ```
> nat=yes,force_rport
> ```
>
> on the peer, user, or friend to work around the issue.
>
> The issue report for this problem is viewable at https://issues.asterisk.org/jira/browse/ASTERISK-20238

As media encryption is a requirement of rtcweb the following must be added to the peer, user, or friend to enable it.

```
encryption=yes
```

Using WebSocket

The res_http_websocket module provides WebSocket at the /ws sub-directory only. This is an implementation specific detail. Some JavaScript libraries may need to be changed slightly to explicitly use the sub-directory. Symptoms of using the incorrect URL are a 404 Not Found response from the Asterisk HTTP server.

JavaScript Libraries

1. JsSIP - Provides a WebRTC compatible JavaScript SIP library, demo is available here for download.
2. sipml5 - Provides a WebRTC compatible JavaScript SIP library.

Issues

All SIP responses are sent from Asterisk to the client.

HTTP Response: 404 Not Found

The JavaScript library is using an incorrect URL for WebSocket access. The URL must use the /ws sub-directory.

SIP Response: 400 Bad Request received over SIP when registering using WebSocket

The version of chan_sip in use has a bug when registering. Update to a newer version.

SIP Response: 488 Not acceptable here received over SIP when placing a call to Asterisk

You have not enabled AVPF support in the peer, user, or friend entry using "avpf=yes" or have not allowed a codec that is supported by the caller.

Call Identifier Logging

Overview

Call ID Logging (which has nothing to do with caller ID) is a new feature of Asterisk 11 intended to help administrators and support givers to more quickly understand problems that occur during the course of calls. Channels are now bound to call identifiers which can be shared among a number of channels, threads, and other consumers.

Usage

No configuration is needed to take advantage of this feature. Asterisk 11 will simply apply an additional bracketed tag to all log messages generated by a thread with a call ID bound or to any log messages specially written to use call identifiers. For example:

- Asterisk receives a request for a non existent extension from SIP/gold
- The following log message is displayed:
  ```
  [Oct 18 10:26:11] NOTICE[27538][C-00000000]: chan_sip.c:25107 handle_request_invite:
  Call from 'gold' (10.24.22.201:5060) to extension '645613' rejected because
  extension not found in context 'default'.
  ```

C-00000000 is the call identifier associated with this attempted call. All call identifiers are represented as C-XXXXXXXX where XXXXXXXX is an 8 digit hexadecimal value much like what you will see with SIP and local channel names.

Aside from log messages, call identifiers are also shown in the output for the 'core show channel <channel name>' command.

Transfers

Transfers can be a little tricky to follow with the call ID logging feature. As a general rule, an attended transfer will always result in a new call ID being made because a separate call must occur between the party that initiates the transfer and whatever extension is going to receive it. Once the attended transfer is completed, the channel that was transferred will use the Call ID created when the transferrer called the recipient.

Blind transfers are slightly more variable. If a SIP peer 'peer1' calls another SIP peer 'peer2' via the dial application and peer2 blind transfers peer1 elsewhere, the call ID will persist. If on the other hand, peer1 blind transfers peer2 at this point a new call ID will be created. When peer1 transfers peer2, peer2 has a new channel created which enters the PBX for the first time, so it creates a new call ID. When peer1 is transferred, it simply resumes running PBX, so the call is still considered the same call. By setting the debug level to 3 for the channel internal API (channel_internal_api.c), all call ID settings for every channel will be logged and this may be able to help when trying to keep track of calls through multiple transfers.

Call Pickup

1. Overview

Call pickup allows you to answer an incoming call from another phone.

Requesting to pickup a call is done by two basic methods.
1) by dialplan using the Pickup or PickupChan applications.
2) by dialing the pickupexten configured in features.conf.

Which calls can be picked up is determined by configuration and dialplan.

2. Dialplan Applications and Functions

2.1. Pickup Application

The Pickup application has three ways to select calls for pickup.
1) With no parameters, Pickup selects calls using the numeric and named call groups like the pickupexten.
2) Extension with PICKUPMARK, Pickup selects calls with the PICKUPMARK channel variable matching the extension.
3) Extension with or without a context, Pickup selects calls with the matching extension and context.

2.2. PickupChan Application

The PickupChan application tries to pickup the specified channels given to it.

2.3. CHANNEL Function

The CHANNEL function allows the pickup groups set on a channel to be changed from the defaults set by the channel driver when the channel was created.

2.3.1. callgroup/namedcallgroup

The CHANNEL(callgroup) option specifies which numeric pickup groups that this channel is a member.

```
same => n,Set(CHANNEL(callgroup)=1,5-7)
```

The CHANNEL(namedcallgroup) option specifies which named pickup groups that this channel is a member.

```
same => n,Set(CHANNEL(namedcallgroup)=engineering,sales)
```

NOTES

- For this option to be effective, you must set it on the outgoing channel.
- You can use the setvar option available with several channel driver configuration files to set the pickup groups.
- You can use a pre-dial handler.

2.3.2. pickupgroup/namedpickupgroup

The CHANNEL(pickupgroup) option specifies which numeric pickup groups this channel can pickup.

```
same => n,Set(CHANNEL(pickupgroup)=1,6-8)
```

The CHANNEL(namedpickupgroup) option specifies which named pickup groups this channel can pickup.

```
same => n,Set(CHANNEL(namedpickupgroup)=engineering,sales)
```

NOTES

- For this option to be effective, you must set it on the channel before executing the Pickup application or calling the pickupexten.
- You can use the setvar option available with several channel driver configuration files to set the pickup groups.

3. Configuration Options

The pickupexten request method selects calls using the numeric and named call groups. The ringing channels have the callgroup assigned when the channel is created by the channel driver or set by the CHANNEL(callgroup) or CHANNEL(namedcallgroup) dialplan function.

Calls picked up using pickupexten can hear an optional sound file for success and failure.

The current channel drivers that support calling the pickupexten to pickup a call are: chan_dahdi/analog, chan_mgcp, chan_misdn, chan_sip, and chan_unistim.

features.conf

```
pickupexten = *8              ; Configure the pickup extension. (default is *8)
pickupsound = beep            ; to indicate a successful pickup (default: no sound)
pickupfailsound = beeperr     ; to indicate that the pickup failed (default: no sound)
```

3.1. Numeric call pickup groups

A numeric callgroup and pickupgroup can be set to a comma separated list of ranges (e.g., 1-4) or numbers that can have a value of 0 to 63. There can be a maximum of 64 numeric groups.

Syntax

```
callgroup=[number[-number][,number[-number][,...]]]
pickupgroup=[number[-number][,number[-number][,...]]]
```

callgroup - specifies which numeric pickup groups that this channel is a member.
pickupgroup - specifies which numeric pickup groups this channel can pickup.

chan_dahdi.conf/analog, misdn.conf, mgcp.conf, sip.conf, unistim.conf

```
callgroup=1,5-7
pickupgroup=1
```

3.2. Named call pickup groups

A named callgroup and pickupgroup can be set to a comma separated list of case sensitive name strings. The number of named groups is unlimited. The number of named groups you can specify at once is limited by the line length supported.

Syntax

```
namedcallgroup=[name[,name[,...]]]
namedpickupgroup=[name[,name[,...]]]
```

namedcallgroup - specifies which named pickup groups that this channel is a member.
namedpickupgroup - specifies which named pickup groups this channel can pickup.

chan_dahdi.conf/analog, misdn.conf, sip.conf

```
namedcallgroup=engineering,sales,netgroup,protgroup
namedpickupgroup=sales
```

NOTES

- You can use named pickup groups in parallel with numeric pickup groups. For example, the named pickup group '4' is not the same as the numeric pickup group '4'.

- Named pickup groups are new with Asterisk 11.

Dynamic DTMF Features

The FEATURE and FEATUREMAP dialplan functions allow you to set some features.conf options on a per channel basis.

To see what options are currently supported, look at the FEATURE and FEATUREMAP function descriptions.

Set the parking time of this channel to be 100 seconds if it is parked.

```
exten => s,1,Set(FEATURE(parkingtime)=100)
same => n,Dial(SIP/100)
same => n,Hangup()
```

Set the DTMF sequence for attended transfer on this channel to *9.

```
exten => s,1,Set(FEATUREMAP(atxfer)=*9)
same => n,Dial(SIP/100,,T)
same => n,Hangup()
```

Hangup Cause

Overview

The Hangup Cause family of functions and dialplan applications allow for inspection of the hangup cause codes for each channel involved in a call. This allows a dialplan writer to determine, for each channel, who hung up and for what reason(s). Note that this extends the functionality available in the HANGUPCAUSE channel variable, by allowing a calling channel to inspect all called channel's hangup causes in a variety of dialling situations.

Note that this feature replaces the technology specific mechanism of using the MASTER_CHANNEL function to access a SIP channel's SIP_CAUSE, as well as extends similar functionality to a variety of other channel drivers.

Dialplan Functions and Applications

HANGUPCAUSE_KEYS

Used to obtain a comma separated list of all channels for which hangup causes are available.

Example

The following example shows one way of accessing the channels that have hangup cause related information after a Dial has completed. In this particular example, a parallel dial occurs to both *SIP/foo* and *SIP/bar*. A hangup handler has been attached to the calling channel, which executes the subroutine at **handler,s,1** when the channel is hung up. This queries the HANGUPCAUSE_KEYS function for the channels with hangup cause information and prints the information as a Verbose message. On the CLI, this would look something like:

```
Channels with hangup cause information: SIP/bar-00000002,SIP/foo-00000001
```

```
[default]

exten => s,1,NoOp()
same => n,Set(CHANNEL(hangup_handler_push)=handler,s,1)
same => n,Dial(SIP/foo&SIP/bar,10)
same => n,Hangup()

[handler]

same => s,1,NoOp()
same => n,Set(HANGUPCAUSE_STRING=${HANGUPCAUSE_KEYS()})
same => n,Verbose(0, Channels with hangup cause information: ${HANGUPCAUSE_STRING})
same => n,Return()
```

HANGUPCAUSE

Used to obtain hangup cause information for a specific channel. For a given channel, there are two sources of hangup cause information:

1. The channel technology specific hangup cause information
2. A text description of the Asterisk specific hangup cause

Note that in some cases, the hangup causes returned may not be reflected in Hangup Cause Mappings. For example, if a Dial to a SIP UA is cancelled by Asterisk, the SIP UA may not have returned any final responses to Asterisk. In these cases, the last known technology code will be returned by the function.

Example

This example illustrates obtaining hangup cause information for a parallel dial to *SIP/foo* and *SIP/bar*. A hangup handler has been attached to the calling channel, which executes the subroutine at **handler,s,1** when the channel is hung up. This queries the hangup cause information using the HANGUPCAUSE_KEYS function and the HANGUPCAUSE function. The channels returned from HANGUPCAUSE_KEYS are parsed out, and each is queried for their hangup cause information. The technology specific cause code as well as the Asterisk cause code are printed to the CLI.

```
[default]

exten => s,1,NoOp()
same => n,Set(CHANNEL(hangup_handler_push)=handler,s,1)
same => n,Dial(SIP/foo&SIP/bar,10)
same => n,Hangup()

[handler]

exten => s,1,NoOp()

same => n,Set(HANGUPCAUSE_STRING=${HANGUPCAUSE_KEYS()})

; start loop
same => n(hu_begin),NoOp()

; check exit condition (no more array to check)
same => n,GotoIf($[${LEN(${HANGUPCAUSE_STRING})}=0]?hu_exit)

; pull the next item
same => n,Set(ARRAY(item)=${HANGUPCAUSE_STRING})
same => n,Set(HANGUPCAUSE_STRING=${HANGUPCAUSE_STRING:${LEN(${item})}})

; display the channel name and cause codes
same => n,Verbose(0, Got Channel ID ${item} with Technology Cause Code
${HANGUPCAUSE(${item},tech)}, Asterisk Cause Code ${HANGUPCAUSE(${item},ast)})

; check exit condition (no more array to check)
same => n,GotoIf($[${LEN(${HANGUPCAUSE_STRING})}=0]?hu_exit)

; we still have entries to process, so strip the leading comma
same => n,Set(HANGUPCAUSE_STRING=${HANGUPCAUSE_STRING:1})

; go back to the beginning of the loop
same => n,Goto(hu_begin)

same => n(hu_exit),NoOp()
same => n,Return()
```

HangupCauseClear

Used to remove all hangup cause information currently stored.

Example

The following example clears the hangup cause information from the channel if *SIP/foo* fails to answer and execution continues in the dialplan. The hangup handler attached to the channel will thus only report the the name of the last channel dialled.

```
exten => s,1,NoOp()
same => n,Set(CHANNEL(hangup_handler_push)=handler,s,1)
same => n,Dial(SIP/foo,10)
same => n,HangupCauseClear()
same => n,Dial(SIP/bar,10)
same => n,Hangup()

[handler]

same => s,1,NoOp()
same => n,Set(HANGUPCAUSE_STRING=${HANGUPCAUSE_KEYS()})
same => n,Verbose(0, Channels with hangup cause information: ${HANGUPCAUSE_STRING})
same => n,Return()
```


Hangup Cause Mappings

Asterisk Hangup Cause Code Mappings

Asterisk Value	ISDN Cause codes (Q.850 & Q.931 unless specified)	MFC/R2	SIP/PJSIP	Motif
AST_CAUSE_NOT_DEFINED	Cause not defined	OR2_CAUSE_UNSPECIFIED		
AST_CAUSE_UNALLOCATED	1. Unallocated (unassigned) number		404, 485, 604	
AST_CAUSE_NO_ROUTE_TRANSIT_NET	2. No route to specified transmit network			
AST_CAUSE_NO_ROUTE_DESTINATION	3. No route to destination		420	
AST_CAUSE_MISDIALLED_TRUNK_PREFIX	5. Misdialled trunk prefix (national use)			
AST_CAUSE_CHANNEL_UNACCEPTABLE	6. Channel unacceptable			
AST_CAUSE_CALL_AWARDED_DELIVERED	7. Call awarded and being delivered in an established channel			
AST_CAUSE_PRE_EMPTED	ISUP - 8. Preemption			
AST_CAUSE_NUMBER_PORTED_NOT_HERE	14. QoR: ported number			
AST_CAUSE_NORMAL_CLEARING	16. Normal Clearing	OR2_CAUSE_NORMAL_CLEARING		gone, success
AST_CAUSE_USER_BUSY	17. User busy	OR2_CAUSE_BUSY_NUMBER	486, 600	busy
AST_CAUSE_NO_USER_RESPONSE	18. No user responding		408	expired
AST_CAUSE_NO_ANSWER	19. No answer from user (user alerted)	OR2_CAUSE_NO_ANSWER	480, 483	
AST_CAUSE_SUBSCRIBER_ABSENT	20. Subscriber absent	OR2_CAUSE_UNALLOCATED_NUMBER		
AST_CAUSE_CALL_REJECTED	21. Call Rejected		401, 403, 407, 603	cancel, decline
AST_CAUSE_NUMBER_CHANGED	22. Number changed		410	
AST_CAUSE_REDIRECTED_TO_NEW_DESTINATION	23. Redirected to new destination			
AST_CAUSE_ANSWERED_ELSEWHERE	26. Non-selected user clearing (ASTERISK-15057)			

AST_CAUSE_DESTINATION_OUT_OF_ORDER	27. Destination out of order	OR2_CAUSE_OUT_OF_ORDER	502	
AST_CAUSE_INVALID_NUMBER_FORMAT	28. Invalid number format		484	
AST_CAUSE_FACILITY_REJECTED	29. Facility rejected		501	
AST_CAUSE_RESPONSE_TO_STATUS_ENQUIRY	30. Response to STATUS ENQUIRY			
AST_CAUSE_NORMAL_UNSPECIFIED	31. Normal, unspecified			
AST_CAUSE_NORMAL_CIRCUIT_CONGESTION	34. No circuit/channel available (Note that we've called this "Circuit/channel congestion" for a while which can cause confusion with code 42)	OR2_CAUSE_NETWORK_CONGESTION		general-error
AST_CAUSE_NETWORK_OUT_OF_ORDER	38. Network out of order		500	
AST_CAUSE_NORMAL_TEMPORARY_FAILURE	41. Temporary failure		409	
AST_CAUSE_SWITCH_CONGESTION	42. Switching equipment congestion		5xx	failed-application
AST_CAUSE_ACCESS_INFO_DISCARDED	43. Access information discarded			
AST_CAUSE_REQUESTED_CHAN_UNAVAIL	44. Requested circuit/channel not available			
AST_CAUSE_FACILITY_NOT_SUBSCRIBED	50. Requested facility not subscribed			
AST_CAUSE_OUTGOING_CALL_BARRED	52. Outgoing call barred			
AST_CAUSE_INCOMING_CALL_BARRED	54. Incoming call barred			
AST_CAUSE_BEARERCAPABILITY_NOTAUTH	57. Bearer capability not authorized			
AST_CAUSE_BEARERCAPABILITY_NOTAVAIL	58. Bearer capability not presently available		488, 606	incompatible-parameters, media-error, unsupported-applications
AST_CAUSE_BEARERCAPABILITY_NOTIMPL	65. Bearer capability not implemented			
AST_CAUSE_CHAN_NOT_IMPLEMENTED	66. Channel type not implemented			
AST_CAUSE_FACILITY_NOT_IMPLEMENTED	69. Requested facility not implemented			unsupported-transports
AST_CAUSE_INVALID_CALL_REFERENCE	81. Invalid call reference value			
AST_CAUSE_INCOMPATIBLE_DESTINATION	88. Incompatible destination			

AST_CAUSE_INVALID_MSG_UNSPECIFIED	95. Invalid message unspecified			
AST_CAUSE_MANDATORY_IE_MISSING	96. Mandatory information element is missing			
AST_CAUSE_MESSAGE_TYPE_NONEXIST	97. Message type non-existent or not implemented			
AST_CAUSE_WRONG_MESSAGE	98. Message not compatible with call state or message type non-existent or not implemented			
AST_CAUSE_IE_NONEXIST	99. Information element nonexistent or not implemented			
AST_CAUSE_INVALID_IE_CONTENTS	100. Invalid information element contents			
AST_CAUSE_WRONG_CALL_STATE	101. Message not compatible with call state			
AST_CAUSE_RECOVERY_ON_TIMER_EXPIRE	102. Recover on timer expiry		504	timeout
AST_CAUSE_MANDATORY_IE_LENGTH_ERROR	? Mandatory IE length error			
AST_CAUSE_PROTOCOL_ERROR	111. Protocol error, unspecified			failed-transport, security-error
AST_CAUSE_INTERWORKING	127. Interworking, unspecified		4xx, 505, 6xx	connectivity-error

Notes

- The hangup cause AST_CAUSE_NOT_DEFINED is not actually a Q.931 cause code, and is used to capture hangup causes that do not map cleanly to a Q.931 cause code.
- IAX2, ISDN, and SS7 are all subsets of the cause codes listed above.
- Analog will always have a hangup cause code of AST_CAUSE_NORMAL_CLEARING.
- SIP causes of 4xx, 5xx, and 6xx correspond to all 400, 500, and 600 response codes not explicitly listed in the table above.
- AST_CAUSE_UNREGISTERED maps to AST_CAUSE_SUBSCRIBER_ABSENT. This error condition is raised when the endpoint is known but has unregistered itself somehow from Asterisk, e.g., a SIP peer has not registered or sent a REGISTER request with an expiration of 0.

Hangup Handlers

Overview

Hangup handlers are subroutines attached to a channel that will execute when that channel hangs up. Unlike the traditional h extension, hangup handlers follow the channel. Thus hangup handlers are always run when a channel is hung up, regardless of where in the dialplan a channel is executing.

Multiple hangup handlers can be attached to a single channel. If multiple hangup handlers are attached to a channel, the hangup handlers will be executed in the order of most recently added first.

NOTES

- Please note that when the hangup handlers execute in relation to the h extension is not defined. They could execute before or after the h extension.
- Call transfers, call pickup, and call parking can result in channels on both sides of a bridge containing hangup handlers.
- Hangup handlers can be attached to any call leg using pre-dial handlers.

WARNINGS

- As hangup handlers are subroutines, they must be terminated with a call to Return.
- Adding a hangup handler in the h extension or during a hangup handler execution is undefined behaviour.
- As always, hangup handlers, like the h extension, need to execute quickly because they are in the hangup sequence path of the call leg. Specific channel driver protocols like ISDN and SIP may not be able to handle excessive delays completing the hangup sequence.

Dialplan Applications and Functions

All manipulation of a channel's hangup handlers are done using the CHANNEL function. All values manipulated for hangup handlers are write-only.

hangup_handler_push

Used to push a hangup handler onto a channel.

```
same => n,Set(CHANNEL(hangup_handler_push)=[[context,]exten,]priority[(arg1[,...][,argN])]);
```

hangup_handler_pop

Used to pop a hangup handler off a channel. Optionally, a replacement hangup handler can be added to the channel.

```
same => n,Set(CHANNEL(hangup_handler_pop)=[[[context,]exten,]priority[(arg1[,...][,argN])]]);
```

hangup_handler_wipe

Remove all hangup handlers on the channel. Optionally, a new hangup handler can be pushed onto the channel.

```
same => n,Set(CHANNEL(hangup_handler_wipe)=[[[context,]exten,]priority[(arg1[,...][,argN])]]);
```

Examples

Adding hangup handlers to a channel

In this example, three hangup handlers are added to a channel: hdlr3, hdlr2, and hdlr1. When the channel is hung up, they will be executed in the order of most recently added first - so hdlr1 will execute first, followed by hdlr2, then hdlr3.


```
; Some dialplan extension
same => n,Set(CHANNEL(hangup_handler_push)=hdlr3,s,1(args));
same => n,Set(CHANNEL(hangup_handler_push)=hdlr2,s,1(args));
same => n,Set(CHANNEL(hangup_handler_push)=hdlr1,s,1(args));
; Continuing in some dialplan extension

[hdlr1]

exten => s,1,Verbose(0, Executed First)
same => n,Return()

[hdlr2]

exten => s,1,Verbose(0, Executed Second)
same => n,Return()

[hdlr3]

exten => s,1,Verbose(0, Executed Third)
same => n,Return()
```

Removing and replacing hangup handlers

In this example, three hangup handlers are added to a channel: hdlr3, hdlr2, and hdlr1. Using the CHANNEL function's **hangup_handler_pop** value, hdlr1 is removed from the stack of hangup handlers. Then, using the **hangup_handler_pop** value again, hdlr2 is replaced with hdlr4. When the channel is hung up, hdlr4 will be executed, followed by hdlr3.

```
; Some dialplan extension
same => n,Set(CHANNEL(hangup_handler_push)=hdlr3,s,1(args));
same => n,Set(CHANNEL(hangup_handler_push)=hdlr2,s,1(args));
same => n,Set(CHANNEL(hangup_handler_push)=hdlr1,s,1(args));
; Remove hdlr1
same => n,Set(CHANNEL(hangup_handler_pop))
; Replace hdlr2 with hdlr4
same => n,Set(CHANNEL(hangup_handler_pop)=hdlr4,s,1(args));

; Continuing in some dialplan extension

[hdlr1]

exten => s,1,Verbose(0, Not Executed)
same => n,Return()

[hdlr2]

exten => s,1,Verbose(0, Not Executed)
same => n,Return()

[hdlr3]

exten => s,1,Verbose(0, Executed Second)
same => n,Return()

[hdlr4]

exten => s,1,Verbose(0, Executed First)
same => n,Return()
```

CLI Commands

Single channel

```
core show hanguphandlers <chan>
```

Output

```
Channel       Handler
<chan-name>   <first handler to execute>
              <second handler to execute>
              <third handler to execute>
```

All channels

```
core show hanguphandlers all
```

Output

```
Channel       Handler
<chan1-name>  <first handler to execute>
              <second handler to execute>
              <third handler to execute>
<chan2-name>  <first handler to execute>
<chan3-name>  <first handler to execute>
              <second handler to execute>
```

Interactive Connectivity Establishment (ICE) in Asterisk

Overview

If an Asterisk server (or any VoIP server for that matter) is directly accessible on the Internet and and is being "called" by the average SIP softphone or appliance, chances are that turning "on" a check box or maybe some STUN server configuration is all that is needed to make everything "just work". Likewise, configuration is straightforward when servers and phones are on the same local network. If host A and host B are both behind *network address translation* (NAT) firewalls and they need to be able to connect and transmit and receive live data, things can become more difficult. If the networks are very basic, relatively static and the NAT sufficiently configurable, it may be possible to successfully configure a solution (DMZ's, port forwarding, etc). However, add a little additional complexity and things quickly become difficult. Common examples are: layers of NATs between A and B, dynamic address allocation, highly restrictive firewalls, shifting network configurations. Even if configuration is possible, it is burdensome to maintain and can be prohibitively costly in time and resources. The problem is common and severe enough that the VoIP community has been working on solutions for some time. The *Interactive Connectivity Establishment* protocol, or ICE, is a relatively recent and promising approach to resolving these kinds of problems.

Support for ICE was added to Asterisk in version 11. ICE is a standardized mechanism for establishing communication suitable for live media streams between software agents running behind NAT firewalls. Establishing connections through NATs is referred to as *traversing* the NAT. To achieve this, the ICE protocol defines:

- a series of tests for determining internally and externally accessible IP addresses;
- a standard form for specifying a set of prioritized candidate IP addresses in SDP that can be used to reach a software agent in an *offer*;
- a series of operations for validating potential candidates and matching with local candidates to the offered candidates, resulting in candidate pairs;
- a standard form for providing an *answer* specifying validated candidates; and
- a series of rules for picking which candidate pair to ultimately use.

There are mechanisms other than ICE that can be used to communicate through NAT firewalls. They generally require specialized end-to-end configuration or fragile assumptions that may not always be valid. With some basic general configuration (i.e. the hostname of a STUN or TURN server), ICE takes a logical approach to an optimal connection. Configured with available TURN server(s), ICE will even find a successful connection "through" symmetric NATs. In short, if all the software agents are properly configured, ICE will *find a way if there is a way*. It is worthwhile noting that while ICE is intended for RTP, there are other standard mechanisms for SIP messaging through firewalls.

There is a lot to ICE that is beyond the scope of this document. For in-depth detail, see the links to the relevant RFCs below. While the RFCs contain a lot of information, it is mostly oriented at implementation of the ICE protocol and is not necessary for using Asterisk's ICE support. At a user level ICE uses SDP offer/answer, so the general concepts are fairly easy to follow for those familiar with SIP. Also, the details of visually interpreting candidate lists are fairly straightforward and are as easily digestible as media format SDP after a small amount of practice.

Configuring ICE Support in Asterisk

Enabling ICE Support

Asterisk ICE support is disabled by default, and can be enabled globally in `rtp.conf` and both globally or on a SIP peer basis in `sip.conf`. However, as ICE needs a STUN and/or TURN server to gather usable candidates, these do need to be configured to get things working. Since ICE is an RTP level feature, the configuration can be found in the `rtp.conf` file. The configuration applies to all RTP based communications so the options are set in the `general` section. To configure a STUN server add a `stunaddr` option with the hostname of the STUN server. For example,

```
stunaddr=setyourphaserson.stun.org
```

A short list of publicly accessible STUN servers can be found at the VoIP-Info's STUN page.

TURN servers are required for relay candidates and are configured through the `turnaddr` property. TURN servers often require authentication so options are provided for configuring the username and password.

```
turnaddr=4everyseason.turn.org
turnusername=relayme
turnpassword=please
```

The `turnport` option can also be used if the TURN server is running on a non-standard port. If omitted, Asterisk uses the standard port number 3478.

Successful configuration can be visually verified by turning SIP debugging on (`sip set debug on`) in an Asterisk console and looking at INVITE messages as they go past. The body of a typical message would look something like this:

```
0: v=0
1: o=root 1903343929 1903343929 IN IP4 10.0.1.40
2: s=Asterisk PBX SVN-trunk-r372051
3: c=IN IP4 10.0.1.40
4: t=0 0
5: m=audio 17234 RTP/AVP 0 3 8 101
6: a=rtpmap:0 PCMU/8000
7: a=rtpmap:3 GSM/8000
8: a=rtpmap:8 PCMA/8000
9: a=rtpmap:101 telephone-event/8000
10: a=fmtp:101 0-16
11: a=silenceSupp:off - - - -
12: a=ptime:20
13: a=ice-ufrag:0d9cc44338ad8ced48b2d92c34556f4e
14: a=ice-pwd:193c1361446d012a1e298d5278b5c4b6
15: a=candidate:Ha000128qZ 1 UDP 2130706431 10.0.1.40 17234 typ host
16: a=candidate:Ha00030f 1 UDP 2130706431 10.0.3.15 17234 typ host
17: a=candidate:S8e86c939 1 UDP 1694498815 142.134.201.57 17234 typ srflx
18: a=candidate:Ha000128 2 UDP 2130706430 10.0.1.40 17235 typ host
19: a=candidate:Ha00030f 2 UDP 2130706430 10.0.3.15 17235 typ host
20: a=candidate:S8e86c939 2 UDP 1694498814 142.134.201.57 17234 typ srflx
21: a=sendrecv
```

The lines 13 through to 20 are ICE specific. Lines 13 and 14 are automatically generated and are used to identify a peer endpoint in an ICE session. Lines 15 through 20 are examples of candidates. Lines 17 and 20 are examples of *server reflexive* candidates as indicated by the "type srflx" at the end of the candidate strings. Server reflexive address are obtained through STUN and indicate an external binding on the NAT firewall. There are two because there is one for RTP and one for RTCP. RTP and RTCP candidates are distinguishable by their *component id* , 1 for RTP or 2 for RTCP, and is the 2nd "field" of the candidate string. The candidate strings that end in "typ host" are for *host candidates* and indicate actual network interfaces on the host computer. In this case, the host running Asterisk had two network interfaces, one bound to 10.0.1.40 and one bound to 10.0.3.15.

Disabling ICE Support

Generation of SDP for ICE candidate lists can be disabled by adding `icesupport = no` to the general section in `sip.conf` or on a peer-by-peer basis. Since ICE operates on RTP, ICE details are configured in the `rtp.conf` file. To disable ICE support in RTP, add `icesupport = no` to the general section in `rtp.conf`.

References

RFCS:

RFC 5245 Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols
RFC 5389 Session Traversal Utilities for NAT (STUN)
RFC 5766 Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)

Named ACLs

Overview

Named ACLs introduce a new way to define Access Control Lists (ACLs) in Asterisk. Unlike traditional ACLs defined in specific module configuration files, Named ACLs can be shared across multiple modules. Named ACLs can also be accessed via the Asterisk Realtime Architecture (ARA), allowing for run-time updates of ACL information that can be retrieved by multiple consumers of ACL information.

Configuration

Static Configuration

Named ACLs can be defined statically in *acl.conf*. Each context in *acl.conf* defines a specific Named ACL, where the name of the context is the name of the ACL. The syntax for each context follows the permit/deny nomenclature used in traditional ACLs defined in a consumer module's configuration file.

Option	Value	Description
deny	IP address [/Mask]	An IP address to deny, with an optional subnet mask to apply
permit	IP address [/Mask]	An IP address to allow, with an optional subnet mask to apply

Examples

```
; within acl.conf

[name_of_acl1]
deny=0.0.0.0/0.0.0.0
permit=127.0.0.1
```

Multiple rules can be specified in an ACL as well by chaining deny/permit specifiers.

```
[name_of_acl2]
deny=10.24.0.0/255.255.0.0
deny=10.25.0.0/255.255.0.0
permit=10.24.11.0/255.255.255.0
permit=10.24.12.0/255.255.255.0
```

Named ACLs support common modifiers like templates and additions within configuration as well.

```
[template_deny_all](!)
deny=0.0.0.0/0.0.0.0

[deny_all_whitelist_these](template_deny_all)
permit=10.24.20.1
permit=10.24.20.2
permit=10.24.20.3
```

ARA Configuration

The ARA supports Named ACLs using the '**acls**' keyword in *extconfig.conf*.

Example Configuration

```
;in extconfig.conf
acls => odbc,asterisk,acltable
```

Schema

Column Name	Type	Description
name	varchar(80)	Name of the ACL

rule_order	integer	Order to apply the ACL rule. Rules are applied in ascending order. Rule numbers do not have to be sequential
sense	varchar(6)	Either 'permit' or 'deny'
rule	varchar(95)	The IP address/Mask pair to apply

Examples

Table Creation Script (PostgreSQL)

```
CREATE TABLE acltable
(
   "name" character varying(80) NOT NULL,
   rule_order integer NOT NULL,
   sense character varying(6) NOT NULL,
   "rule" character varying(95) NOT NULL,
   CONSTRAINT aclrulekey PRIMARY KEY (name, rule_order, rule, sense)
)
WITH (
   OIDS=FALSE
);
ALTER TABLE acltable OWNER TO asterisk;
GRANT ALL ON TABLE acltable TO asterisk;
)
```

Table Creation Script (SQLite3)

```
BEGIN TRANSACTION;
CREATE TABLE acltable (rule TEXT, sense TEXT, rule_order NUMERIC, name TEXT);
COMMIT;
```

These scripts were generated by pgadmin III and SQLite Database Browser. They might not necessarily apply for your own setup.

Since ACLs are obtained by consumer modules when they are loaded, an ACL updated in an ARA backend will not be propagated automatically to consumers using static configuration. Consumer modules also using ARA for their configuration (such as SIP/IAX2 peers) will similarly be up to date if and only if they have built the peer in question since the changes to the realtime ACL have taken place.

Named ACL Consumers

Named ACLs are supported by the following Asterisk components:

- Manager (IPv4 and IPv6)
- chan_sip (IPv4 and IPv6)
- chan_iax2 (IPv4 only)

Configuration

A consumer of Named ACLs can be configured to use a named ACL using the *acl* option in their ACL access rules. This can be in addition to the ACL rules traditionally defined in those configuration files.

Example 1: referencing a Named ACL

```
; within sip.conf

[peer1]
;stuff
;deny=0.0.0.0/0.0.0.0
;permit=127.0.0.1
acl=name_of_acl_1 ; an ACL included from acl.conf that matches peer1's commented out permits/denies
```

Multiple named ACLs can be referenced as well by specifying a comma delineated list of Named ACLs to apply.

Example 2: multiple Named ACL references

```
; within sip.conf

[peer1]
;stuff
acl=named_acl_1,named_acl_2
```

Similarly, a SIP or IAX2 peer defined in ARA can include an '*acl*' column and list the Named ACLs to apply in that column.

NOTE
Named ACLs can also be defined using multiple instances of the *acl* keyword. This is discouraged, however, as the order in which ACLs are applied can be less obvious then the comma delineated list format.

```
acl=named_acl_1
acl=named_acl_2
```

ACL Rule Application

Each module consumer of ACL information maintains, for each object that uses the information, a list of the defined ACL rule sets that apply to that object. When an address is evaluated for the particular object, the address is evaluated against each rule. For an address to pass the ACL rules, it must pass each ACL rule set that was defined for that object. Failure of any ACL rule set will result in a rejection of the address.

Module Reloads

ACL information is static once a consumer module references that information. Hence, changes in ACL information in an ARA backend will not automatically update consumers of that information. In order for consumers to receive updated ACL information, the Named ACL component must be reloaded.

The Named ACL component supports module reloads, in the same way as other Asterisk components. When the Named ACL component is reloaded, it will issue a request to all consumers of Named ACLs. Those consumer modules will also be automatically reloaded.

WARNING
This implies that reloading the Named ACL component will force a reload of manager, chan_sip, etc. Only reload the Named ACL component if you want all consumers of that information to be reloaded as well.

Pre-Dial Handlers

Overview

Pre-dial handlers allow you to execute a dialplan subroutine on a channel before a call is placed but after the application performing a dial action is invoked. This means that the handlers are executed after the creation of the caller/callee channels, but before any actions have been taken to actually dial the callee channels. You can execute a dialplan subroutine on the caller channel and on each callee channel dialled.

There are two ways in which a pre-dial handler can be invoked:

- The '**B**' option in an application executes a dialplan subroutine on the caller channel before any callee channels are created.
- The '**b**' option in an application executes a dialplan subroutine on each callee channel after it is created but before the call is placed to the end-device.

Pre-dial handlers are supported in the Dial application and the FollowMe application.

WARNINGS

- As pre-dial handlers are implemented using Gosub subroutines, they must be terminated with a call to Return.
- Taking actions in pre-dial handlers that would put the caller/callee channels into other applications will result in undefined behaviour. Pre-dial handlers should be short routines that do not impact the state that the dialling application assumes the channel will be in.

Syntax

Handlers are invoked using similar nomenclature as other options (such as **M** or **U**) in Dial or FollowMe that cause some portion of the dialplan to execute.

```
b([[context^]exten^]priority[(arg1[^...][^argN])])
B([[context^]exten^]priority[(arg1[^...][^argN])])
```

If context or exten are not supplied then the current values from the caller channel are used.

Examples

The examples illustrated below use the following channels:

- *SIP/foo* is calls either *SIP/bar*, *SIP/baz*, or both
- *SIP/foo* is the caller
- *SIP/bar* is a callee
- *SIP/baz* is another callee

Example 1 - Executing a pre-dial handler on the caller channel

```
[default]

exten => s,1,NoOp()
same => n,Dial(SIP/bar,,B(default^caller_handler^1))
same => n,Hangup()

exten => caller_handler,1,NoOp()
same => n,Verbose(0, In caller pre-dial handler!)
same => n,Return()
```


Example 1 CLI Output

```
<SIP/foo-123> Dial(SIP/bar,,B(default^caller_handler^1))
<SIP/foo-123> Executing default,caller_handler,1
<SIP/foo-123> In caller pre-dial handler!
<SIP/foo-123> calling SIP/bar-124
```

Example 2 - Executing a pre-dial handler on a callee channel

```
[default]

exten => s,1,NoOp()
same => n,Dial(SIP/bar,,b(default^callee_handler^1))
same => n,Hangup()

exten => callee_handler,1,NoOp()
same => n,Verbose(0, In callee pre-dial handler!)
same => n,Return()
```

Example 2 CLI Output

```
<SIP/foo-123> Dial(SIP/bar,,b(default^callee_handler^1))
<SIP/bar-124> Executing default,callee_handler,1
<SIP/bar-124> In callee pre-dial handler!
<SIP/foo-123> calling SIP/bar-124
```

Example 3 - Executing a pre-dial handler on multiple callee channels

```
[default]

exten => s,1,NoOp()
same => n,Dial(SIP/bar&SIP/baz,,b(default^callee_handler^1))
same => n,Hangup()

exten => callee_handler,1,NoOp()
same => n,Verbose(0, In callee pre-dial handler!)
same => n,Return()
```

Example 3 CLI Output

```
<SIP/foo-123> Dial(SIP/bar&SIP/baz,,b(default^callee_handler^1))
<SIP/bar-124> Executing default,callee_handler,1
<SIP/bar-124> In callee pre-dial handler!
<SIP/baz-125> Executing default,callee_handler,1
<SIP/baz-125> In callee pre-dial handler!
<SIP/foo-123> calling SIP/bar-124
<SIP/foo-123> calling SIP/baz-125
```

Presence State

Overview

Asterisk 11 has been outfitted with support for presence states. An easy way to understand this is to compare presence state support to the device state support Asterisk has always had. Like with device state support, Asterisk has a core API so that modules can register themselves as presence state providers, alert others to changes in presence state, and query the presence state of others. The biggest difference between the concepts is that device state reflects the current state of a physical device connected to Asterisk, while presence state reflects the current state of the user of the device. For example, a device may currently be `not in use` but the person is `away`. This can be a critical detail when determining the availability of the person.

Asterisk offers the following presence states:

- `not_set`: No presence state has been set for this entity.
- `unavailable`: This entity is present but currently not available for communications.
- `available`: This entity is available for communication.
- `away`: This entity is not present and is unable to communicate.
- `xa`: This entity is not present and is not expected to return for a while.
- `chat`: This entity is available to communicate but would rather use instant messaging than speak.
- `dnd`: This entity does not wish to be disturbed.

In addition to the basic presence states provided, presence also has the concept of a subtype and a message. The subtype is a brief method of describing the nature of the state. For instance, a subtype for the `away` status might be "at home". The message is a longer explanation of the current presence state. Using the same `away` example from before, the message may be "Sick with the flu. Out until the 18th".

Like with device state, presence state can be placed in hints. Presence state hints come after device state hints and are separated by a comma (`,`). As an example:

```
[default]
exten => 2000,hint,SIP/2000,CustomPresence:2000
exten => 2000,1,Dial(SIP/2000)
same => n,Hangup()
```

This would allow for someone subscribing to the extension state of `2000@default` to be notified of device state changes for device `SIP/2000` as well as presence state changes for device `CustomPresence:2000`. The `CustomPresence` presence state provider will be discussed further on this page.

Also like with device state, there is an Asterisk Manager Interface command for querying presence state. Documentation for the AMI `PresenceState` command can be found here.

Differences Between Presence State and Device State Support

While the architectures of presence state and device state support in Asterisk are similar, there are some key differences between the two.

- Asterisk cannot infer presence state changes the same way it can device state changes. For instance, when a SIP endpoint is on a call, Asterisk can infer that the device is being used and report the device state as `in use`. Asterisk cannot infer whether a user of such a device does not wish to be disturbed or would rather chat, though. Thus, all presence state changes have to be manually enacted.
- Asterisk does not take presence into consideration when determining availability of a device. For instance, members of a queue whose device state is `busy` will not be called; however, if that member's device is `not in use` but his presence is `away` then Asterisk will still attempt to call the queue member.
- Asterisk cannot aggregate multiple presence states into a single combined state. Multiple device states can be listed in an extension's hint priority to have a combined state reported. Presence state support in Asterisk lacks this concept.

`func_presencestate` And The `CustomPresence` Provider

The only provider of presence state in Asterisk 11 is the `CustomPresence` provider. This provider is supplied by the `func_presencestate.so` module, which grants access to the `PRESENCE_STATE` dialplan function. The documentation for `PRESENCE_STATE` can be found here.

`CustomPresence` is device-agnostic and can be a handy way to set and query presence. A simple use for `CustomPresence` is demonstrated below.

The following dialplan is meant strictly for demonstration. It is not intended to be used as-is in a production environment.

```
[default]
exten => 2000,1,Answer()
same => n,Set(CURRENT_PRESENCE=${PRESENCE_STATE(CustomPresence:Bob,value)})
same => n,GotoIf($[${CURRENT_PRESENCE}=available]?set_unavailable:set_available)
same => n(set_available),Set(PRESENCE_STATE(CustomPresence:Bob)=available,,)
same => n,Goto(finished)
same => n(set_unavailable),Set(PRESENCE_STATE(CustomPresence:Bob)=unavailable,,)
same => n(finished),Playback(queue-thankyou)
same => n,Hangup

exten => 2001,1,GotoIf($[${PRESENCE_STATE(CustomPresence:Bob,value)}!=available]?voicemail)
same => n,Dial(SIP/Bob)
same => n(voicemail)VoiceMail(Bob@default)
```

With this dialplan, a user can dial `2000@default` to toggle Bob's presence between `available` and `unavailable`. When a user attempts to call Bob using `2001@default`, if Bob's presence is currently not `available` then the call will go directly to voicemail.

One thing to keep in mind with the `PRESENCE_STATE` dialplan function is that, like with `DEVICE_STATE`, state may be queried from any presence provider, but `PRESENCE_STATE` is only capable of setting presence state for the `CustomPresence` presence state provider.

Digium Phone Support

Digium phones have built-in support for Asterisk's presence state. This Video provides more insight on how presence can be set and viewed on Digium phones.

When using Digium phones with the Digium Phone Module for Asterisk, you can set hints in Asterisk so that when one Digium phone's presence is updated, other Digium phones can be notified of the presence change. The DPMA automatically creates provisions such that when a Digium Phone updates its presence, `CustomPresence:<line name>` is updated, where `<line name>` is the value set for the `line=` option in a `type=phone` categor y. Using the example dialplan from the Overview section, Digium phones that are subscribed to `2000@default` will automatically be updated about line 2000's presence whenever line 2000's presence changes.

Digium phones support only the available, away, dnd, xa, and chat states. The unavailable and not_set states are not supported.

Private Representation of Party Information

This page was written by Thomas Arimont of DATUS AG, Germany
Minor editing by Matt Jordan
Private Representation of Caller, Connected and Redirecting Party IDs
Feature implemented in Asterisk 11 by Thomas Arimont, DATUS AG, Germany

Overview

Asterisk already offers a lot of techniques to set and modify party names and numbers of different kinds. There are a number of dialplan functions - CALLERID, CONNECTEDLINE, REDIRECTING - that allow you to read and write a wide range of parameters for Asterisk. However, prior to Asterisk 11 it is quite difficult to modify a party number or name which can only be seen by exactly one particular instantiated channel resp. subscriber.

One example where a modified party number or name on one channel is spread over several channels are supplementary services like call transfer or pickup. To implement these features Asterisk internally copies (has to copy) Caller and Connected IDs from one channel to another.

Another example are extension subscriptions. The monitoring entities (watchers) are notified of state changes and - if desired - of party numbers or names which represent the involved call parties. Also in this case the provided party numbers or names are (have to be) taken from the Caller or Connected ID of the corresponding Asterisk channels.

One major feature where a private representation of party names is essentially needed, i.e., where a party name shall exclusively be signalled to only one particular user, is a private user-specific name resolution for party numbers. A lookup in a private destination-dependent telephone book shall provide party names which cannot be seen by any other user at any time.

Asterisk 11 now provides a mechanism by which private party identification information can be signalled to a particular device.

> For more information on advanced manipulation of Party Identification in Asterisk, see Manipulating Party ID Information. Note that it is **highly** recommended that you are intimately familiar with manipulating party information in Asterisk before reading further.

This feature is supported in the following channel drivers:

- SIP (chan_sip)
- mISDN (chan_misdn)
- PRI (chan_dahdi).

Feature Description

This feature defines additional private number and name elements for Caller ID, Connected ID, and Redirecting IDs inside Asterisk channels. The private number and private name elements can be read or set by the user using the respective Asterisk dialplan functions for those elements.

When a channel initiates a call, if it receives an internal connected line update event or an internal redirecting update event, it first checks if there is a valid Connected ID or Redirecting ID private name or number element present. If this is the case it uses this private representation for protocol signalling. If there is no valid private name or number present, then the valid 'regular' non-private name or number element is used instead.

Automatic Invalidation of Private Information

Once a private name or number on a channel is set and (implicitly) made valid, it is generally used for any further protocol signalling until it is rewritten or invalidated. To simplify the invalidation of private IDs all internally generated connected/redirecting update events and also all connected/redirecting update events which are generated by channels – receiving regarding protocol information - automatically trigger the invalidation of private IDs.

This explicitly takes place when one of the following conditions occurs:

- An internal Asterisk channel masquerading is processed (blind and attended transfers, pickup, parking)
- Extra connected line update events are internally generated during a SIP specific attended transfer
- Extra connected line update events are generated during a call pickup
- Connected line update events are generated in the SIP channel due to receipt of corresponding SIP protocol elements (remote party id header, p-asserted-identity header)
- Redirecting update events are generated in the SIP channel due to receipt of corresponding SIP protocol elements (diversion header)
- Connected line update events are generated in the DAHDI/PRI or mISDN channel due to receipt of corresponding ISDN protocol elements (display IE, connected number IE)

- Redirecting update events are generated in the DAHDI/PRI or mISDN channel due to receipt of corresponding ISDN protocol elements (redirecting/redirection IE, divleginfo facility IE)

Manual Invalidation of Private Information

In some cases the invalidation of private IDs cannot be done automatically and therefore it is a job the user has to do by applying appropriate and explicit dialplan commands. This is done by setting the `priv-nam-valid` item to `0`.

Example

```
same => n,Set(CONNECTEDLINE(priv-name-valid)=0)
```

As an example of where manual invalidation is necessary, consider the case when a private connected name is set towards a particular caller, and after this a blind transfer is executed by the callee to another target where only a 'regular' non-private connected name is set. Because of the priority of the still valid private connected name id on the caller's channel the 'regular' non-private connected name will not become visible to the caller. To solve this issue the user should in general explicitly invalidate the private connected name/number when setting a 'regular' non-private connected name/number.

The user is advised to do the setting of private ids directly on the particular channel which is transmitting the corresponding protocol elements. Using the forward inheritance of the CALLERID, CONNECTLINE and REDIRECTING channel variables can lead to unexpected results.

Recommended Mechanism for Setting Private Identity

The setting of private calling party numbers or names **shall** be done by using the Pre-Dial CALLEE subroutine (Dial application option 'b'). The setting of private REDIRECTING ids towards the redirected-to party **shall** also be done using the Pre-Dial CALLEE subroutine. Since the setting of the private CALLERID in the Pre-Dial CALLEE subroutine has to be made by using the CONNECTEDLINE setter function, the setter function for CALLERID is not used at all in this scenario.

The setting of any kind of private connected numbers or names as well as private `REDIRECTING` IDs towards a caller shall be done by using the system subroutines
CONNECTED_LINE_SEND_SUB resp. REDIRECTING_SEND_SUB.

If not using the private number and name representation feature at all, i.e., if using only the 'regular' CALLERID, CONNECTEDLINE and REDIRECTING related function datatypes, the current characteristics of Asterisk's manipulation of party identification is not affected by the new extended functionality.

Dialplan Manipulation

To grant access to the private name and number representation out of the asterisk dialplan, the read and write functions of the three Asterisk defined functions CALLERID, CONNECTEDLINE and REDIRECTING are extended by the following datatypes. The formats of these additional datatypes are equivalent to the corresponding regular 'non-private' already existing datatypes:

CALLERID:

- `priv-all`
- `priv-name`
- `priv-name-valid`
- `priv-name-charset`
- `priv-name-pres`
- `priv-num`
- `priv-num-valid`
- `priv-num-plan`
- `priv-num-pres`
- `priv-subaddr`
- `priv-subaddr-valid`
- `priv-subaddr-type`

- `priv-subaddr-odd`
- `priv-tag`

CONNECTEDLINE:

- `priv-name`
- `priv-name-valid`
- `priv-name-pres`
- `priv-name-charset`
- `priv-num`
- `priv-num-valid`
- `priv-num-pres`
- `priv-num-plan`
- `priv-subaddr`
- `priv-subaddr-valid`
- `priv-subaddr-type`
- `priv-subaddr-odd`
- `priv-tag`

REDIRECTING:

- `priv-orig-name`
- `priv-orig-name-valid`
- `priv-orig-name-pres`
- `priv-orig-name-charset`
- `priv-orig-num`
- `priv-orig-num-valid`
- `priv-orig-num-pres`
- `priv-orig-num-plan`
- `priv-orig-subaddr`
- `priv-orig-subaddr-valid`
- `priv-orig-subaddr-type`
- `priv-orig-subaddr-odd`
- `priv-orig-tag`
- `priv-from-name`
- `priv-from-name-valid`
- `priv-from-name-pres`
- `priv-from-name-charset`
- `priv-from-num`
- `priv-from-num-valid`
- `priv-from-num-pres`
- `priv-from-num-plan`
- `priv-from-subaddr`
- `priv-from-subaddr-valid`
- `priv-from-subaddr-type`
- `priv-from-subaddr-odd`
- `priv-from-tag`
- `priv-to-name`
- `priv-to-name-valid`
- `priv-to-name-pres`
- `priv-to-name-charset`
- `priv-to-num`
- `priv-to-num-valid`
- `priv-to-num-pres`
- `priv-to-num-plan`
- `priv-to-subaddr`
- `priv-to-subaddr-valid`
- `priv-to-subaddr-type`
- `priv-to-subaddr-odd`
- `priv-to-tag`

Mixing of private and public id elements is valid.

The presentation and the numbering plan datatypes of private ids become visible when a private id number or name is valid.

Example Dialplans

Setting the private calling name

```
[incoming_context]
; (optionally) Setting the public calling name as usual
exten => 10,1,ExecIf($[${CALLERID(num)}=20]?
    Set(CALLERID(name)=Peter Public):)
exten => 10,n,ExecIf($[${CALLERID(num)}=30]?
    Set(CALLERID(name)=Mark Public):)

;using the Dial() b-option to execute a pre-dial subroutine on the target channel
exten => 10,n,Dial(SIP/10, ,b(privatecallingname))
exten => 10,n,Hangup()

; setting  the private calling name (on target/dialed-to channel only)
; Since the Pre-dial CALLEE subroutine is acting on the target channel the
; CONNECTEDLINE parameter has to be set instead of the CALLERID parameter
; This might look strange at first sight but the CALLERID of the dialing channel
; becomes the CONNECTEDLINE of the dialed-to channel since the the Dial()-
; application copies the caller id of the dialing channel to the connected id of the
; target channel
exten => 10,n(privatecallingname), NoOp()
exten => 10,n,ExecIf($[${CONNECTEDLINE(num}=20]?
    Set(CONNECTEDLINE(priv-name,i)=Peter Private):)
exten => 10,n,ExecIf($[${CONNECTEDLINE(num}=30]?
    Set(CONNECTEDLINE(priv-name,i)=Mark Private):)

; Of course in a more general approach the private calling name should be determined
; by a database lookup instead. For this purpose the database would have to provide
; user-specific (identified by the called party number, i.e. ${EXTEN}) private party
; names of the calling party (identified by the calling party number,
; i.e. ${CONNECTEDLINE(num)})

exten => 10,n,Return()
```

The setting of a private redirecting-from name (if an internal diversion shall be indicated to the forwarded-to party) can be done similar in the same context.

Setting the private connected name

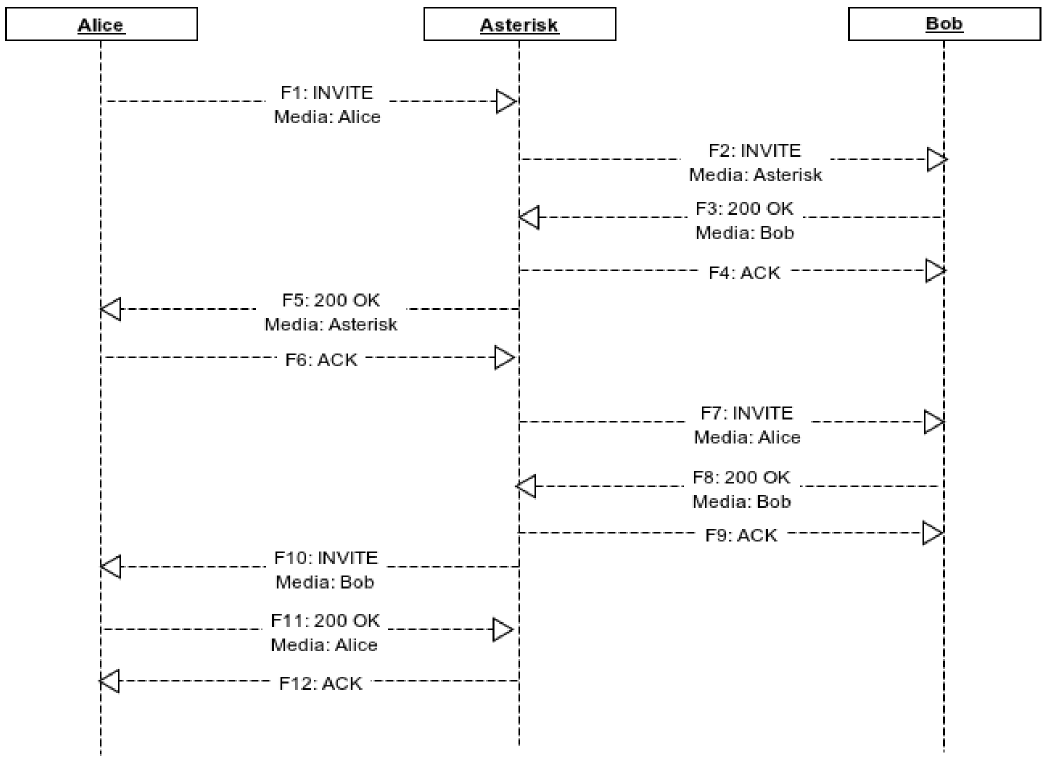

```
[globals]
 CONNECTED_LINE_SEND_SUB = connectedline,s,1
CONNECTED_LINE_SEND_SUB_ARGS =,

[incoming context]
; Setting the public connected name or/and number as usual
exten => 10,1,Set(CONNECTEDLINE(name,i)=Donald Public)
exten => 10,1,Set(CONNECTEDLINE(num,i)=10)

; Generally invalidate the private connected name and number in case it is not
; explicitly set somewhere later. This prevents the technology channel which is
; receiving the internal connectedline update event from using a former set private
; connected id representation

exten => 10,n,Set(CONNECTEDLINE(priv-name-valid,i)=0)
exten => 10,n,Set(CONNECTEDLINE(priv-num-valid,i)=0)
exten => 10,n,Set(CONNECTEDLINE(source)=answer)

exten => 10,n,Dial(SIP/10, ,)
exten => 10,n,Hangup()

[connectedline]
exten => s,1, NoOp()
exten => s,n,GotoIf($[${CHANNEL(channeltype)}=Local]?out:)
; The following setting is done on the technology channel which signals
; the connected name by its corresponding protocol elements

; e.g., only signal the private connected name of callee '10' to caller '20'
exten => s,n,GotoIf($[${CALLERID(num)}!=20]?out:)
exten => s,n,GotoIf($[${CONNECTEDLINE(num)}!=10]?out:)
exten => s,n,Set(CONNECTEDLINE(priv-name)=Donald Private)

; Of course in a more general approach the private connected name should be
; determined by a database lookup instead. For this purpose the database would have
; to provide user-specific (identified by the destination of the connectedline update
; event, i.e. ${CALLERID(num)}) private party names which represents the
; originator of the connectedline update event (i.e. identified by
; ${CONNECTEDLINE(num)})

exten => s,n(out),Return()
```

Setting the private `redirecting-from` or `redirecting-to` names (if an internal diversion shall be indicated to the `forwarded-to` party or a diversion signalling shall be manipulated towards an involved subscriber) can be done similar using the `REDIRECTING_SEND_SUB` subroutine.

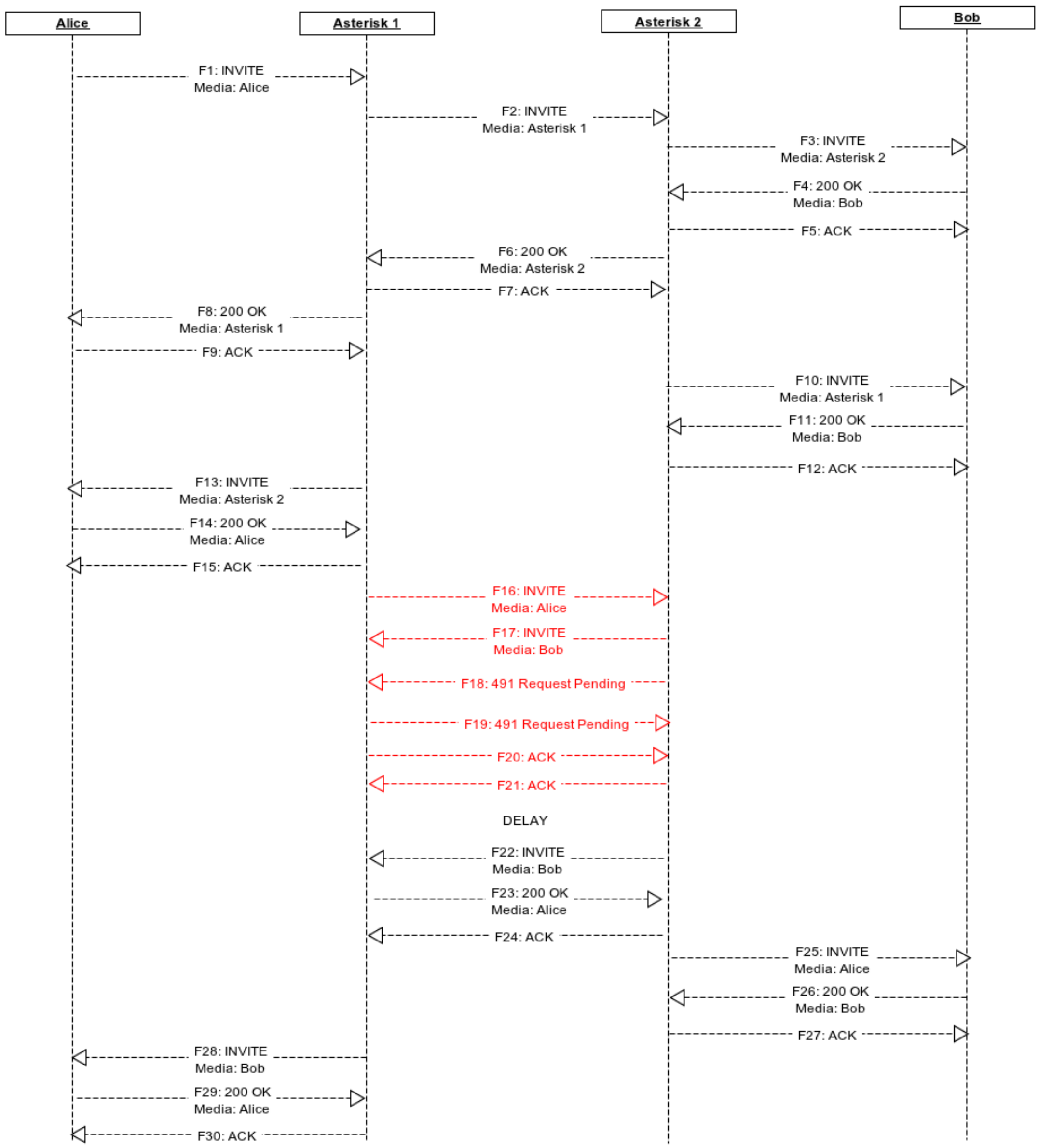
SIP Direct Media Reinvite Glare Avoidance

Overview

When SIP endpoints communicate by way of Asterisk, Asterisk will attempt to send SIP reinvites in order to allow the endpoints to communicate directly. This allows for the computational load on the Asterisk server to be decreased while also lessening the latency of the media streams between the endpoints. Typical SIP traffic for a call might look like this:



When multiple Asterisk servers are in the path between the endpoints, then both Asterisk servers will attempt to send direct media reinvites. If it happens to be that the two Asterisk servers direct their reinvites to each other at the same time, then each of the Asterisk servers will respond to the reinvites with 491 responses. After a delay, the downstream Asterisk server will attempt its reinvite again and succeed. A diagram of this situation looks like this:

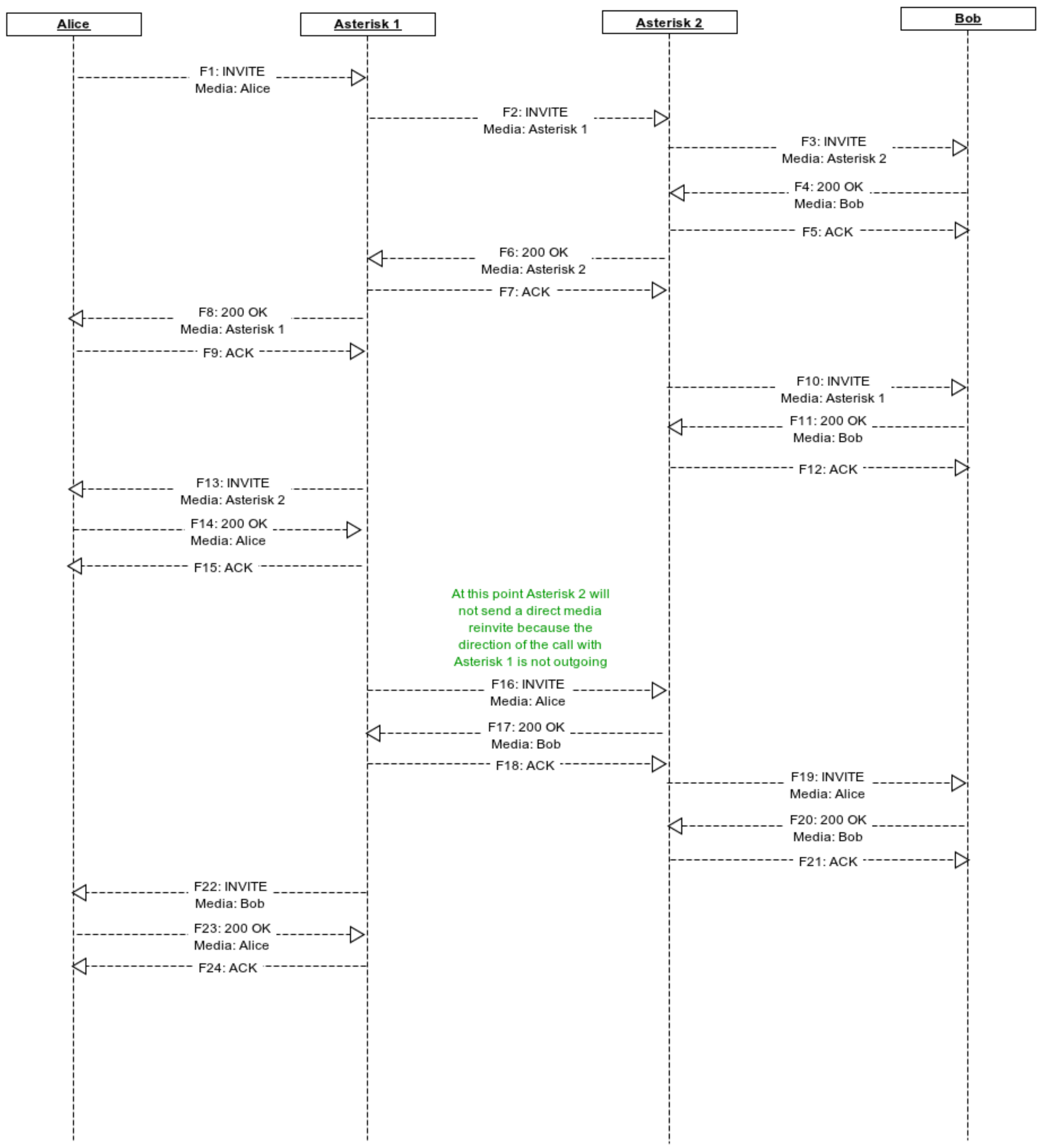


The problematic area is higlighted in red. While this eventually results in direct media flowing between the endpoints, the delay between the 491 responses and the re-attempt at reinviting the media may be noticeable to the end users. If more than two Asterisk servers are in the path between callers, this delay can be longer. In Asterisk 11, a new option has been added to chan_sip in an attempt to address this.

`directmedia = outgoing`

The problem in the second diagram was that both Asterisk servers assumed control of the path between them. In reality, it is only required that one of the Asterisk servers does this. The `outgoing` setting for the `directmedia` option addresses this problem.

The way this option works is when the SIP channel driver is told by the RTP layer to send a direct media reinvite out, we check to see if the `directmedia` setting is set to `outgoing` for the dialog. If it is, and the call direction is not outgoing, then the SIP channel driver will refrain from sending a reinvite. After this first denial to send the direct media reinvite, the SIP channel driver will no longer refuse to send if the RTP layer requests it again. Here is a diagram showing how this works if Asterisk 2 has `directmedia = outgoing` set:



If Asterisk 1 also has `directmedia` set to `outgoing` then calls from Asterisk 2 to Asterisk 1 will also avoid reinvite glares.

Caveats

Since this option is a new value accepted for the `directmedia` setting in `sip.conf`, this setting can be applied globally. This is almost assuredly not what you want to do. You should only ever set `directmedia` to `outgoing` on individual peers.

When choosing which peers to set this option on, you should be careful. It is best to only set this option on peers that are also under your control and that will also have this option set. For instance, if your setup has multiple peered Asterisk servers, then it is a great idea to use this option for those peers. If, on the other hand, you have had SIP reinvite glare issues with a SIP provider, then you should be hesitant to set this option without thoroughly testing with your provider first.

When setting `directmedia = outgoing` on your peered Asterisk servers, it is a good idea to set the option in the sip.conf file (or realtime storage) of all the Asterisk servers in question. This way calls can go from any Asterisk server to any other Asterisk server and glares will be prevented.

Asterisk 11 Command Reference

This page is the top level page for all of the Asterisk 11 applications, functions, manager actions, manager events, and AGI commands that are kept in the XML based documentation that is included with Asterisk 11.

Asterisk 11 AGI Commands

Asterisk 11 AGICommand_answer

ANSWER

Synopsis

Answer channel

Description

Answers channel if not already in answer state. Returns `-1` on channel failure, or `0` if successful.

Syntax

```
ANSWER
```

Arguments

See Also

- Asterisk 11 AGICommand_hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_asyncagi break

ASYNCAGI BREAK

Synopsis

Interrupts Async AGI

Description

Interrupts expected flow of Async AGI commands and returns control to previous source (typically, the PBX dialplan).

Syntax

```
ASYNCAGI BREAK
```

Arguments

See Also

- Asterisk 11 AGICommand_hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_channel status

CHANNEL STATUS

Synopsis

Returns status of the connected channel.

Description

Returns the status of the specified *channelname*. If no channel name is given then returns the status of the current channel.

Return values:

- `0` - Channel is down and available.
- `1` - Channel is down, but reserved.
- `2` - Channel is off hook.
- `3` - Digits (or equivalent) have been dialed.
- `4` - Line is ringing.
- `5` - Remote end is ringing.
- `6` - Line is up.
- `7` - Line is busy.

Syntax

```
CHANNEL STATUS CHANNELNAME
```

Arguments

- `channelname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_control stream file

CONTROL STREAM FILE

Synopsis

Sends audio file on channel and allows the listener to control the stream.

Description

Send the given file, allowing playback to be controlled by the given digits, if any. Use double quotes for the digits if you wish none to be permitted. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed, or `-1` on error or if the channel was disconnected.

Syntax

```
CONTROL STREAM FILE FILENAME ESCAPE_DIGITS SKIPMS FFCHAR REWCHR PAUSECHR
```

Arguments

- `filename` - The file extension must not be included in the filename.
- `escape_digits`
- `skipms`
- `ffchar` - Defaults to `*`
- `rewchr` - Defaults to `#`
- `pausechr`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_database del

DATABASE DEL

Synopsis

Removes database key/value

Description

Deletes an entry in the Asterisk database for a given *family* and *key*.

Returns `1` if successful, `0` otherwise.

Syntax

```
DATABASE DEL FAMILY KEY
```

Arguments

- `family`
- `key`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_database deltree

DATABASE DELTREE

Synopsis

Removes database keytree/value

Description

Deletes a *family* or specific *keytree* within a *family* in the Asterisk database.

Returns `1` if successful, `0` otherwise.

Syntax

```
DATABASE DELTREE FAMILY KEYTREE
```

Arguments

- `family`
- `keytree`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_database get

DATABASE GET

Synopsis

Gets database value

Description

Retrieves an entry in the Asterisk database for a given *family* and *key*.

Returns `0` if *key* is not set. Returns `1` if *key* is set and returns the variable in parenthesis.

Example return code: 200 result=1 (testvariable)

Syntax

```
DATABASE GET FAMILY KEY
```

Arguments

- `family`
- `key`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_database put

DATABASE PUT

Synopsis

Adds/updates database value

Description

Adds or updates an entry in the Asterisk database for a given *family*, *key*, and *value*.

Returns `1` if successful, `0` otherwise.

Syntax

```
DATABASE PUT FAMILY KEY VALUE
```

Arguments

- `family`
- `key`
- `value`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_exec

EXEC

Synopsis

Executes a given Application

Description

Executes *application* with given *options*.

Returns whatever the *application* returns, or `-2` on failure to find *application*.

Syntax

```
EXEC APPLICATION OPTIONS
```

Arguments

- `application`
- `options`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_get data

GET DATA

Synopsis

Prompts for DTMF on a channel

Description

Stream the given *file*, and receive DTMF data.

Returns the digits received from the channel at the other end.

Syntax

```
GET DATA FILE TIMEOUT MAXDIGITS
```

Arguments

- `file`
- `timeout`
- `maxdigits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_get full variable

GET FULL VARIABLE

Synopsis

Evaluates a channel expression

Description

Returns `0` if *variablename* is not set or channel does not exist. Returns `1` if *variablename* is set and returns the variable in parenthesis. Understands complex variable names and builtin variables, unlike GET VARIABLE.

Example return code: 200 result=1 (testvariable)

Syntax

```
GET FULL VARIABLE VARIABLENAME CHANNEL NAME
```

Arguments

- `variablename`
- `channel name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_get option

GET OPTION

Synopsis

Stream file, prompt for DTMF, with timeout.

Description

Behaves similar to STREAM FILE but used with a timeout option.

Syntax

```
GET OPTION FILENAME ESCAPE_DIGITS TIMEOUT
```

Arguments

- `filename`
- `escape_digits`
- `timeout`

See Also

- Asterisk 11 AGICommand_stream file

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_get variable

GET VARIABLE

Synopsis

Gets a channel variable.

Description

Returns `0` if *variablename* is not set. Returns `1` if *variablename* is set and returns the variable in parentheses.

Example return code: 200 result=1 (testvariable)

Syntax

```
GET VARIABLE VARIABLENAME
```

Arguments

- `variablename`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_gosub

GOSUB

Synopsis

Cause the channel to execute the specified dialplan subroutine.

Description

Cause the channel to execute the specified dialplan subroutine, returning to the dialplan with execution of a Return().

Syntax

```
GOSUB CONTEXT EXTENSION PRIORITY OPTIONAL-ARGUMENT
```

Arguments

- `context`
- `extension`
- `priority`
- `optional-argument`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_hangup

HANGUP

Synopsis

Hangup a channel.

Description

Hangs up the specified channel. If no channel name is given, hangs up the current channel

Syntax

```
HANGUP CHANNELNAME
```

Arguments

- `channelname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_noop

NOOP

Synopsis

Does nothing.

Description

Does nothing.

Syntax

```
NOOP
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_receive char

RECEIVE CHAR

Synopsis

Receives one character from channels supporting it.

Description

Receives a character of text on a channel. Most channels do not support the reception of text. Returns the decimal value of the character if one is received, or `0` if the channel does not support text reception. Returns `-1` only on error/hangup.

Syntax

```
RECEIVE CHAR TIMEOUT
```

Arguments

- `timeout` - The maximum time to wait for input in milliseconds, or `0` for infinite. Most channels

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_receive text

RECEIVE TEXT

Synopsis

Receives text from channels supporting it.

Description

Receives a string of text on a channel. Most channels do not support the reception of text. Returns `-1` for failure or `1` for success, and the string in parenthesis.

Syntax

```
RECEIVE TEXT TIMEOUT
```

Arguments

- `timeout` - The timeout to be the maximum time to wait for input in milliseconds, or `0` for infinite.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_record file

RECORD FILE

Synopsis

Records to a given file.

Description

Record to a file until a given dtmf digit in the sequence is received. Returns `-1` on hangup or error. The format will specify what kind of file will be recorded. The *timeout* is the maximum record time in milliseconds, or `-1` for no *timeout*. *offset samples* is optional, and, if provided, will seek to the offset without exceeding the end of the file. *silence* is the number of seconds of silence allowed before the function returns despite the lack of dtmf digits or reaching *time out*. *silence* value must be preceded by `s=` and is also optional.

Syntax

```
RECORD FILE FILENAME FORMAT ESCAPE_DIGITS TIMEOUT OFFSET SAMPLES BEEP S=SILENCE
```

Arguments

- `filename`
- `format`
- `escape_digits`
- `timeout`
- `offset samples`
- `BEEP`
- `s=silence`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say alpha

SAY ALPHA

Synopsis

Says a given character string.

Description

Say a given character string, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed or `-1` on error/hangup.

Syntax

```
SAY ALPHA NUMBER ESCAPE_DIGITS
```

Arguments

- `number`
- `escape_digits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say date

SAY DATE

Synopsis

Says a given date.

Description

Say a given date, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed or `-1` on error/hangup.

Syntax

```
SAY DATE DATE ESCAPE_DIGITS
```

Arguments

- `date` - Is number of seconds elapsed since 00:00:00 on January 1, 1970. Coordinated Universal Time (UTC).
- `escape_digits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say datetime

SAY DATETIME

Synopsis

Says a given time as specified by the format given.

Description

Say a given time, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed or `-1` on error/hangup.

Syntax

```
SAY DATETIME TIME ESCAPE_DIGITS FORMAT TIMEZONE
```

Arguments

- `time` - Is number of seconds elapsed since 00:00:00 on January 1, 1970, Coordinated Universal Time (UTC)
- `escape_digits`
- `format` - Is the format the time should be said in. See `voicemail.conf` (defaults to `ABdY 'digits/at' IMp`).
- `timezone` - Acceptable values can be found in `/usr/share/zoneinfo` Defaults to machine default.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say digits

SAY DIGITS

Synopsis

Says a given digit string.

Description

Say a given digit string, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed or `-1` on error/hangup.

Syntax

```
SAY DIGITS NUMBER ESCAPE_DIGITS
```

Arguments

- `number`
- `escape_digits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say number

SAY NUMBER

Synopsis

Says a given number.

Description

Say a given number, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed or `-1` on error/hangup.

Syntax

```
SAY NUMBER NUMBER ESCAPE_DIGITS GENDER
```

Arguments

- `number`
- `escape_digits`
- `gender`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say phonetic

SAY PHONETIC

Synopsis

Says a given character string with phonetics.

Description

Say a given character string with phonetics, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit pressed, the ASCII numerical value of the digit if one was pressed, or `-1` on error/hangup.

Syntax

```
SAY PHONETIC STRING ESCAPE_DIGITS
```

Arguments

- `string`
- `escape_digits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_say time

SAY TIME

Synopsis

Says a given time.

Description

Say a given time, returning early if any of the given DTMF digits are received on the channel. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed or `-1` on error/hangup.

Syntax

```
SAY TIME TIME ESCAPE_DIGITS
```

Arguments

- `time` - Is number of seconds elapsed since 00:00:00 on January 1, 1970. Coordinated Universal Time (UTC).
- `escape_digits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_send image

SEND IMAGE

Synopsis

Sends images to channels supporting it.

Description

Sends the given image on a channel. Most channels do not support the transmission of images. Returns `0` if image is sent, or if the channel does not support image transmission. Returns `-1` only on error/hangup. Image names should not include extensions.

Syntax

```
SEND IMAGE IMAGE
```

Arguments

- `image`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_send text

SEND TEXT

Synopsis

Sends text to channels supporting it.

Description

Sends the given text on a channel. Most channels do not support the transmission of text. Returns `0` if text is sent, or if the channel does not support text transmission. Returns `-1` only on error/hangup.

Syntax

```
SEND TEXT TEXT TO SEND
```

Arguments

- `text to send` - Text consisting of greater than one word should be placed in quotes since the command only accepts a single argument.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set autohangup

SET AUTOHANGUP

Synopsis

Autohangup channel in some time.

Description

Cause the channel to automatically hangup at *time* seconds in the future. Of course it can be hungup before then as well. Setting to `0` will cause the autohangup feature to be disabled on this channel.

Syntax

```
SET AUTOHANGUP TIME
```

Arguments

- `time`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set callerid

SET CALLERID

Synopsis

Sets callerid for the current channel.

Description

Changes the callerid of the current channel.

Syntax

```
SET CALLERID NUMBER
```

Arguments

- `number`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set context

SET CONTEXT

Synopsis

Sets channel context.

Description

Sets the context for continuation upon exiting the application.

Syntax

```
SET CONTEXT DESIRED CONTEXT
```

Arguments

- `desired context`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set extension

SET EXTENSION

Synopsis

Changes channel extension.

Description

Changes the extension for continuation upon exiting the application.

Syntax

```
SET EXTENSION NEW EXTENSION
```

Arguments

- `new extension`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set music

SET MUSIC

Synopsis

Enable/Disable Music on hold generator

Description

Enables/Disables the music on hold generator. If *class* is not specified, then the `default` music on hold class will be used. This generator will be stopped automatically when playing a file.

Always returns `0`.

Syntax

```
SET MUSIC  CLASS
```

Arguments

- {{}}
    - {{}}
        -
            - `on`
    - {{}}
        -
            - `off`
- `class`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set priority

SET PRIORITY

Synopsis

Set channel dialplan priority.

Description

Changes the priority for continuation upon exiting the application. The priority must be a valid priority or label.

Syntax

```
SET PRIORITY PRIORITY
```

Arguments

- `priority`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_set variable

SET VARIABLE

Synopsis

Sets a channel variable.

Description

Sets a variable to the current channel.

Syntax

```
SET VARIABLE VARIABLENAME VALUE
```

Arguments

- `variablename`
- `value`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech activate grammar

SPEECH ACTIVATE GRAMMAR

Synopsis

Activates a grammar.

Description

Activates the specified grammar on the speech object.

Syntax

```
SPEECH ACTIVATE GRAMMAR GRAMMAR NAME
```

Arguments

- `grammar name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech create

SPEECH CREATE

Synopsis

Creates a speech object.

Description

Create a speech object to be used by the other Speech AGI commands.

Syntax

```
SPEECH CREATE ENGINE
```

Arguments

- `engine`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech deactivate grammar

SPEECH DEACTIVATE GRAMMAR

Synopsis

Deactivates a grammar.

Description

Deactivates the specified grammar on the speech object.

Syntax

```
SPEECH DEACTIVATE GRAMMAR GRAMMAR NAME
```

Arguments

- `grammar name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech destroy

SPEECH DESTROY

Synopsis

Destroys a speech object.

Description

Destroy the speech object created by `SPEECH CREATE`.

Syntax

```
SPEECH DESTROY
```

Arguments

See Also

- Asterisk 11 AGICommand_speech create

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech load grammar

SPEECH LOAD GRAMMAR

Synopsis

Loads a grammar.

Description

Loads the specified grammar as the specified name.

Syntax

```
SPEECH LOAD GRAMMAR GRAMMAR NAME PATH TO GRAMMAR
```

Arguments

- `grammar name`
- `path to grammar`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech recognize

SPEECH RECOGNIZE

Synopsis

Recognizes speech.

Description

Plays back given *prompt* while listening for speech and dtmf.

Syntax

```
SPEECH RECOGNIZE PROMPT TIMEOUT OFFSET
```

Arguments

- `prompt`
- `timeout`
- `offset`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech set

SPEECH SET

Synopsis

Sets a speech engine setting.

Description

Set an engine-specific setting.

Syntax

```
SPEECH SET NAME VALUE
```

Arguments

- `name`
- `value`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_speech unload grammar

SPEECH UNLOAD GRAMMAR

Synopsis

Unloads a grammar.

Description

Unloads the specified grammar.

Syntax

```
SPEECH UNLOAD GRAMMAR GRAMMAR NAME
```

Arguments

- `grammar name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_stream file

STREAM FILE

Synopsis

Sends audio file on channel.

Description

Send the given file, allowing playback to be interrupted by the given digits, if any. Returns `0` if playback completes without a digit being pressed, or the ASCII numerical value of the digit if one was pressed, or `-1` on error or if the channel was disconnected. If musiconhold is playing before calling stream file it will be automatically stopped and will not be restarted after completion.

Syntax

```
STREAM FILE FILENAME ESCAPE_DIGITS SAMPLE OFFSET
```

Arguments

- `filename` - File name to play. The file extension must not be included in the *filename*.
- `escape_digits` - Use double quotes for the digits if you wish none to be permitted.
- `sample offset` - If sample offset is provided then the audio will seek to sample offset before play starts.

See Also

- Asterisk 11 AGICommand_control stream file

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_tdd mode

TDD MODE

Synopsis

Toggles TDD mode (for the deaf).

Description

Enable/Disable TDD transmission/reception on a channel. Returns `1` if successful, or `0` if channel is not TDD-capable.

Syntax

```
TDD MODE BOOLEAN
```

Arguments

- `boolean`
  - `on`
  - `off`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_verbose

VERBOSE

Synopsis

Logs a message to the asterisk verbose log.

Description

Sends *message* to the console via verbose message system. *level* is the verbose level (1-4). Always returns `1`

Syntax

```
VERBOSE MESSAGE LEVEL
```

Arguments

- `message`
- `level`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AGICommand_wait for digit

WAIT FOR DIGIT

Synopsis

Waits for a digit to be pressed.

Description

Waits up to *timeout* milliseconds for channel to receive a DTMF digit. Returns `-1` on channel failure, `0` if no digit is received in the timeout, or the numerical value of the ascii of the digit if one is received. Use `-1` for the *timeout* value if you desire the call to block indefinitely.

Syntax

```
WAIT FOR DIGIT TIMEOUT
```

Arguments

- `timeout`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AMI Actions

Asterisk 11 ManagerAction_AbsoluteTimeout

AbsoluteTimeout

Synopsis

Set absolute timeout.

Description

Hangup a channel after a certain time. Acknowledges set time with `Timeout Set` message.

Syntax

```
Action: AbsoluteTimeout
ActionID: <value>
Channel: <value>
Timeout: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to hangup.
- `Timeout` - Maximum duration of the call (sec).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_AgentLogoff

AgentLogoff

Synopsis

Sets an agent as no longer logged in.

Description

Sets an agent as no longer logged in.

Syntax

```
Action: AgentLogoff
ActionID: <value>
Agent: <value>
Soft: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Agent` - Agent ID of the agent to log off.
- `Soft` - Set to `true` to not hangup existing calls.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Agents

Agents

Synopsis

Lists agents and their status.

Description

Will list info about all possible agents.

Syntax

```
Action: Agents
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_AGI

AGI

Synopsis

Add an AGI command to execute by Async AGI.

Description

Add an AGI command to the execute queue of the channel in Async AGI.

Syntax

```
Action: AGI
ActionID: <value>
Channel: <value>
Command: <value>
CommandID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel that is currently in Async AGI.
- `Command` - Application to execute.
- `CommandID` - This will be sent back in CommandID header of AsyncAGI exec event notification.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_AOCMessage

AOCMessage

Synopsis

Generate an Advice of Charge message on a channel.

Description

Generates an AOC-D or AOC-E message on a channel.

Syntax

```
Action: AOCMessage
ActionID: <value>
Channel: <value>
ChannelPrefix: <value>
MsgType: <value>
ChargeType: <value>
UnitAmount(0): <value>
UnitType(0): <value>
CurrencyName: <value>
CurrencyAmount: <value>
CurrencyMultiplier: <value>
TotalType: <value>
AOCBillingId: <value>
ChargingAssociationId: <value>
ChargingAssociationNumber: <value>
ChargingAssociationPlan: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to generate the AOC message on.
- `ChannelPrefix` - Partial channel prefix. By using this option one can match the beginning part of a channel name without having to put the entire name in. For example if a channel name is SIP/snom-00000001 and this value is set to SIP/snom, then that channel matches and the message will be sent. Note however that only the first matched channel has the message sent on it.
- `MsgType` - Defines what type of AOC message to create, AOC-D or AOC-E
    - `D`
    - `E`
- `ChargeType` - Defines what kind of charge this message represents.
    - `NA`
    - `FREE`
    - `Currency`
    - `Unit`
- `UnitAmount(0)` - This represents the amount of units charged. The ETSI AOC standard specifies that this value along with the optional UnitType value are entries in a list. To accommodate this these values take an index value starting at 0 which can be used to generate this list of unit entries. For Example, If two unit entires were required this could be achieved by setting the paramter UnitAmount(0)=1234 and UnitAmount(1)=5678. Note that UnitAmount at index 0 is required when ChargeType=Unit, all other entries in the list are optional.
- `UnitType(0)` - Defines the type of unit. ETSI AOC standard specifies this as an integer value between 1 and 16, but this value is left open to accept any positive integer. Like the UnitAmount parameter, this value represents a list entry and has an index parameter that starts at 0.
- `CurrencyName` - Specifies the currency's name. Note that this value is truncated after 10 characters.
- `CurrencyAmount` - Specifies the charge unit amount as a positive integer. This value is required when ChargeType==Currency.
- `CurrencyMultiplier` - Specifies the currency multiplier. This value is required when ChargeType==Currency.
    - `OneThousandth`
    - `OneHundredth`
    - `OneTenth`
    - `One`
    - `Ten`
    - `Hundred`
    - `Thousand`
- `TotalType` - Defines what kind of AOC-D total is represented.
    - `Total`

        - `SubTotal`
- `AOCBillingId` - Represents a billing ID associated with an AOC-D or AOC-E message. Note that only the first 3 items of the enum are valid AOC-D billing IDs
    - `Normal`
    - `ReverseCharge`
    - `CreditCard`
    - `CallFwdUnconditional`
    - `CallFwdBusy`
    - `CallFwdNoReply`
    - `CallDeflection`
    - `CallTransfer`
- `ChargingAssociationId` - Charging association identifier. This is optional for AOC-E and can be set to any value between -32768 and 32767
- `ChargingAssociationNumber` - Represents the charging association party number. This value is optional for AOC-E.
- `ChargingAssociationPlan` - Integer representing the charging plan associated with the ChargingAssociationNumber. The value is bits 7 through 1 of the Q.931 octet containing the type-of-number and numbering-plan-identification fields.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Atxfer

Atxfer

Synopsis

Attended transfer.

Description

Attended transfer.

Syntax

```
Action: Atxfer
ActionID: <value>
Channel: <value>
Exten: <value>
Context: <value>
Priority: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Transferer's channel.
- `Exten` - Extension to transfer to.
- `Context` - Context to transfer to.
- `Priority` - Priority to transfer to.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Bridge

Bridge

Synopsis

Bridge two channels already in the PBX.

Description

Bridge together two channels already in the PBX.

Syntax

```
Action: Bridge
ActionID: <value>
Channel1: <value>
Channel2: <value>
Tone: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel1` - Channel to Bridge to Channel2.
- `Channel2` - Channel to Bridge to Channel1.
- `Tone` - Play courtesy tone to Channel 2.
    - `yes`
    - `no`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Challenge

Challenge

Synopsis

Generate Challenge for MD5 Auth.

Description

Generate a challenge for MD5 authentication.

Syntax

```
Action: Challenge
ActionID: <value>
AuthType: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `AuthType` - Digest algorithm to use in the challenge. Valid values are:
    - `MD5`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ChangeMonitor

ChangeMonitor

Synopsis

Change monitoring filename of a channel.

Description

This action may be used to change the file started by a previous 'Monitor' action.

Syntax

```
Action: ChangeMonitor
ActionID: <value>
Channel: <value>
File: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Used to specify the channel to record.
- `File` - Is the new name of the file created in the monitor spool directory.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Command

Command

Synopsis

Execute Asterisk CLI Command.

Description

Run a CLI command.

Syntax

```
Action: Command
ActionID: <value>
Command: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Command` - Asterisk CLI command to run.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeKick

ConfbridgeKick

Synopsis

Kick a Confbridge user.

Description

Syntax

```
Action: ConfbridgeKick
ActionID: <value>
Conference: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`
- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeList

ConfbridgeList

Synopsis

List participants in a conference.

Description

Lists all users in a particular ConfBridge conference. ConfbridgeList will follow as separate events, followed by a final event called ConfbridgeListComplete.

Syntax

```
Action: ConfbridgeList
ActionID: <value>
Conference: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference` - Conference number.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r380028

Asterisk 11 ManagerAction_ConfbridgeListRooms

ConfbridgeListRooms

Synopsis

List active conferences.

Description

Lists data about all active conferences. ConfbridgeListRooms will follow as separate events, followed by a final event called ConfbridgeListRoomsComplete.

Syntax

```
Action: ConfbridgeListRooms
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeLock

ConfbridgeLock

Synopsis

Lock a Confbridge conference.

Description

Syntax

```
Action: ConfbridgeLock
ActionID: <value>
Conference: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeMute

ConfbridgeMute

Synopsis

Mute a Confbridge user.

Description

Syntax

```
Action: ConfbridgeMute
ActionID: <value>
Conference: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`
- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeSetSingleVideoSrc

ConfbridgeSetSingleVideoSrc

Synopsis

Set a conference user as the single video source distributed to all other participants.

Description

Syntax

```
Action: ConfbridgeSetSingleVideoSrc
ActionID: <value>
Conference: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`
- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeStartRecord

ConfbridgeStartRecord

Synopsis

Start recording a Confbridge conference.

Description

Start recording a conference. If recording is already present an error will be returned. If RecordFile is not provided, the default record file specified in the conference's bridge profile will be used, if that is not present either a file will automatically be generated in the monitor directory.

Syntax

```
Action: ConfbridgeStartRecord
ActionID: <value>
Conference: <value>
[RecordFile:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`
- `RecordFile`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeStopRecord

ConfbridgeStopRecord

Synopsis

Stop recording a Confbridge conference.

Description

Syntax

```
Action: ConfbridgeStopRecord
ActionID: <value>
Conference: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeUnlock

ConfbridgeUnlock

Synopsis

Unlock a Confbridge conference.

Description

Syntax

```
Action: ConfbridgeUnlock
ActionID: <value>
Conference: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ConfbridgeUnmute

ConfbridgeUnmute

Synopsis

Unmute a Confbridge user.

Description

Syntax

```
Action: ConfbridgeUnmute
ActionID: <value>
Conference: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference`
- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_CoreSettings

CoreSettings

Synopsis

Show PBX core settings (version etc).

Description

Query for Core PBX settings.

Syntax

```
Action: CoreSettings
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_CoreShowChannels

CoreShowChannels

Synopsis

List currently active channels.

Description

List currently defined channels and some information about them.

Syntax

```
Action: CoreShowChannels
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_CoreStatus

CoreStatus

Synopsis

Show PBX core status variables.

Description

Query for Core PBX status.

Syntax

```
Action: CoreStatus
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_CreateConfig

CreateConfig

Synopsis

Creates an empty file in the configuration directory.

Description

This action will create an empty file in the configuration directory. This action is intended to be used before an UpdateConfig action.

Syntax

```
Action: CreateConfig
ActionID: <value>
Filename: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Filename` - The configuration filename to create (e.g. `foo.conf`).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DAHDIDialOffhook

DAHDIDialOffhook

Synopsis

Dial over DAHDI channel while offhook.

Description

Generate DTMF control frames to the bridged peer.

Syntax

```
Action: DAHDIDialOffhook
ActionID: <value>
DAHDIChannel: <value>
Number: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `DAHDIChannel` - DAHDI channel number to dial digits.
- `Number` - Digits to dial.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DAHDIDNDoff

DAHDIDNDoff

Synopsis

Toggle DAHDI channel Do Not Disturb status OFF.

Description

Equivalent to the CLI command "dahdi set dnd `channel` off".

Note
Feature only supported by analog channels.

Syntax

```
Action: DAHDIDNDoff
ActionID: <value>
DAHDIChannel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `DAHDIChannel` - DAHDI channel number to set DND off.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_DAHDIDNDon

DAHDIDNDon

Synopsis

Toggle DAHDI channel Do Not Disturb status ON.

Description

Equivalent to the CLI command "dahdi set dnd `channel` on".

> **Note**
> Feature only supported by analog channels.

Syntax

```
Action: DAHDIDNDon
ActionID: <value>
DAHDIChannel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `DAHDIChannel` - DAHDI channel number to set DND on.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_DAHDIHangup

DAHDIHangup

Synopsis

Hangup DAHDI Channel.

Description

Simulate an on-hook event by the user connected to the channel.

> **Note**
> Valid only for analog channels.

Syntax

```
Action: DAHDIHangup
ActionID: <value>
DAHDIChannel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `DAHDIChannel` - DAHDI channel number to hangup.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DAHDIRestart

DAHDIRestart

Synopsis

Fully Restart DAHDI channels (terminates calls).

Description

Equivalent to the CLI command "dahdi restart".

Syntax

```
Action: DAHDIRestart
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_DAHDIShowChannels

DAHDIShowChannels

Synopsis

Show status of DAHDI channels.

Description

Similar to the CLI command "dahdi show channels".

Syntax

```
Action: DAHDIShowChannels
ActionID: <value>
DAHDIChannel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `DAHDIChannel` - Specify the specific channel number to show. Show all channels if zero or not present.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_DAHDITransfer

DAHDITransfer

Synopsis

Transfer DAHDI Channel.

Description

Simulate a flash hook event by the user connected to the channel.

Note
Valid only for analog channels.

Syntax

```
Action: DAHDITransfer
ActionID: <value>
DAHDIChannel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `DAHDIChannel` - DAHDI channel number to transfer.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DataGet

DataGet

Synopsis

Retrieve the data api tree.

Description

Retrieve the data api tree.

Syntax

```
Action: DataGet
ActionID: <value>
Path: <value>
Search: <value>
Filter: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Path`
- `Search`
- `Filter`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DBDel

DBDel

Synopsis

Delete DB entry.

Description

Syntax

```
Action: DBDel
ActionID: <value>
Family: <value>
Key: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Family`
- `Key`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DBDelTree

DBDelTree

Synopsis

Delete DB Tree.

Description

Syntax

```
Action: DBDelTree
ActionID: <value>
Family: <value>
Key: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Family`
- `Key`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DBGet

DBGet

Synopsis

Get DB Entry.

Description

Syntax

```
Action: DBGet
ActionID: <value>
Family: <value>
Key: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Family`
- `Key`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_DBPut

DBPut

Synopsis

Put DB entry.

Description

Syntax

```
Action: DBPut
ActionID: <value>
Family: <value>
Key: <value>
Val: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Family`
- `Key`
- `Val`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Events

Events

Synopsis

Control Event Flow.

Description

Enable/Disable sending of events to this manager client.

Syntax

```
Action: Events
ActionID: <value>
EventMask: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `EventMask`
    - `on` - If all events should be sent.
    - `off` - If no events should be sent.
    - `system,call,log,...` - To select which flags events should have to be sent.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ExtensionState

ExtensionState

Synopsis

Check Extension Status.

Description

Report the extension state for given extension. If the extension has a hint, will use devicestate to check the status of the device connected to the extension.

Will return an `Extension Status` message. The response will include the hint for the extension and the status.

Syntax

```
Action: ExtensionState
ActionID: <value>
Exten: <value>
Context: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Exten` - Extension to check state on.
- `Context` - Context for extension.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Filter

Filter

Synopsis

Dynamically add filters for the current manager session.

Description

The filters added are only used for the current session. Once the connection is closed the filters are removed.

This comand requires the system permission because this command can be used to create filters that may bypass filters defined in manager.conf

Syntax

```
Action: Filter
ActionID: <value>
Operation: <value>
Filter: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Operation`
    - `Add` - Add a filter.
- `Filter` - Filters can be whitelist or blacklist
  Example whitelist filter: "Event: Newchannel"
  Example blacklist filter: "!Channel: DAHDI.*"
  This filter option is used to whitelist or blacklist events per user to be reported with regular expressions and are allowed if both the regex matches and the user has read access as defined in manager.conf. Filters are assumed to be for whitelisting unless preceeded by an exclamation point, which marks it as being black. Evaluation of the filters is as follows:

- If no filters are configured all events are reported as normal.
- If there are white filters only: implied black all filter processed first, then white filters.
- If there are black filters only: implied white all filter processed first, then black filters.
- If there are both white and black filters: implied black all filter processed first, then white filters, and lastly black filters.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_FilterList

FilterList

Synopsis

Show current event filters for this session

Description

The filters displayed are for the current session. Only those filters defined in manager.conf will be present upon starting a new session.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_GetConfig

GetConfig

Synopsis

Retrieve configuration.

Description

This action will dump the contents of a configuration file by category and contents or optionally by specified category only.

Syntax

```
Action: GetConfig
ActionID: <value>
Filename: <value>
Category: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Filename` - Configuration filename (e.g. `foo.conf`).
- `Category` - Category in configuration file.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_GetConfigJSON

GetConfigJSON

Synopsis

Retrieve configuration (JSON format).

Description

This action will dump the contents of a configuration file by category and contents in JSON format. This only makes sense to be used using rawman over the HTTP interface.

Syntax

```
Action: GetConfigJSON
ActionID: <value>
Filename: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Filename` - Configuration filename (e.g. `foo.conf`).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Getvar

Getvar

Synopsis

Gets a channel variable.

Description

Get the value of a global or local channel variable.

> **Note**
> If a channel name is not provided then the variable is global.

Syntax

```
Action: Getvar
ActionID: <value>
Channel: <value>
Variable: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel to read variable from.
- `Variable` - Variable name.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Hangup

Hangup

Synopsis

Hangup channel.

Description

Hangup a channel.

Syntax

```
Action: Hangup
ActionID: <value>
Channel: <value>
Cause: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The exact channel name to be hungup, or to use a regular expression, set this parameter to: /regex/
  Example exact channel: SIP/provider-0000012a
  Example regular expression: /^SIP/provider-.*$/
- `Cause` - Numeric hangup cause.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_IAXnetstats

IAXnetstats

Synopsis

Show IAX Netstats.

Description

Show IAX channels network statistics.

Syntax

```
Action: IAXnetstats
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_IAXpeerlist

IAXpeerlist

Synopsis

List IAX Peers.

Description

List all the IAX peers.

Syntax

```
Action: IAXpeerlist
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_IAXpeers

IAXpeers

Synopsis

List IAX peers.

Description

Syntax

```
Action: IAXpeers
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_IAXregistry

IAXregistry

Synopsis

Show IAX registrations.

Description

Show IAX registrations.

Syntax

```
Action: IAXregistry
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_JabberSend

Moved to Asterisk 11 Application_JabberSend_res_xmpp

Asterisk 11 ManagerAction_JabberSend_res_jabber

JabberSend - [res_jabber]

Synopsis

Sends a message to a Jabber Client.

Description

Sends a message to a Jabber Client.

Syntax

```
Action: JabberSend
ActionID: <value>
Jabber: <value>
JID: <value>
Message: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Jabber` - Client or transport Asterisk uses to connect to JABBER.
- `JID` - XMPP/Jabber JID (Name) of recipient.
- `Message` - Message to be sent to the buddy.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_JabberSend_res_xmpp

JabberSend - [res_xmpp]

Synopsis

Sends a message to a Jabber Client.

Description

Sends a message to a Jabber Client.

Syntax

```
Action: JabberSend
ActionID: <value>
Jabber: <value>
JID: <value>
Message: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Jabber` - Client or transport Asterisk uses to connect to JABBER.
- `JID` - XMPP/Jabber JID (Name) of recipient.
- `Message` - Message to be sent to the buddy.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ListCategories

ListCategories

Synopsis

List categories in configuration file.

Description

This action will dump the categories in a given file.

Syntax

```
Action: ListCategories
ActionID: <value>
Filename: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Filename` - Configuration filename (e.g. `foo.conf`).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ListCommands

ListCommands

Synopsis

List available manager commands.

Description

Returns the action name and synopsis for every action that is available to the user.

Syntax

```
Action: ListCommands
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_LocalOptimizeAway

LocalOptimizeAway

Synopsis

Optimize away a local channel when possible.

Description

A local channel created with "/n" will not automatically optimize away. Calling this command on the local channel will clear that flag and allow it to optimize away if it's bridged or when it becomes bridged.

Syntax

```
Action: LocalOptimizeAway
ActionID: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The channel name to optimize away.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_Login

Login

Synopsis

Login Manager.

Description

Login Manager.

Syntax

```
Action: Login
ActionID: <value>
Username: <value>
Secret: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Username` - Username to login with as specified in manager.conf.
- `Secret` - Secret to login with as specified in manager.conf.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Logoff

Logoff

Synopsis

Logoff Manager.

Description

Logoff the current manager session.

Syntax

```
Action: Logoff
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MailboxCount

MailboxCount

Synopsis

Check Mailbox Message Count.

Description

Checks a voicemail account for new messages.

Returns number of urgent, new and old messages.

Message: Mailbox Message Count

Mailbox: *mailboxid*

UrgentMessages: *count*

NewMessages: *count*

OldMessages: *count*

Syntax

```
Action: MailboxCount
ActionID: <value>
Mailbox: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Mailbox` - Full mailbox ID *mailbox@vm-context*.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MailboxStatus

MailboxStatus

Synopsis

Check mailbox.

Description

Checks a voicemail account for status.

Returns whether there are messages waiting.

Message: Mailbox Status.

Mailbox: *mailboxid*.

Waiting: `0` if messages waiting, `1` if no messages waiting.

Syntax

```
Action: MailboxStatus
ActionID: <value>
Mailbox: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Mailbox` - Full mailbox ID *mailbox@vm-context*.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MeetmeList

MeetmeList

Synopsis

List participants in a conference.

Description

Lists all users in a particular MeetMe conference. MeetmeList will follow as separate events, followed by a final event called MeetmeListComplete.

Syntax

```
Action: MeetmeList
ActionID: <value>
[Conference:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Conference` - Conference number.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MeetmeListRooms

MeetmeListRooms

Synopsis

List active conferences.

Description

Lists data about all active conferences. MeetmeListRooms will follow as separate events, followed by a final event called MeetmeListRoomsComplete.

Syntax

```
Action: MeetmeListRooms
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MeetmeMute

MeetmeMute

Synopsis

Mute a Meetme user.

Description

Syntax

```
Action: MeetmeMute
ActionID: <value>
Meetme: <value>
Usernum: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Meetme`
- `Usernum`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MeetmeUnmute

MeetmeUnmute

Synopsis

Unmute a Meetme user.

Description

Syntax

```
Action: MeetmeUnmute
ActionID: <value>
Meetme: <value>
Usernum: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Meetme`
- `Usernum`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MessageSend

MessageSend

Synopsis

Send an out of call message to an endpoint.

Syntax

```
Action: MessageSend
ActionID: <value>
To: <value>
From: <value>
Body: <value>
Base64Body: <value>
Variable: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `To` - The URI the message is to be sent to.

Technology: SIP
Specifying a prefix of `sip:` will send the message as a SIP MESSAGE request.

Technology: XMPP
Specifying a prefix of `xmpp:` will send the message as an XMPP chat message.

- `From` - A From URI for the message if needed for the message technology being used to send this message.

Technology: SIP
The `from` parameter can be a configured peer name or in the form of "display-name" <URI>.

Technology: XMPP
Specifying a prefix of `xmpp:` will specify the account defined in `xmpp.conf` to send the message from. Note that this field is required for XMPP messages.

- `Body` - The message body text. This must not contain any newlines as that conflicts with the AMI protocol.
- `Base64Body` - Text bodies requiring the use of newlines have to be base64 encoded in this field. Base64Body will be decoded before being sent out. Base64Body takes precedence over Body.
- `Variable` - Message variable to set, multiple Variable: headers are allowed. The header value is a comma separated list of name=value pairs.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_MixMonitor

MixMonitor

Synopsis

Record a call and mix the audio during the recording. Use of StopMixMonitor is required to guarantee the audio file is available for processing during dialplan execution.

Description

This action records the audio on the current channel to the specified file.

- `MIXMONITOR_FILENAME` - Will contain the filename used to record the mixed stream.

Syntax

```
Action: MixMonitor
ActionID: <value>
Channel: <value>
File: <value>
options: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Used to specify the channel to record.
- `File` - Is the name of the file created in the monitor spool directory. Defaults to the same name as the channel (with slashes replaced with dashes). This argument is optional if you specify to record unidirectional audio with either the r(filename) or t(filename) options in the options field. If neither MIXMONITOR_FILENAME or this parameter is set, the mixed stream won't be recorded.
- `options` - Options that apply to the MixMonitor in the same way as they would apply if invoked from the MixMonitor application. For a list of available options, see the documentation for the mixmonitor application.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MixMonitorMute

MixMonitorMute

Synopsis

Mute / unMute a Mixmonitor recording.

Description

This action may be used to mute a MixMonitor recording.

Syntax

```
Action: MixMonitorMute
ActionID: <value>
Channel: <value>
Direction: <value>
State: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Used to specify the channel to mute.
- `Direction` - Which part of the recording to mute: read, write or both (from channel, to channel or both channels).
- `State` - Turn mute on or off : 1 to turn on, 0 to turn off.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ModuleCheck

ModuleCheck

Synopsis

Check if module is loaded.

Description

Checks if Asterisk module is loaded. Will return Success/Failure. For success returns, the module revision number is included.

Syntax

```
Action: ModuleCheck
Module: <value>
```

Arguments

- `Module` - Asterisk module name (not including extension).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ModuleLoad

ModuleLoad

Synopsis

Module management.

Description

Loads, unloads or reloads an Asterisk module in a running system.

Syntax

```
Action: ModuleLoad
ActionID: <value>
Module: <value>
LoadType: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Module` - Asterisk module name (including .so extension) or subsystem identifier:
    - `cdr`
    - `dnsmgr`
    - `extconfig`
    - `enum`
    - `acl`
    - `manager`
    - `http`
    - `logger`
    - `features`
    - `dsp`
    - `udptl`
    - `indications`
    - `cel`
    - `plc`
- `LoadType` - The operation to be done on module. Subsystem identifiers may only be reloaded.
    - `load`
    - `unload`
    - `reload`
      If no module is specified for a `reload` loadtype, all modules are reloaded.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Monitor

Monitor

Synopsis

Monitor a channel.

Description

This action may be used to record the audio on a specified channel.

Syntax

```
Action: Monitor
ActionID: <value>
Channel: <value>
File: <value>
Format: <value>
Mix: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Used to specify the channel to record.
- `File` - Is the name of the file created in the monitor spool directory. Defaults to the same name as the channel (with slashes replaced with dashes).
- `Format` - Is the audio recording format. Defaults to `wav`.
- `Mix` - Boolean parameter as to whether to mix the input and output channels together after the recording is finished.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_MuteAudio

MuteAudio

Synopsis

Mute an audio stream.

Description

Mute an incoming or outgoing audio stream on a channel.

Syntax

```
Action: MuteAudio
ActionID: <value>
Channel: <value>
Direction: <value>
State: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The channel you want to mute.
- `Direction`
    - `in` - Set muting on inbound audio stream. (to the PBX)
    - `out` - Set muting on outbound audio stream. (from the PBX)
    - `all` - Set muting on inbound and outbound audio streams.
- `State`
    - `on` - Turn muting on.
    - `off` - Turn muting off.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Originate

Originate

Synopsis

Originate a call.

Description

Generates an outgoing call to a *Extension*/*Context*/*Priority* or *Application*/*Data*

Syntax

```
Action: Originate
ActionID: <value>
Channel: <value>
Exten: <value>
Context: <value>
Priority: <value>
Application: <value>
Data: <value>
Timeout: <value>
CallerID: <value>
Variable: <value>
Account: <value>
EarlyMedia: <value>
Async: <value>
Codecs: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to call.
- `Exten` - Extension to use (requires `Context` and `Priority`)
- `Context` - Context to use (requires `Exten` and `Priority`)
- `Priority` - Priority to use (requires `Exten` and `Context`)
- `Application` - Application to execute.
- `Data` - Data to use (requires `Application`).
- `Timeout` - How long to wait for call to be answered (in ms.).
- `CallerID` - Caller ID to be set on the outgoing channel.
- `Variable` - Channel variable to set, multiple Variable: headers are allowed.
- `Account` - Account code.
- `EarlyMedia` - Set to `true` to force call bridge on early media..
- `Async` - Set to `true` for fast origination.
- `Codecs` - Comma-separated list of codecs to use for this call.

See Also

- Asterisk 11 ManagerEvent_OriginateResponse

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Park

Park

Synopsis

Park a channel.

Description

Park a channel.

Syntax

```
Action: Park
ActionID: <value>
Channel: <value>
Channel2: <value>
Timeout: <value>
Parkinglot: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to park.
- `Channel2` - Channel to return to if timeout.
- `Timeout` - Number of milliseconds to wait before callback.
- `Parkinglot` - Specify in which parking lot to park the channel.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ParkedCalls

ParkedCalls

Synopsis

List parked calls.

Description

List parked calls.

Syntax

```
Action: ParkedCalls
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Parkinglots

Parkinglots

Synopsis

Get a list of parking lots

Description

List all parking lots as a series of AMI events

Syntax

```
Action: Parkinglots
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_PauseMonitor

PauseMonitor

Synopsis

Pause monitoring of a channel.

Description

This action may be used to temporarily stop the recording of a channel.

Syntax

```
Action: PauseMonitor
ActionID: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Used to specify the channel to record.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Ping

Ping

Synopsis

Keepalive command.

Description

A 'Ping' action will ellicit a 'Pong' response. Used to keep the manager connection open.

Syntax

```
Action: Ping
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_PlayDTMF

PlayDTMF

Synopsis

Play DTMF signal on a specific channel.

Description

Plays a dtmf digit on the specified channel.

Syntax

```
Action: PlayDTMF
ActionID: <value>
Channel: <value>
Digit: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to send digit to.
- `Digit` - The DTMF digit to play.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_PresenceState

PresenceState

Synopsis

Check Presence State

Description

Report the presence state for the given presence provider.

Will return a `Presence State` message. The response will include the presence state and, if set, a presence subtype and custom message.

Syntax

```
Action: PresenceState
ActionID: <value>
Provider: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Provider` - Presence Provider to check the state of

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_PRIShowSpans

PRIShowSpans

Synopsis

Show status of PRI spans.

Description

Similar to the CLI command "pri show spans".

Syntax

```
Action: PRIShowSpans
ActionID: <value>
Span: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Span` - Specify the specific span to show. Show all spans if zero or not present.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 ManagerAction_QueueAdd

QueueAdd

Synopsis

Add interface to queue.

Description

Syntax

```
Action: QueueAdd
ActionID: <value>
Queue: <value>
Interface: <value>
Penalty: <value>
Paused: <value>
MemberName: <value>
StateInterface: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`
- `Interface`
- `Penalty`
- `Paused`
- `MemberName`
- `StateInterface`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueLog

QueueLog

Synopsis

Adds custom entry in queue_log.

Description

Syntax

```
Action: QueueLog
ActionID: <value>
Queue: <value>
Event: <value>
Uniqueid: <value>
Interface: <value>
Message: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`
- `Event`
- `Uniqueid`
- `Interface`
- `Message`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueMemberRingInUse

QueueMemberRingInUse

Synopsis

Set the ringinuse value for a queue member.

Description

Syntax

```
Action: QueueMemberRingInUse
ActionID: <value>
Interface: <value>
RingInUse: <value>
Queue: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Interface`
- `RingInUse`
- `Queue`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueuePause

QueuePause

Synopsis

Makes a queue member temporarily unavailable.

Description

Syntax

```
Action: QueuePause
ActionID: <value>
Interface: <value>
Paused: <value>
Queue: <value>
Reason: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Interface`
- `Paused`
- `Queue`
- `Reason`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueuePenalty

QueuePenalty

Synopsis

Set the penalty for a queue member.

Description

Syntax

```
Action: QueuePenalty
ActionID: <value>
Interface: <value>
Penalty: <value>
Queue: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Interface`
- `Penalty`
- `Queue`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueReload

QueueReload

Synopsis

Reload a queue, queues, or any sub-section of a queue or queues.

Description

Syntax

```
Action: QueueReload
ActionID: <value>
Queue: <value>
Members: <value>
Rules: <value>
Parameters: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`
- `Members`
    - `yes`
    - `no`
- `Rules`
    - `yes`
    - `no`
- `Parameters`
    - `yes`
    - `no`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueRemove

QueueRemove

Synopsis

Remove interface from queue.

Description

Syntax

```
Action: QueueRemove
ActionID: <value>
Queue: <value>
Interface: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`
- `Interface`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueReset

QueueReset

Synopsis

Reset queue statistics.

Description

Syntax

```
Action: QueueReset
ActionID: <value>
Queue: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueRule

QueueRule

Synopsis

Queue Rules.

Description

Syntax

```
Action: QueueRule
ActionID: <value>
Rule: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Rule`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Queues

Queues

Synopsis

Queues.

Description

Syntax

```
Action: Queues
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueStatus

QueueStatus

Synopsis

Show queue status.

Description

Syntax

```
Action: QueueStatus
ActionID: <value>
Queue: <value>
Member: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`
- `Member`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_QueueSummary

QueueSummary

Synopsis

Show queue summary.

Description

Syntax

```
Action: QueueSummary
ActionID: <value>
Queue: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Redirect

Redirect

Synopsis

Redirect (transfer) a call.

Description

Redirect (transfer) a call.

Syntax

```
Action: Redirect
ActionID: <value>
Channel: <value>
ExtraChannel: <value>
Exten: <value>
ExtraExten: <value>
Context: <value>
ExtraContext: <value>
Priority: <value>
ExtraPriority: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel to redirect.
- `ExtraChannel` - Second call leg to transfer (optional).
- `Exten` - Extension to transfer to.
- `ExtraExten` - Extension to transfer extrachannel to (optional).
- `Context` - Context to transfer to.
- `ExtraContext` - Context to transfer extrachannel to (optional).
- `Priority` - Priority to transfer to.
- `ExtraPriority` - Priority to transfer extrachannel to (optional).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Reload

Reload

Synopsis

Send a reload event.

Description

Send a reload event.

Syntax

```
Action: Reload
ActionID: <value>
Module: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Module` - Name of the module to reload.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SendText

SendText

Synopsis

Send text message to channel.

Description

Sends A Text Message to a channel while in a call.

Syntax

```
Action: SendText
ActionID: <value>
Channel: <value>
Message: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel to send message to.
- `Message` - Message to send.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Setvar

Setvar

Synopsis

Set a channel variable.

Description

Set a global or local channel variable.

> **Note**
> If a channel name is not provided then the variable is global.

Syntax

```
Action: Setvar
ActionID: <value>
Channel: <value>
Variable: <value>
Value: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel to set variable for.
- `Variable` - Variable name.
- `Value` - Variable value.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_ShowDialPlan

ShowDialPlan

Synopsis

Show dialplan contexts and extensions

Description

Show dialplan contexts and extensions. Be aware that showing the full dialplan may take a lot of capacity.

Syntax

```
Action: ShowDialPlan
ActionID: <value>
Extension: <value>
Context: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Extension` - Show a specific extension.
- `Context` - Show a specific context.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SIPnotify

SIPnotify

Synopsis

Send a SIP notify.

Description

Sends a SIP Notify event.

All parameters for this event must be specified in the body of this request via multiple `Variable: name=value` sequences.

Syntax

```
Action: SIPnotify
ActionID: <value>
Channel: <value>
Variable: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Peer to receive the notify.
- `Variable` - At least one variable pair must be specified. *name*=*value*

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SIPpeers

SIPpeers

Synopsis

List SIP peers (text format).

Description

Lists SIP peers in text format with details on current status. `Peerlist` will follow as separate events, followed by a final event called `PeerlistComplete`.

Syntax

```
Action: SIPpeers
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SIPpeerstatus

SIPpeerstatus

Synopsis

Show the status of one or all of the sip peers.

Description

Retrieves the status of one or all of the sip peers. If no peer name is specified, status for all of the sip peers will be retrieved.

Syntax

```
Action: SIPpeerstatus
ActionID: <value>
[Peer:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Peer` - The peer name you want to check.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SIPqualifypeer

SIPqualifypeer

Synopsis

Qualify SIP peers.

Description

Qualify a SIP peer.

Syntax

```
Action: SIPqualifypeer
ActionID: <value>
Peer: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Peer` - The peer name you want to qualify.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SIPshowpeer

SIPshowpeer

Synopsis

show SIP peer (text format).

Description

Show one SIP peer with details on current status.

Syntax

```
Action: SIPshowpeer
ActionID: <value>
Peer: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Peer` - The peer name you want to check.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SIPshowregistry

SIPshowregistry

Synopsis

Show SIP registrations (text format).

Description

Lists all registration requests and status. Registrations will follow as separate events followed by a final event called `RegistrationsComplete`.

Syntax

```
Action: SIPshowregistry
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SKINNYdevices

SKINNYdevices

Synopsis

List SKINNY devices (text format).

Description

Lists Skinny devices in text format with details on current status. Devicelist will follow as separate events, followed by a final event called DevicelistComplete.

Syntax

```
Action: SKINNYdevices
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SKINNYlines

SKINNYlines

Synopsis

List SKINNY lines (text format).

Description

Lists Skinny lines in text format with details on current status. Linelist will follow as separate events, followed by a final event called LinelistComplete.

Syntax

```
Action: SKINNYlines
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SKINNYshowdevice

SKINNYshowdevice

Synopsis

Show SKINNY device (text format).

Description

Show one SKINNY device with details on current status.

Syntax

```
Action: SKINNYshowdevice
ActionID: <value>
Device: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Device` - The device name you want to check.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_SKINNYshowline

SKINNYshowline

Synopsis

Show SKINNY line (text format).

Description

Show one SKINNY line with details on current status.

Syntax

```
Action: SKINNYshowline
ActionID: <value>
Line: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Line` - The line name you want to check.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_Status

Status

Synopsis

List channel status.

Description

Will return the status information of each channel along with the value for the specified channel variables.

Syntax

```
Action: Status
ActionID: <value>
Channel: <value>
Variables: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The name of the channel to query for status.
- `Variables` - Comma `,` separated list of variable to include.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_StopMixMonitor

StopMixMonitor

Synopsis

Stop recording a call through MixMonitor, and free the recording's file handle.

Description

This action stops the audio recording that was started with the `MixMonitor` action on the current channel.

Syntax

```
Action: StopMixMonitor
ActionID: <value>
Channel: <value>
[MixMonitorID:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The name of the channel monitored.
- `MixMonitorID` - If a valid ID is provided, then this command will stop only that specific MixMonitor.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_StopMonitor

StopMonitor

Synopsis

Stop monitoring a channel.

Description

This action may be used to end a previously started 'Monitor' action.

Syntax

```
Action: StopMonitor
ActionID: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The name of the channel monitored.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_UnpauseMonitor

UnpauseMonitor

Synopsis

Unpause monitoring of a channel.

Description

This action may be used to re-enable recording of a channel after calling PauseMonitor.

Syntax

```
Action: UnpauseMonitor
ActionID: <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Used to specify the channel to record.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_UpdateConfig

UpdateConfig

Synopsis

Update basic configuration.

Description

This action will modify, create, or delete configuration elements in Asterisk configuration files.

Syntax

```
Action: UpdateConfig
ActionID: <value>
SrcFilename: <value>
DstFilename: <value>
Reload: <value>
Action-XXXXXX: <value>
Cat-XXXXXX: <value>
Var-XXXXXX: <value>
Value-XXXXXX: <value>
Match-XXXXXX: <value>
Line-XXXXXX: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `SrcFilename` - Configuration filename to read (e.g. `foo.conf`).
- `DstFilename` - Configuration filename to write (e.g. `foo.conf`)
- `Reload` - Whether or not a reload should take place (or name of specific module).
- `Action-XXXXXX` - Action to take.
  X's represent 6 digit number beginning with 000000.
  - `NewCat`
  - `RenameCat`
  - `DelCat`
  - `EmptyCat`
  - `Update`
  - `Delete`
  - `Append`
  - `Insert`
- `Cat-XXXXXX` - Category to operate on.
  X's represent 6 digit number beginning with 000000.
- `Var-XXXXXX` - Variable to work on.
  X's represent 6 digit number beginning with 000000.
- `Value-XXXXXX` - Value to work on.
  X's represent 6 digit number beginning with 000000.
- `Match-XXXXXX` - Extra match required to match line.
  X's represent 6 digit number beginning with 000000.
- `Line-XXXXXX` - Line in category to operate on (used with delete and insert actions).
  X's represent 6 digit number beginning with 000000.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_UserEvent

UserEvent

Synopsis

Send an arbitrary event.

Description

Send an event to manager sessions.

Syntax

```
Action: UserEvent
ActionID: <value>
UserEvent: <value>
Header1: <value>
HeaderN: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `UserEvent` - Event string to send.
- `Header1` - Content1.
- `HeaderN` - ContentN.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_VoicemailUsersList

VoicemailUsersList

Synopsis

List All Voicemail User Information.

Description

Syntax

```
Action: VoicemailUsersList
ActionID: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerAction_WaitEvent

WaitEvent

Synopsis

Wait for an event to occur.

Description

This action will ellicit a `Success` response. Whenever a manager event is queued. Once WaitEvent has been called on an HTTP manager session, events will be generated and queued.

Syntax

```
Action: WaitEvent
ActionID: <value>
Timeout: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Timeout` - Maximum time (in seconds) to wait for events, `-1` means forever.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 AMI Events

Asterisk 11 ManagerEvent_AgentCalled

AgentCalled

Synopsis

Raised when an Agent is notified of a member in the queue.

Syntax

```
Event: AgentCalled
Queue: <value>
AgentCalled: <value>
AgentName: <value>
[Variable:] <value>
Queue: <value>
ChannelCalling: <value>
DestinationChannel: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
Context: <value>
Extension: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Queue` - The name of the queue.
- `AgentCalled` - The agent's technology or location.
- `AgentName` - The name of the agent.
- `Variable` - Optional channel variables from the ChannelCalling channel
- `Queue`
- `ChannelCalling`
- `DestinationChannel`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `Context`
- `Extension`
- `Priority`
- `Uniqueid`

See Also

- Asterisk 11 ManagerEvent_AgentRingNoAnswer
- Asterisk 11 ManagerEvent_AgentComplete
- Asterisk 11 ManagerEvent_AgentConnect

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_AgentComplete

AgentComplete

Synopsis

Raised when an agent has finished servicing a member in the queue.

Syntax

```
Event: AgentComplete
Queue: <value>
Member: <value>
MemberName: <value>
HoldTime: <value>
[Variable:] <value>
TalkTime: <value>
Reason: <value>
Queue: <value>
Uniqueid: <value>
Channel: <value>
Member: <value>
MemberName: <value>
HoldTime: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Member` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `HoldTime` - The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Variable` - Optional channel variables from the ChannelCalling channel
- `TalkTime` - The time the agent talked with the member in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Reason`
    - `caller`
    - `agent`
    - `transfer`
- `Queue`
- `Uniqueid`
- `Channel`
- `Member`
- `MemberName`
- `HoldTime`

See Also

- Asterisk 11 ManagerEvent_AgentCalled
- Asterisk 11 ManagerEvent_AgentConnect

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_AgentConnect

AgentConnect

Synopsis

Raised when an agent answers and is bridged to a member in the queue.

Syntax

```
Event: AgentConnect
Queue: <value>
Member: <value>
MemberName: <value>
RingTime: <value>
HoldTime: <value>
[Variable:] <value>
Queue: <value>
Uniqueid: <value>
Channel: <value>
Member: <value>
MemberName: <value>
HoldTime: <value>
BridgedChannel: <value>
RingTime: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Member` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `RingTime` - The time the agent was rung, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `HoldTime` - The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Variable` - Optional channel variables from the ChannelCalling channel
- `Queue`
- `Uniqueid`
- `Channel`
- `Member`
- `MemberName`
- `HoldTime`
- `BridgedChannel`
- `RingTime`

See Also

- Asterisk 11 ManagerEvent_AgentCalled
- Asterisk 11 ManagerEvent_AgentComplete
- Asterisk 11 ManagerEvent_AgentDump

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_AgentDump

AgentDump

Synopsis

Raised when an agent hangs up on a member in the queue.

Syntax

```
Event: AgentDump
Queue: <value>
Member: <value>
MemberName: <value>
[Variable:] <value>
Queue: <value>
Uniqueid: <value>
Channel: <value>
Member: <value>
MemberName: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Member` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `Variable` - Optional channel variables from the ChannelCalling channel
- `Queue`
- `Uniqueid`
- `Channel`
- `Member`
- `MemberName`

See Also

- Asterisk 11 ManagerEvent_AgentCalled
- Asterisk 11 ManagerEvent_AgentConnect

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Agentlogin

Agentlogin

Synopsis

Raised when an Agent has logged in.

Syntax

```
Event: Agentlogin
Agent: <value>
Channel: <value>
Uniqueid: <value>
```

Arguments

- `Agent` - The name of the agent.
- `Channel`
- `Uniqueid`

See Also

- Asterisk 11 Application_AgentLogin
- Asterisk 11 ManagerEvent_Agentlogoff

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Agentlogoff

Agentlogoff

Synopsis

Raised when an Agent has logged off.

Syntax

```
Event: Agentlogoff
Agent: <value>
Agent: <value>
Logintime: <value>
Uniqueid: <value>
```

Arguments

- `Agent` - The name of the agent.
- `Agent`
- `Logintime`
- `Uniqueid`

See Also

- Asterisk 11 ManagerEvent_Agentlogin

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_AgentRingNoAnswer

AgentRingNoAnswer

Synopsis

Raised when an agent is notified of a member in the queue and fails to answer.

Syntax

```
Event: AgentRingNoAnswer
Queue: <value>
MemberName: <value>
[Variable:] <value>
Member: <value>
RingTime: <value>
Queue: <value>
Uniqueid: <value>
Channel: <value>
MemberName: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Variable` - Optional channel variables from the ChannelCalling channel
- `Member` - The queue member's channel technology or location.
- `RingTime` - The time the agent was rung, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Queue`
- `Uniqueid`
- `Channel`
- `MemberName`

See Also

- Asterisk 11 ManagerEvent_AgentCalled

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Alarm

Alarm

Synopsis

Raised when an alarm is set on a DAHDI channel.

Syntax

```
Event: Alarm
Alarm: <value>
Channel: <value>
```

Arguments

- `Alarm`
- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_AlarmClear

AlarmClear

Synopsis

Raised when an alarm is cleared on a DAHDI channel.

Syntax

```
Event: AlarmClear
Channel: <value>
```

Arguments

- `Channel`

Synopsis

Raised when an Alarm is cleared on an Analog channel.

Syntax

```
Event: AlarmClear
Channel: <value>
```

Arguments

- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Bridge

Bridge

Synopsis

Raised when a bridge changes between two channels.

Syntax

```
Event: Bridge
Bridgestate: <value>
Bridgetype: <value>
Channel1: <value>
Channel2: <value>
Uniqueid1: <value>
Uniqueid2: <value>
CallerID1: <value>
CallerID2: <value>
```

Arguments

- `Bridgestate`
  - `Link`
  - `Unlink`
- `Bridgetype`
  - `core`
  - `native`
- `Channel1`
- `Channel2`
- `Uniqueid1`
- `Uniqueid2`
- `CallerID1`
- `CallerID2`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_BridgeAction

BridgeAction

Synopsis

Raised when a bridge is successfully created due to a manager action.

Syntax

```
Event: BridgeAction
Response: <value>
Channel1: <value>
Channel2: <value>
```

Arguments

- `Response`
  - `Success`
  - `Failed`
- `Channel1`
- `Channel2`

See Also

- Asterisk 11 ManagerAction_Bridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_BridgeExec

BridgeExec

Synopsis

Raised when an error occurs during bridge creation.

Syntax

```
Event: BridgeExec
Response: <value>
Reason: <value>
Channel1: <value>
Channel2: <value>
```

Arguments

- `Response`
- `Reason`
- `Channel1`
- `Channel2`

See Also

- Asterisk 11 Application_Bridge

Synopsis

Raised when the bridge is created successfully.

Syntax

```
Event: BridgeExec
Response: <value>
Channel1: <value>
Channel2: <value>
```

Arguments

- `Response`
- `Channel1`
- `Channel2`

See Also

- Asterisk 11 Application_Bridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ChanSpyStart

ChanSpyStart

Synopsis

Raised when a channel has started spying on another channel.

Syntax

```
Event: ChanSpyStart
SpyerChannel: <value>
SpyeeChannel: <value>
```

Arguments

- `SpyerChannel`
- `SpyeeChannel`

See Also

- Asterisk 11 Application_ChanSpy
- Asterisk 11 Application_ExtenSpy
- Asterisk 11 ManagerEvent_ChanSpyStop

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ChanSpyStop

ChanSpyStop

Synopsis

Raised when a channel has stopped spying on another channel.

Syntax

```
Event: ChanSpyStop
SpyeeChannel: <value>
```

Arguments

- `SpyeeChannel`

See Also

- Asterisk 11 ManagerEvent_ChanSpyStart

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ConfbridgeEnd

ConfbridgeEnd

Synopsis

Raised when a conference ends.

Syntax

```
Event: ConfbridgeEnd
Conference: <value>
Conference: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `Conference`

See Also

- Asterisk 11 ManagerEvent_ConfbridgeStart
- Asterisk 11 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ConfbridgeJoin

ConfbridgeJoin

Synopsis

Raised when a channel joins a Confbridge conference.

Syntax

```
Event: ConfbridgeJoin
Conference: <value>
Channel: <value>
Uniqueid: <value>
Conference: <value>
CallerIDnum: <value>
CallerIDname: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `Channel`
- `Uniqueid`
- `Conference`
- `CallerIDnum`
- `CallerIDname`

See Also

- Asterisk 11 ManagerEvent_ConfbridgeLeave
- Asterisk 11 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ConfbridgeLeave

ConfbridgeLeave

Synopsis

Raised when a channel leaves a Confbridge conference.

Syntax

```
Event: ConfbridgeLeave
Conference: <value>
Channel: <value>
Uniqueid: <value>
Conference: <value>
CallerIDnum: <value>
CallerIDname: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `Channel`
- `Uniqueid`
- `Conference`
- `CallerIDnum`
- `CallerIDname`

See Also

- Asterisk 11 ManagerEvent_ConfbridgeJoin

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ConfbridgeStart

ConfbridgeStart

Synopsis

Raised when a conference starts.

Syntax

```
Event: ConfbridgeStart
Conference: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.

See Also

- Asterisk 11 ManagerEvent_ConfbridgeEnd

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ConfbridgeTalking

ConfbridgeTalking

Synopsis

Raised when a conference participant has started or stopped talking.

Syntax

```
Event: ConfbridgeTalking
Conference: <value>
TalkingStatus: <value>
Channel: <value>
Uniqueid: <value>
Conference: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `TalkingStatus`
    - `on`
    - `off`
- `Channel`
- `Uniqueid`
- `Conference`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_DAHDIChannel

DAHDIChannel

Synopsis

Raised when a DAHDI channel is created or an underlying technology is associated with a DAHDI channel.

Syntax

```
Event: DAHDIChannel
Channel: <value>
Uniqueid: <value>
DAHDISpan: <value>
DAHDIChannel: <value>
```

Arguments

- `Channel`
- `Uniqueid`
- `DAHDISpan`
- `DAHDIChannel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Dial

Dial

Synopsis

Raised when a dial action has started.

Syntax

```
Event: Dial
SubEvent: <value>
Channel: <value>
Destination: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
UniqueID: <value>
DestUniqueID: <value>
Dialstring: <value>
```

Arguments

- `SubEvent` - A sub event type, specifying whether the dial action has begun or ended.
    - `Begin`
    - `End`
- `Channel`
- `Destination`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `UniqueID`
- `DestUniqueID`
- `Dialstring`

Synopsis

Raised when a dial action has ended.

Syntax

```
Event: Dial
DialStatus: <value>
SubEvent: <value>
Channel: <value>
UniqueID: <value>
```

Arguments

- `DialStatus` - The value of the `DIALSTATUS` channel variable.
- `SubEvent` - A sub event type, specifying whether the dial action has begun or ended.
    - `Begin`
    - `End`
- `Channel`
- `UniqueID`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_DNDState

DNDState

Synopsis

Raised when the Do Not Disturb state is changed on a DAHDI channel.

Syntax

```
Event: DNDState
Status: <value>
Channel: <value>
```

Arguments

- `Status`
    - `enabled`
    - `disabled`
- `Channel`

Synopsis

Raised when the Do Not Disturb state is changed on an Analog channel.

Syntax

```
Event: DNDState
Status: <value>
Channel: <value>
```

Arguments

- `Status`
    - `enabled`
    - `disabled`
- `Channel`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_DTMF

DTMF

Synopsis

Raised when a DTMF digit has started or ended on a channel.

Syntax

```
Event: DTMF
Direction: <value>
Begin: <value>
End: <value>
Channel: <value>
Uniqueid: <value>
Digit: <value>
```

Arguments

- `Direction`
    - `Received`
    - `Sent`
- `Begin`
    - `Yes`
    - `No`
- `End`
    - `Yes`
    - `No`
- `Channel`
- `Uniqueid`
- `Digit`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ExtensionStatus

ExtensionStatus

Synopsis

Raised when an extension state has changed.

Syntax

```
Event: ExtensionStatus
Exten: <value>
Context: <value>
Hint: <value>
Status: <value>
```

Arguments

- `Exten`
- `Context`
- `Hint`
- `Status`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_FullyBooted

FullyBooted

Synopsis

Raised when all Asterisk initialization procedures have finished.

Syntax

```
Event: FullyBooted
Status: <value>
```

Arguments

- `Status`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Hangup

Hangup

Synopsis

Raised when a channel is hung up.

Syntax

```
Event: Hangup
Cause: <value>
Cause-txt: <value>
Channel: <value>
Uniqueid: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
```

Arguments

- `Cause` - A numeric cause code for why the channel was hung up.
- `Cause-txt` - A description of why the channel was hung up.
- `Channel`
- `Uniqueid`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_HangupHandlerPop

HangupHandlerPop

Synopsis

Raised when a hangup handler is removed from the handler stack by the CHANNEL() function.

Syntax

```
Event: HangupHandlerPop
Handler: <value>
Channel: <value>
Uniqueid: <value>
```

Arguments

- `Handler` - Hangup handler parameter string passed to the Gosub application.
- `Channel`
- `Uniqueid`

See Also

- Asterisk 11 ManagerEvent_HangupHandlerPush
- Asterisk 11 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_HangupHandlerPush

HangupHandlerPush

Synopsis

Raised when a hangup handler is added to the handler stack by the CHANNEL() function.

Syntax

```
Event: HangupHandlerPush
Handler: <value>
Channel: <value>
Uniqueid: <value>
```

Arguments

- `Handler` - Hangup handler parameter string passed to the Gosub application.
- `Channel`
- `Uniqueid`

See Also

- Asterisk 11 ManagerEvent_HangupHandlerPop
- Asterisk 11 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_HangupHandlerRun

HangupHandlerRun

Synopsis

Raised when a hangup handler is about to be called.

Syntax

```
Event: HangupHandlerRun
Handler: <value>
Channel: <value>
Uniqueid: <value>
```

Arguments

- `Handler` - Hangup handler parameter string passed to the Gosub application.
- `Channel`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_HangupRequest

HangupRequest

Synopsis

Raised when a hangup is requested with no set cause.

Syntax

```
Event: HangupRequest
Channel: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `Uniqueid`

Synopsis

Raised when a hangup is requested with a specific cause code.

Syntax

```
Event: HangupRequest
Cause: <value>
Channel: <value>
Uniqueid: <value>
Cause: <value>
```

Arguments

- `Cause` - A numeric cause code for why the channel was hung up.
- `Channel`
- `Uniqueid`
- `Cause`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Hold

Hold

Synopsis

Raised when a PRI channel is put on Hold.

Syntax

```
Event: Hold
Status: <value>
Channel: <value>
Uniqueid: <value>
```

Arguments

- `Status`
    - `On`
    - `Off`
- `Channel`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Join

Join

Synopsis

Raised when a channel joins a Queue.

Syntax

```
Event: Join
Queue: <value>
Position: <value>
Count: <value>
Channel: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
Queue: <value>
Uniqueid: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Position` - This channel's current position in the queue.
- `Count` - The total number of channels in the queue.
- `Channel`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `Queue`
- `Uniqueid`

See Also

- Asterisk 11 ManagerEvent_Leave
- Asterisk 11 Application_Queue

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Leave

Leave

Synopsis

Raised when a channel leaves a Queue.

Syntax

```
Event: Leave
Queue: <value>
Count: <value>
Position: <value>
Channel: <value>
Queue: <value>
Count: <value>
Position: <value>
Uniqueid: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Count` - The total number of channels in the queue.
- `Position` - This channel's current position in the queue.
- `Channel`
- `Queue`
- `Count`
- `Position`
- `Uniqueid`

See Also

- Asterisk 11 ManagerEvent_Join

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_LocalBridge

LocalBridge

Synopsis

Raised when two halves of a Local Channel form a bridge.

Syntax

```
Event: LocalBridge
Channel1: <value>
Channel2: <value>
Context: <value>
Exten: <value>
LocalOptimization: <value>
Uniqueid1: <value>
Uniqueid2: <value>
```

Arguments

- `Channel1` - The name of the Local Channel half that bridges to another channel.
- `Channel2` - The name of the Local Channel half that executes the dialplan.
- `Context` - The context in the dialplan that Channel2 starts in.
- `Exten` - The extension in the dialplan that Channel2 starts in.
- `LocalOptimization`
  - `Yes`
  - `No`
- `Uniqueid1`
- `Uniqueid2`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_LogChannel

LogChannel

Synopsis

Raised when a logging channel is re-enabled after a reload operation.

Syntax

```
Event: LogChannel
Channel: <value>
Enabled: <value>
```

Arguments

- `Channel` - The name of the logging channel.
- `Enabled`

Synopsis

Raised when a logging channel is disabled.

Syntax

```
Event: LogChannel
Channel: <value>
Enabled: <value>
Reason: <value>
```

Arguments

- `Channel` - The name of the logging channel.
- `Enabled`
- `Reason`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Masquerade

Masquerade

Synopsis

Raised when a masquerade occurs between two channels, wherein the Clone channel's internal information replaces the Original channel's information.

Syntax

```
Event: Masquerade
Clone: <value>
CloneState: <value>
Original: <value>
OriginalState: <value>
```

Arguments

- `Clone` - The name of the channel whose information will be going into the Original channel.
- `CloneState` - The current state of the clone channel.
- `Original` - The name of the channel whose information will be replaced by the Clone channel's information.
- `OriginalState` - The current state of the original channel.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MeetmeEnd

MeetmeEnd

Synopsis

Raised when a MeetMe conference ends.

Syntax

```
Event: MeetmeEnd
Meetme: <value>
Meetme: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Meetme`

See Also

- Asterisk 11 ManagerEvent_MeetmeJoin

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MeetmeJoin

MeetmeJoin

Synopsis

Raised when a user joins a MeetMe conference.

Syntax

```
Event: MeetmeJoin
Meetme: <value>
Usernum: <value>
Channel: <value>
Uniqueid: <value>
CallerIDnum: <value>
CallerIDname: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Channel`
- `Uniqueid`
- `CallerIDnum`
- `CallerIDname`
- `ConnectedLineNum`
- `ConnectedLineName`

See Also

- Asterisk 11 ManagerEvent_MeetmeLeave
- Asterisk 11 Application_MeetMe

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MeetmeLeave

MeetmeLeave

Synopsis

Raised when a user leaves a MeetMe conference.

Syntax

```
Event: MeetmeLeave
Meetme: <value>
Usernum: <value>
Duration: <value>
Channel: <value>
Uniqueid: <value>
Meetme: <value>
Usernum: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Duration` - The length of time in seconds that the Meetme user was in the conference.
- `Channel`
- `Uniqueid`
- `Meetme`
- `Usernum`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`

See Also

- Asterisk 11 ManagerEvent_MeetmeJoin

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MeetmeMute

MeetmeMute

Synopsis

Raised when a MeetMe user is muted.

Syntax

```
Event: MeetmeMute
Meetme: <value>
Usernum: <value>
Status: <value>
Channel: <value>
Uniqueid: <value>
Meetme: <value>
Usernum: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Status`
    - `on`
    - `off`
- `Channel`
- `Uniqueid`
- `Meetme`
- `Usernum`

Synopsis

Raised when a MeetMe user is unmuted.

Syntax

```
Event: MeetmeMute
Channel: <value>
Uniqueid: <value>
Meetme: <value>
Usernum: <value>
Status: <value>
```

Arguments

- `Channel`
- `Uniqueid`
- `Meetme`
- `Usernum`
- `Status`
    - `on`
    - `off`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MeetmeTalking

MeetmeTalking

Synopsis

Raised when a MeetMe user begins or ends talking.

Syntax

```
Event: MeetmeTalking
Meetme: <value>
Usernum: <value>
Status: <value>
Channel: <value>
Uniqueid: <value>
Meetme: <value>
Usernum: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Status`
    - `on`
    - `off`
- `Channel`
- `Uniqueid`
- `Meetme`
- `Usernum`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MeetmeTalkRequest

MeetmeTalkRequest

Synopsis

Raised when a MeetMe user has started talking.

Syntax

```
Event: MeetmeTalkRequest
Meetme: <value>
Usernum: <value>
Status: <value>
Channel: <value>
Uniqueid: <value>
Meetme: <value>
Usernum: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Status`
    - `on`
    - `off`
- `Channel`
- `Uniqueid`
- `Meetme`
- `Usernum`

Synopsis

Raised when a MeetMe user has finished talking.

Syntax

```
Event: MeetmeTalkRequest
Channel: <value>
Uniqueid: <value>
Meetme: <value>
Usernum: <value>
Status: <value>
```

Arguments

- `Channel`
- `Uniqueid`
- `Meetme`
- `Usernum`
- `Status`
    - `on`
    - `off`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_MessageWaiting

MessageWaiting

Synopsis

Raised when a new message has been left in a voicemail mailbox.

Syntax

```
Event: MessageWaiting
Mailbox: <value>
Waiting: <value>
New: <value>
Old: <value>
```

Arguments

- `Mailbox` - The mailbox with the new message, specified as **mailbox@context**
- `Waiting` - Whether or not the mailbox has access to a voicemail application.
- `New` - The number of new messages.
- `Old` - The number of old messages.

Synopsis

Raised when a user has finished listening to their messages.

Syntax

```
Event: MessageWaiting
Mailbox: <value>
Waiting: <value>
```

Arguments

- `Mailbox` - The mailbox with the new message, specified as **mailbox@context**
- `Waiting` - Whether or not the mailbox has access to a voicemail application.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ModuleLoadReport

ModuleLoadReport

Synopsis

Raised when all dynamic modules have finished their initial loading.

Syntax

```
Event: ModuleLoadReport
ModuleSelection: <value>
ModuleLoadStatus: <value>
ModuleCount: <value>
```

Arguments

- `ModuleSelection`
    - `Preload`
    - `All`
- `ModuleLoadStatus`
- `ModuleCount`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_NewAccountCode

NewAccountCode

Synopsis

Raised when a CDR's AccountCode is changed.

Syntax

```
Event: NewAccountCode
Channel: <value>
Uniqueid: <value>
AccountCode: <value>
OldAccountCode: <value>
```

Arguments

- `Channel`
- `Uniqueid`
- `AccountCode`
- `OldAccountCode`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_NewCallerid

NewCallerid

Synopsis

Raised when a channel receives new Caller ID information.

Syntax

```
Event: NewCallerid
CID-CallingPres: <value>
Channel: <value>
CallerIDNum: <value>
CallerIDName: <value>
Uniqueid: <value>
```

Arguments

- `CID-CallingPres` - A description of the Caller ID presentation.
- `Channel`
- `CallerIDNum`
- `CallerIDName`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Newchannel

Newchannel

Synopsis

Raised when a new channel is created.

Syntax

```
Event: Newchannel
ChannelState: <value>
ChannelStateDesc: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
AccountCode: <value>
Exten: <value>
Context: <value>
Uniqueid: <value>
```

Arguments

- `ChannelState` - A numeric code for the channel's current state, related to ChannelStateDesc
- `ChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `Channel`
- `ChannelState`
- `ChannelStateDesc`
- `CallerIDNum`
- `CallerIDName`
- `AccountCode`
- `Exten`
- `Context`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Newexten

Newexten

Synopsis

Raised when a channel enters a new context, extension, priority.

Syntax

```
Event: Newexten
Application: <value>
AppData: <value>
Channel: <value>
Context: <value>
Extension: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Application` - The application about to be executed.
- `AppData` - The data to be passed to the application.
- `Channel`
- `Context`
- `Extension`
- `Priority`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_NewPeerAccount

NewPeerAccount

Synopsis

Raised when a CDR's PeerAccount is changed.

Syntax

```
Event: NewPeerAccount
Channel: <value>
Uniqueid: <value>
PeerAccount: <value>
OldPeerAccount: <value>
```

Arguments

- `Channel`
- `Uniqueid`
- `PeerAccount`
- `OldPeerAccount`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Newstate

Newstate

Synopsis

Raised when a channel's state changes.

Syntax

```
Event: Newstate
ChannelState: <value>
ChannelStateDesc: <value>
Channel: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
Uniqueid: <value>
```

Arguments

- `ChannelState` - A numeric code for the channel's current state, related to ChannelStateDesc
- `ChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `Channel`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_OriginateResponse

OriginateResponse

Synopsis

Raised in response to an Originate command.

Syntax

```
Event: OriginateResponse
[ActionID:] <value>
Resonse: <value>
Channel: <value>
Context: <value>
Exten: <value>
Reason: <value>
Uniqueid: <value>
CallerIDNum: <value>
CallerIDName: <value>
```

Arguments

- `ActionID`
- `Resonse`
    - `Failure`
    - `Success`
- `Channel`
- `Context`
- `Exten`
- `Reason`
- `Uniqueid`
- `CallerIDNum`
- `CallerIDName`

See Also

- Asterisk 11 ManagerAction_Originate

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ParkedCall

ParkedCall

Synopsis

Raised when a call has been parked.

Syntax

```
Event: ParkedCall
Exten: <value>
Parkinglot: <value>
From: <value>
Channel: <value>
Timeout: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
Uniqueid: <value>
```

Arguments

- `Exten` - The parking lot extension.
- `Parkinglot` - The name of the parking lot.
- `From` - The name of the channel that parked the call.
- `Channel`
- `Timeout`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `Uniqueid`

See Also

- Asterisk 11 Application_Park
- Asterisk 11 ManagerAction_Park
- Asterisk 11 ManagerEvent_ParkedCallTimeOut
- Asterisk 11 ManagerEvent_ParkedCallGiveUp

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ParkedCallGiveUp

ParkedCallGiveUp

Synopsis

Raised when a parked call hangs up while in the parking lot.

Syntax

```
Event: ParkedCallGiveUp
Exten: <value>
Channel: <value>
Parkinglot: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
UniqueID: <value>
```

Arguments

- `Exten` - The parking lot extension.
- `Channel`
- `Parkinglot` - The name of the parking lot.
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `UniqueID`

See Also

- Asterisk 11 ManagerEvent_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_ParkedCallTimeOut

ParkedCallTimeOut

Synopsis

Raised when a parked call times out.

Syntax

```
Event: ParkedCallTimeOut
Exten: <value>
Channel: <value>
Parkinglot: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
UniqueID: <value>
```

Arguments

- `Exten` - The parking lot extension.
- `Channel`
- `Parkinglot` - The name of the parking lot.
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `UniqueID`

See Also

- Asterisk 11 ManagerEvent_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Pickup

Pickup

Synopsis

Raised when a call pickup occurs.

Syntax

```
Event: Pickup
Channel: <value>
TargetChannel: <value>
```

Arguments

- `Channel` - The name of the channel that initiated the pickup.
- `TargetChannel` - The name of the channel that is being picked up.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueCallerAbandon

QueueCallerAbandon

Synopsis

Raised when an caller abandons the queue.

Syntax

```
Event: QueueCallerAbandon
Queue: <value>
Position: <value>
OriginalPosition: <value>
HoldTime: <value>
Queue: <value>
Uniqueid: <value>
Position: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Position` - This channel's current position in the queue.
- `OriginalPosition` - The channel's original position in the queue.
- `HoldTime` - The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Queue`
- `Uniqueid`
- `Position`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueMemberAdded

QueueMemberAdded

Synopsis

Raised when a member is added to the queue.

Syntax

```
Event: QueueMemberAdded
Queue: <value>
Location: <value>
MemberName: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Queue: <value>
Location: <value>
MemberName: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `StateInterface` - Channel technology or location from which to read device state changes.
- `Membership`
    - `dynamic`
    - `realtime`
    - `static`
- `Penalty` - The penalty associated with the queue member.
- `CallsTaken` - The number of calls this queue member has serviced.
- `LastCall` - The time this member last took call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Status` - The numeric device state status of the queue member.
    - `0` - AST_DEVICE_UNKNOWN
    - `1` - AST_DEVICE_NOT_INUSE
    - `2` - AST_DEVICE_INUSE
    - `3` - AST_DEVICE_BUSY
    - `4` - AST_DEVICE_INVALID
    - `5` - AST_DEVICE_UNAVAILABLE
    - `6` - AST_DEVICE_RINGING
    - `7` - AST_DEVICE_RINGINUSE
    - `8` - AST_DEVICE_ONHOLD
- `Paused`
    - `0`
    - `1`
- `Queue`
- `Location`
- `MemberName`
- `StateInterface`
- `Membership`
- `Penalty`
- `CallsTaken`

* `LastCall`
* `Status`
* `Paused`

See Also

* Asterisk 11 ManagerEvent_QueueMemberRemoved
* Asterisk 11 Application_AddQueueMember

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueMemberPaused

QueueMemberPaused

Synopsis

Raised when a member is paused/unpaused in the queue with a reason.

Syntax

```
Event: QueueMemberPaused
Queue: <value>
Location: <value>
MemberName: <value>
Paused: <value>
Queue: <value>
Location: <value>
MemberName: <value>
Paused: <value>
Reason: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `Paused`
    - `0`
    - `1`
- `Queue`
- `Location`
- `MemberName`
- `Paused`
- `Reason`

See Also

- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnPauseQueueMember

Synopsis

Raised when a member is paused/unpaused in the queue without a reason.

Syntax

```
Event: QueueMemberPaused
Queue: <value>
Location: <value>
MemberName: <value>
Paused: <value>
Queue: <value>
Location: <value>
MemberName: <value>
Paused: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `Paused`
    - `0`
    - `1`
- `Queue`
- `Location`

* `MemberName`
* `Paused`

See Also

* Asterisk 11 Application_PauseQueueMember
* Asterisk 11 Application_UnPauseQueueMember

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueMemberPenalty

QueueMemberPenalty

Synopsis

Raised when a member's penalty is changed.

Syntax

```
Event: QueueMemberPenalty
Queue: <value>
Location: <value>
Penalty: <value>
Queue: <value>
Location: <value>
Penalty: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `Penalty` - The penalty associated with the queue member.
- `Queue`
- `Location`
- `Penalty`

See Also

- Asterisk 11 Function_QUEUE_MEMBER

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueMemberRemoved

QueueMemberRemoved

Synopsis

Raised when a member is removed from the queue.

Syntax

```
Event: QueueMemberRemoved
Queue: <value>
Location: <value>
MemberName: <value>
Queue: <value>
Location: <value>
MemberName: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `Queue`
- `Location`
- `MemberName`

See Also

- Asterisk 11 ManagerEvent_QueueMemberAdded
- Asterisk 11 Application_RemoveQueueMember

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueMemberRinginuse

QueueMemberRinginuse

Synopsis

Raised when a member's ringinuse setting is changed.

Syntax

```
Event: QueueMemberRinginuse
Queue: <value>
Location: <value>
Ringinuse: <value>
Queue: <value>
Location: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `Ringinuse`
  - `0`
  - `1`
- `Queue`
- `Location`

See Also

- Asterisk 11 Function_QUEUE_MEMBER

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_QueueMemberStatus

QueueMemberStatus

Synopsis

Raised when a Queue member's status has changed.

Syntax

```
Event: QueueMemberStatus
Queue: <value>
Location: <value>
MemberName: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
```

Arguments

- `Queue` - The name of the queue.
- `Location` - The queue member's channel technology or location.
- `MemberName` - The name of the queue member.
- `StateInterface` - Channel technology or location from which to read device state changes.
- `Membership`
    - `dynamic`
    - `realtime`
    - `static`
- `Penalty` - The penalty associated with the queue member.
- `CallsTaken` - The number of calls this queue member has serviced.
- `LastCall` - The time this member last took call, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- `Status` - The numeric device state status of the queue member.
    - `0` - AST_DEVICE_UNKNOWN
    - `1` - AST_DEVICE_NOT_INUSE
    - `2` - AST_DEVICE_INUSE
    - `3` - AST_DEVICE_BUSY
    - `4` - AST_DEVICE_INVALID
    - `5` - AST_DEVICE_UNAVAILABLE
    - `6` - AST_DEVICE_RINGING
    - `7` - AST_DEVICE_RINGINUSE
    - `8` - AST_DEVICE_ONHOLD
- `Paused`
    - `0`
    - `1`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Rename

Rename

Synopsis

Raised when the name of a channel is changed.

Syntax

```
Event: Rename
Channel: <value>
Newname: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `Newname`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_Shutdown

Shutdown

Synopsis

Raised when Asterisk is shutdown or restarted.

Syntax

```
Event: Shutdown
Shutdown: <value>
Restart: <value>
```

Arguments

- `Shutdown`
    - `Uncleanly`
    - `Cleanly`
- `Restart`
    - `True`
    - `False`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_SoftHangupRequest

SoftHangupRequest

Synopsis

Raised when a soft hangup is requested with a specific cause code.

Syntax

```
Event: SoftHangupRequest
Cause: <value>
Channel: <value>
Uniqueid: <value>
Cause: <value>
```

Arguments

- `Cause` - A numeric cause code for why the channel was hung up.
- `Channel`
- `Uniqueid`
- `Cause`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_SpanAlarm

SpanAlarm

Synopsis

Raised when an alarm is set on a DAHDI span.

Syntax

```
Event: SpanAlarm
Alarm: <value>
Span: <value>
```

Arguments

- `Alarm`
- `Span`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_SpanAlarmClear

SpanAlarmClear

Synopsis

Raised when an alarm is cleared on a DAHDI span.

Syntax

```
Event: SpanAlarmClear
Span: <value>
```

Arguments

- `Span`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_UnParkedCall

UnParkedCall

Synopsis

Raised when a call has been unparked.

Syntax

```
Event: UnParkedCall
Exten: <value>
Parkinglot: <value>
From: <value>
Exten: <value>
Channel: <value>
Parkinglot: <value>
From: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
Uniqueid: <value>
```

Arguments

- `Exten` - The parking lot extension.
- `Parkinglot` - The name of the parking lot.
- `From` - The name of the channel that parked the call.
- `Exten`
- `Channel`
- `Parkinglot`
- `From`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `Uniqueid`

See Also

- Asterisk 11 Application_ParkedCall
- Asterisk 11 ManagerEvent_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_UserEvent

UserEvent

Synopsis

A user defined event raised from the dialplan.

Syntax

```
Event: UserEvent
UserEvent: <value>
Uniqueid: <value>
```

Arguments

- `UserEvent` - The event name, as specified in the dialplan.
- `Uniqueid`

See Also

- Asterisk 11 Application_UserEvent

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 ManagerEvent_VarSet

VarSet

Synopsis

Raised when a LOCAL channel variable is set due to a subroutine call.

Syntax

```
Event: VarSet
Channel: <value>
Variable: <value>
Value: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `Variable`
- `Value`
- `Uniqueid`

See Also

- Asterisk 11 Application_GoSub

Synopsis

Raised when a variable is set to a particular value.

Syntax

```
Event: VarSet
Channel: <value>
Variable: <value>
Value: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `Variable`
- `Value`
- `Uniqueid`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Dialplan Applications

Asterisk 11 Application_AddQueueMember

AddQueueMember()

Synopsis

Dynamically adds queue members.

Description

Dynamically adds interface to an existing queue. If the interface is already in the queue it will return an error.

This application sets the following channel variable upon completion:

- `AQMSTATUS` - The status of the attempt to add a queue member as a text string.
  - ADDED
  - MEMBERALREADY
  - NOSUCHQUEUE

Syntax

```
AddQueueMember(queuename,interface,penalty,options,membername,stateinterface)
```

Arguments

- `queuename`
- `interface`
- `penalty`
- `options`
- `membername`
- `stateinterface`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_ADSIProg

ADSIProg()

Synopsis

Load Asterisk ADSI Scripts into phone

Description

This application programs an ADSI Phone with the given script

Syntax

```
ADSIProg([script])
```

Arguments

- `script` - adsi script to use. If not given uses the default script `asterisk.adsi`

See Also

- Asterisk 11 Application_GetCPEID
- `adsi.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_AELSub

AELSub()

Synopsis

Launch subroutine built with AEL

Description

Execute the named subroutine, defined in AEL, from another dialplan language, such as extensions.conf, Realtime extensions, or Lua.

The purpose of this application is to provide a sane entry point into AEL subroutines, the implementation of which may change from time to time.

Syntax

```
AELSub(routine[,args])
```

Arguments

- `routine` - Named subroutine to execute.
- `args`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_AgentLogin

AgentLogin()

Synopsis

Call agent login.

Description

Asks the agent to login to the system. Always returns `-1`. While logged in, the agent can receive calls and will hear a `beep` when a new call comes in. The agent can dump the call by pressing the star key.

Syntax

```
AgentLogin(AgentNo,options)
```

Arguments

- `AgentNo`
- `options`
    - s - silent login - do not announce the login ok segment after agent logged on/off

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_AGENT
- `agents.conf`
- `queues.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_AgentMonitorOutgoing

AgentMonitorOutgoing()

Synopsis

Record agent's outgoing call.

Description

Tries to figure out the id of the agent who is placing outgoing call based on comparison of the callerid of the current interface and the global variable placed by the AgentCallbackLogin application. That's why it should be used only with the AgentCallbackLogin app. Uses the monitoring functions in chan_agent instead of Monitor application. That has to be configured in the `agents.conf` file.

Normally the app returns `0` unless the options are passed.

Syntax

```
AgentMonitorOutgoing(options)
```

Arguments

- `options`
    - d - make the app return `-1` if there is an error condition.
    - c - change the CDR so that the source of the call is `Agent/agent_id`
    - n - don't generate the warnings when there is no callerid or the agentid is not known. It's handy if you want to have one context for agent and non-agent calls.

See Also

- `agents.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_AGI

AGI()

Synopsis

Executes an AGI compliant application.

Description

Executes an Asterisk Gateway Interface compliant program on a channel. AGI allows Asterisk to launch external programs written in any language to control a telephony channel, play audio, read DTMF digits, etc. by communicating with the AGI protocol on **stdin** and **stdout**. As of `1.6.0`, this channel will not stop dialplan execution on hangup inside of this application. Dialplan execution will continue normally, even upon hangup until the AGI application signals a desire to stop (either by exiting or, in the case of a net script, by closing the connection). A locally executed AGI script will receive SIGHUP on hangup from the channel except when using DeadAGI. A fast AGI server will correspondingly receive a HANGUP inline with the command dialog. Both of theses signals may be disabled by setting the `AGISIGHUP` channel variable to `no` before executing the AGI application. Alternatively, if you would like the AGI application to exit immediately after a channel hangup is detected, set the `AGIEXITONHANGUP` variable to `yes`.

Use the CLI command `agi show commands` to list available agi commands.

This application sets the following channel variable upon completion:

- `AGISTATUS` - The status of the attempt to the run the AGI script text string, one of:
    - SUCCESS
    - FAILURE
    - NOTFOUND
    - HANGUP

Syntax

```
AGI(commandarg1arg2[...])
```

Arguments

- `command`
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 11 Application_EAGI
- Asterisk 11 Application_DeadAGI

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_AlarmReceiver

AlarmReceiver()

Synopsis

Provide support for receiving alarm reports from a burglar or fire alarm panel.

Description

This application should be called whenever there is an alarm panel calling in to dump its events. The application will handshake with the alarm panel, and receive events, validate them, handshake them, and store them until the panel hangs up. Once the panel hangs up, the application will run the system command specified by the eventcmd setting in `alarmreceiver.conf` and pipe the events to the standard input of the application. The configuration file also contains settings for DTMF timing, and for the loudness of the acknowledgement tones.

> **Note**
> Only 1 signalling format is supported at this time: Ademco Contact ID.

Syntax

```
AlarmReceiver()
```

Arguments

See Also

- `alarmreceiver.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_AMD

AMD()

Synopsis

Attempt to detect answering machines.

Description

This application attempts to detect answering machines at the beginning of outbound calls. Simply call this application after the call has been answered (outbound only, of course).

When loaded, AMD reads amd.conf and uses the parameters specified as default values. Those default values get overwritten when the calling AMD with parameters.

This application sets the following channel variables:

- `AMDSTATUS` - This is the status of the answering machine detection
  - MACHINE
  - HUMAN
  - NOTSURE
  - HANGUP
- `AMDCAUSE` - Indicates the cause that led to the conclusion
  - TOOLONG - Total Time.
  - INITIALSILENCE - Silence Duration - Initial Silence.
  - HUMAN - Silence Duration - afterGreetingSilence.
  - LONGGREETING - Voice Duration - Greeting.
  - MAXWORDLENGTH - Word Count - maximum number of words.

Syntax

```
AMD([initialSilence[,greeting[,afterGreetingSilence[,totalAnalysis
Time[,miniumWordLength[,betweenWordSilence[,maximumNumberOfWords[,silenceThreshold[,maximumWordLength]]]]]]]]])
```

Arguments

- `initialSilence` - Is maximum initial silence duration before greeting.
  If this is exceeded set as MACHINE
- `greeting` - is the maximum length of a greeting.
  If this is exceeded set as MACHINE
- `afterGreetingSilence` - Is the silence after detecting a greeting.
  If this is exceeded set as HUMAN
- `totalAnalysis Time` - Is the maximum time allowed for the algorithm
  to decide HUMAN or MACHINE
- `miniumWordLength` - Is the minimum duration of Voice considered to be a word
- `betweenWordSilence` - Is the minimum duration of silence after a word to consider the audio that follows to be a new word
- `maximumNumberOfWords` - Is the maximum number of words in a greeting
  If this is exceeded set as MACHINE
- `silenceThreshold` - How long do we consider silence
- `maximumWordLength` - Is the maximum duration of a word to accept.
  If exceeded set as MACHINE

See Also

- Asterisk 11 Application_WaitForSilence
- Asterisk 11 Application_WaitForNoise

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Answer

Answer()

Synopsis

Answer a channel if ringing.

Description

If the call has not been answered, this application will answer it. Otherwise, it has no effect on the call.

Syntax

```
Answer(delay,nocdr)
```

Arguments

- `delay` - Asterisk will wait this number of milliseconds before returning to the dialplan after answering the call.
- `nocdr` - Asterisk will send an answer signal to the calling phone, but will not set the disposition or answer time in the CDR for this call.

See Also

- Asterisk 11 Application_Hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Authenticate

Authenticate()

Synopsis

Authenticate a user

Description

This application asks the caller to enter a given password in order to continue dialplan execution.

If the password begins with the `/` character, it is interpreted as a file which contains a list of valid passwords, listed 1 password per line in the file.

When using a database key, the value associated with the key can be anything.

Users have three attempts to authenticate before the channel is hung up.

Syntax

```
Authenticate(password[,options[,maxdigits[,prompt]]])
```

Arguments

- `password` - Password the user should know
- `options`
    - `a` - Set the channels' account code to the password that is entered
    - `d` - Interpret the given path as database key, not a literal file.
    - `m` - Interpret the given path as a file which contains a list of account codes and password hashes delimited with `:`, listed one per line in the file. When one of the passwords is matched, the channel will have its account code set to the corresponding account code in the file.
    - `r` - Remove the database key upon successful entry (valid with `d` only)
- `maxdigits` - maximum acceptable number of digits. Stops reading after maxdigits have been entered (without requiring the user to press the `#` key). Defaults to `0` - no limit - wait for the user press the `#` key.
- `prompt` - Override the agent-pass prompt file.

See Also

- Asterisk 11 Application_VMAuthenticate
- Asterisk 11 Application_DISA

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_BackGround

BackGround()

Synopsis

Play an audio file while waiting for digits of an extension to go to.

Description

This application will play the given list of files **(do not put extension)** while waiting for an extension to be dialed by the calling channel. To continue waiting for digits after this application has finished playing files, the `WaitExten` application should be used.

If one of the requested sound files does not exist, call processing will be terminated.

This application sets the following channel variable upon completion:

- `BACKGROUNDSTATUS` - The status of the background attempt as a text string.
    - SUCCESS
    - FAILED

Syntax

```
BackGround(filename1&filename2[&...],options,langoverride,context)
```

Arguments

- `filenames`
    - `filename1`
    - `filename2`
- `options`
    - `s` - Causes the playback of the message to be skipped if the channel is not in the `up` state (i.e. it hasn't been answered yet). If this happens, the application will return immediately.
    - `n` - Don't answer the channel before playing the files.
    - `m` - Only break if a digit hit matches a one digit extension in the destination context.
- `langoverride` - Explicitly specifies which language to attempt to use for the requested sound files.
- `context` - This is the dialplan context that this application will use when exiting to a dialed extension.

See Also

- Asterisk 11 Application_ControlPlayback
- Asterisk 11 Application_WaitExten
- Asterisk 11 Application_BackgroundDetect
- Asterisk 11 Function_TIMEOUT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_BackgroundDetect

BackgroundDetect()

Synopsis

Background a file with talk detect.

Description

Plays back *filename*, waiting for interruption from a given digit (the digit must start the beginning of a valid extension, or it will be ignored). During the playback of the file, audio is monitored in the receive direction, and if a period of non-silence which is greater than *min* ms yet less than *max* ms is followed by silence for at least *sil* ms, which occurs during the first *analysistime* ms, then the audio playback is aborted and processing jumps to the *talk* extension, if available.

Syntax

```
BackgroundDetect(filename,sil,min,max,analysistime)
```

Arguments

- `filename`
- `sil` - If not specified, defaults to `1000`.
- `min` - If not specified, defaults to `100`.
- `max` - If not specified, defaults to `infinity`.
- `analysistime` - If not specified, defaults to `infinity`.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Bridge

Bridge()

Synopsis

Bridge two channels.

Description

Allows the ability to bridge two channels via the dialplan.

This application sets the following channel variable upon completion:

- `BRIDGERESULT` - The result of the bridge attempt as a text string.
    - SUCCESS
    - FAILURE
    - LOOP
    - NONEXISTENT
    - INCOMPATIBLE

Syntax

```
Bridge(channel,options)
```

Arguments

- `channel` - The current channel is bridged to the specified *channel*.
- `options`
    - `p` - Play a courtesy tone to *channel*.
    - `F` - When the bridger hangs up, transfer the **bridged** party to the specified destination and **start** execution at that location.
        - `context`
        - `exten`
        - `priority`
    - `F` - When the bridger hangs up, transfer the **bridged** party to the next priority ofthe current extension and **start** execution at that location.
    - `h` - Allow the called party to hang up by sending the `*` DTMF digit.
    - `H` - Allow the calling party to hang up by pressing the `*` DTMF digit.
    - `k` - Allow the called party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
    - `K` - Allow the calling party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
    - `L(xyz)` - Limit the call to *x* ms. Play a warning when *y* ms are left. Repeat the warning every *z* ms. The following special variables can be used with this option:
        - `LIMIT_PLAYAUDIO_CALLER` - Play sounds to the caller. yes|no (default yes)
        - `LIMIT_PLAYAUDIO_CALLEE` - Play sounds to the callee. yes|no
        - `LIMIT_TIMEOUT_FILE` - File to play when time is up.
        - `LIMIT_CONNECT_FILE` - File to play when call begins.
        - `LIMIT_WARNING_FILE` - File to play as warning if *y* is defined. The default is to say the time remaining.
    - `S` - Hang up the call after *x* seconds **after** the called party has answered the call.
    - `t` - Allow the called party to transfer the calling party by sending the DTMF sequence defined in `features.conf`.
    - `T` - Allow the calling party to transfer the called party by sending the DTMF sequence defined in `features.conf`.
    - `w` - Allow the called party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
    - `W` - Allow the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
    - `x` - Cause the called party to be hung up after the bridge, instead of being restarted in the dialplan.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Busy

Busy()

Synopsis

Indicate the Busy condition.

Description

This application will indicate the busy condition to the calling channel.

Syntax

```
Busy(timeout)
```

Arguments

- `timeout` - If specified, the calling channel will be hung up after the specified number of seconds. Otherwise, this application will wait until the calling channel hangs up.

See Also

- Asterisk 11 Application_Congestion
- Asterisk 11 Application_Progress
- Asterisk 11 Application_Playtones
- Asterisk 11 Application_Hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_CallCompletionCancel

CallCompletionCancel()

Synopsis

Cancel call completion service

Description

Cancel a Call Completion Request.

This application sets the following channel variables:

- `CC_CANCEL_RESULT` - This is the returned status of the cancel.
  - SUCCESS
  - FAIL
- `CC_CANCEL_REASON` - This is the reason the cancel failed.
  - NO_CORE_INSTANCE
  - NOT_GENERIC
  - UNSPECIFIED

Syntax

```
CallCompletionCancel()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_CallCompletionRequest

CallCompletionRequest()

Synopsis

Request call completion service for previous call

Description

Request call completion service for a previously failed call attempt.

This application sets the following channel variables:

- `CC_REQUEST_RESULT` - This is the returned status of the request.
    - SUCCESS
    - FAIL
- `CC_REQUEST_REASON` - This is the reason the request failed.
    - NO_CORE_INSTANCE
    - NOT_GENERIC
    - TOO_MANY_REQUESTS
    - UNSPECIFIED

Syntax

```
CallCompletionRequest()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_CELGenUserEvent

CELGenUserEvent()

Synopsis

Generates a CEL User Defined Event.

Description

A CEL event will be immediately generated by this channel, with the supplied name for a type.

Syntax

```
CELGenUserEvent(event-name[extra])
```

Arguments

- `event-name`
  - `event-name`
  - `extra` - Extra text to be included with the event.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ChangeMonitor

ChangeMonitor()

Synopsis

Change monitoring filename of a channel.

Description

Changes monitoring filename of a channel. Has no effect if the channel is not monitored.

Syntax

```
ChangeMonitor(filename_base)
```

Arguments

- `filename_base` - The new filename base to use for monitoring this channel.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ChanIsAvail

ChanIsAvail()

Synopsis

Check channel availability

Description

This application will check to see if any of the specified channels are available.

This application sets the following channel variables:

- `AVAILCHAN` - The name of the available channel, if one exists
- `AVAILORIGCHAN` - The canonical channel name that was used to create the channel
- `AVAILSTATUS` - The device state for the device
- `AVAILCAUSECODE` - The cause code returned when requesting the channel

Syntax

```
ChanIsAvail(Technology2/Resource2[&...][,options])
```

Arguments

- `Technology/Resource` - ** `Technology2/Resource2` - Optional extra devices to check
  If you need more then one enter them as Technology2/Resource2&Technology3/Resourse3&.....
  Specification of the device(s) to check. These must be in the format of `Technology/Resource`, where *Technology* represents a particular channel driver, and *Resource* represents a resource available to that particular channel driver.
- `options`
  - `a` - Check for all available channels, not only the first one
  - `s` - Consider the channel unavailable if the channel is in use at all
  - `t` - Simply checks if specified channels exist in the channel list

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ChannelRedirect

ChannelRedirect()

Synopsis

Redirects given channel to a dialplan target

Description

Sends the specified channel to the specified extension priority

This application sets the following channel variables upon completion

- `CHANNELREDIRECT_STATUS` - Are set to the result of the redirection
  - NOCHANNEL
  - SUCCESS

Syntax

```
ChannelRedirect(channel[,context[,extension,priority]])
```

Arguments

- `channel`
- `context`
- `extension`
- `priority`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ChanSpy

ChanSpy()

Synopsis

Listen to a channel, and optionally whisper into it.

Description

This application is used to listen to the audio from an Asterisk channel. This includes the audio coming in and out of the channel being spied on. If the `chanprefix` parameter is specified, only channels beginning with this string will be spied upon.

While spying, the following actions may be performed:

- Dialing # cycles the volume level.
- Dialing `*` will stop spying and look for another channel to spy on.
- Dialing a series of digits followed by # builds a channel name to append to 'chanprefix'. For example, executing ChanSpy(Agent) and then dialing the digits '1234#' while spying will begin spying on the channel 'Agent/1234'. Note that this feature will be overridden if the 'd' option is used

> **Note**
> The *X* option supersedes the three features above in that if a valid single digit extension exists in the correct context ChanSpy will exit to it. This also disables choosing a channel based on `chanprefix` and a digit sequence.

Syntax

```
ChanSpy(chanprefix,options)
```

Arguments

- `chanprefix`
- `options`
    - b - Only spy on channels involved in a bridged call.
    - B - Instead of whispering on a single channel barge in on both channels involved in the call.
    - c
        - `digit` - Specify a DTMF digit that can be used to spy on the next available channel.
    - d - Override the typical numeric DTMF functionality and instead use DTMF to switch between spy modes.
        - 4 - spy mode
        - 5 - whisper mode
        - 6 - barge mode
    - e - Enable **enforced** mode, so the spying channel can only monitor extensions whose name is in the *ext* : delimited list.
        - `ext`
    - E - Exit when the spied-on channel hangs up.
    - g
        - `grp` - Only spy on channels in which one or more of the groups listed in *grp* matches one or more groups from the `SPYGROUP` variable set on the channel to be spied upon.
    - n - Say the name of the person being spied on if that person has recorded his/her name. If a context is specified, then that voicemail context will be searched when retrieving the name, otherwise the `default` context be used when searching for the name (i.e. if SIP/1000 is the channel being spied on and no mailbox is specified, then `1000` will be used when searching for the name).
        - `mailbox`
        - `context`
    - o - Only listen to audio coming from this channel.
    - q - Don't play a beep when beginning to spy on a channel, or speak the selected channel name.
    - r - Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is `chanspy`.
        - `basename`

- `s` - Skip the playback of the channel type (i.e. SIP, IAX, etc) when speaking the selected channel name.
- `S` - Stop when no more channels are left to spy on.
- `v` - Adjust the initial volume in the range from `-4` to `4`. A negative value refers to a quieter setting.
    - `value`
- `w` - Enable `whisper` mode, so the spying channel can talk to the spied-on channel.
- `W` - Enable `private whisper` mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.
- `x`
    - `digit` - Specify a DTMF digit that can be used to exit the application.
- `X` - Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the `SPY_EXIT_CONTEXT` channel variable. The name of the last channel that was spied on will be stored in the `SPY_CHANNEL` variable.

See Also

- Asterisk 11 Application_ExtenSpy

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_ClearHash

ClearHash()

Synopsis

Clear the keys from a specified hashname.

Description

Clears all keys out of the specified *hashname*.

Syntax

```
ClearHash(hashname)
```

Arguments

- `hashname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ConfBridge

ConfBridge()

Synopsis

Conference bridge application.

Description

Enters the user into a specified conference bridge. The user can exit the conference by hangup or DTMF menu option.

Syntax

```
ConfBridge(conference,bridge_profile,user_profile,menu)
```

Arguments

- `conference` - Name of the conference bridge. You are not limited to just numbers.
- `bridge_profile` - The bridge profile name from confbridge.conf. When left blank, a dynamically built bridge profile created by the CONFBRIDGE dialplan function is searched for on the channel and used. If no dynamic profile is present, the 'default_bridge' profile found in confbridge.conf is used.
  It is important to note that while user profiles may be unique for each participant, mixing bridge profiles on a single conference is _NOT_ recommended and will produce undefined results.
- `user_profile` - The user profile name from confbridge.conf. When left blank, a dynamically built user profile created by the CONFBRIDGE dialplan function is searched for on the channel and used. If no dynamic profile is present, the 'default_user' profile found in confbridge.conf is used.
- `menu` - The name of the DTMF menu in confbridge.conf to be applied to this channel. No menu is applied by default if this option is left blank.

See Also

- Asterisk 11 Application_ConfBridge
- Asterisk 11 Function_CONFBRIDGE
- Asterisk 11 Function_CONFBRIDGE_INFO

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Congestion

Congestion()

Synopsis

Indicate the Congestion condition.

Description

This application will indicate the congestion condition to the calling channel.

Syntax

```
Congestion(timeout)
```

Arguments

- `timeout` - If specified, the calling channel will be hung up after the specified number of seconds. Otherwise, this application will wait until the calling channel hangs up.

See Also

- Asterisk 11 Application_Busy
- Asterisk 11 Application_Progress
- Asterisk 11 Application_Playtones
- Asterisk 11 Application_Hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ContinueWhile

ContinueWhile()

Synopsis

Restart a While loop.

Description

Returns to the top of the while loop and re-evaluates the conditional.

Syntax

```
ContinueWhile()
```

Arguments

See Also

- Asterisk 11 Application_While
- Asterisk 11 Application_EndWhile
- Asterisk 11 Application_ExitWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ControlPlayback

ControlPlayback()

Synopsis

Play a file with fast forward and rewind.

Description

This application will play back the given *filename*.

It sets the following channel variables upon completion:

- `CPLAYBACKSTATUS` - Contains the status of the attempt as a text string
    - SUCCESS
    - USERSTOPPED
    - ERROR
- `CPLAYBACKOFFSET` - Contains the offset in ms into the file where playback was at when it stopped. `-1` is end of file.
- `CPLAYBACKSTOPKEY` - If the playback is stopped by the user this variable contains the key that was pressed.

Syntax

```
ControlPlayback(filename,skipms,ff,rew,stop,pause,restart,options)
```

Arguments

- `filename`
- `skipms` - This is number of milliseconds to skip when rewinding or fast-forwarding.
- `ff` - Fast-forward when this DTMF digit is received. (defaults to `#`)
- `rew` - Rewind when this DTMF digit is received. (defaults to `*`)
- `stop` - Stop playback when this DTMF digit is received.
- `pause` - Pause playback when this DTMF digit is received.
- `restart` - Restart playback when this DTMF digit is received.
- `options`
    - `o`
        - `time` - Start at *time* ms from the beginning of the file.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DAHDIAcceptR2Call

DAHDIAcceptR2Call()

Synopsis

Accept an R2 call if its not already accepted (you still need to answer it)

Description

This application will Accept the R2 call either with charge or no charge.

Syntax

```
DAHDIAcceptR2Call(charge)
```

Arguments

- `charge` - Yes or No.
  Whether you want to accept the call with charge or without charge.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DAHDIBarge

DAHDIBarge()

Synopsis

Barge in (monitor) DAHDI channel.

Description

Barges in on a specified DAHDI *channel* or prompts if one is not specified. Returns `-1` when caller user hangs up and is independent of the state of the channel being monitored.

Syntax

```
DAHDIBarge(channel)
```

Arguments

- `channel` - Channel to barge.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DAHDIRAS

DAHDIRAS()

Synopsis

Executes DAHDI ISDN RAS application.

Description

Executes a RAS server using pppd on the given channel. The channel must be a clear channel (i.e. PRI source) and a DAHDI channel to be able to use this function (No modem emulation is included).

Your pppd must be patched to be DAHDI aware.

Syntax

```
DAHDIRAS(args)
```

Arguments

- `args` - A list of parameters to pass to the pppd daemon, separated by `,` characters.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DAHDIScan

DAHDIScan()

Synopsis

Scan DAHDI channels to monitor calls.

Description

Allows a call center manager to monitor DAHDI channels in a convenient way. Use # to select the next channel and use * to exit.

Syntax

```
DAHDIScan(group)
```

Arguments

- `group` - Limit scanning to a channel *group* by setting this option.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DAHDISendCallreroutingFacility

DAHDISendCallreroutingFacility()

Synopsis

Send an ISDN call rerouting/deflection facility message.

Description

This application will send an ISDN switch specific call rerouting/deflection facility message over the current channel. Supported switches depend upon the version of libpri in use.

Syntax

```
DAHDISendCallreroutingFacility(destination,original,reason)
```

Arguments

- `destination` - Destination number.
- `original` - Original called number.
- `reason` - Diversion reason, if not specified defaults to `unknown`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DAHDISendKeypadFacility

DAHDISendKeypadFacility()

Synopsis

Send digits out of band over a PRI.

Description

This application will send the given string of digits in a Keypad Facility IE over the current channel.

Syntax

```
DAHDISendKeypadFacility(digits)
```

Arguments

- `digits`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DateTime

DateTime()

Synopsis

Says a specified time in a custom format.

Description

Say the date and time in a specified format.

Syntax

```
DateTime(unixtime,timezone,format)
```

Arguments

- `unixtime` - time, in seconds since Jan 1, 1970. May be negative. Defaults to now.
- `timezone` - timezone, see `/usr/share/zoneinfo` for a list. Defaults to machine default.
- `format` - a format the time is to be said in. See `voicemail.conf`. Defaults to `ABdY "digits/at" IMp`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_DBdel

DBdel()

Synopsis

Delete a key from the asterisk database.

Description

This application will delete a *key* from the Asterisk database.

> **Note**
> This application has been DEPRECATED in favor of the DB_DELETE function.

Syntax

```
DBdel(family/key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 11 Function_DB_DELETE
- Asterisk 11 Application_DBdeltree
- Asterisk 11 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DBdeltree

DBdeltree()

Synopsis

Delete a family or keytree from the asterisk database.

Description

This application will delete a *family* or *keytree* from the Asterisk database.

Syntax

```
DBdeltree(family/keytree)
```

Arguments

- `family`
- `keytree`

See Also

- Asterisk 11 Function_DB_DELETE
- Asterisk 11 Application_DBdel
- Asterisk 11 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DeadAGI

DeadAGI()

Synopsis

Executes AGI on a hungup channel.

Description

Executes an Asterisk Gateway Interface compliant program on a channel. AGI allows Asterisk to launch external programs written in any language to control a telephony channel, play audio, read DTMF digits, etc. by communicating with the AGI protocol on **stdin** and **stdout**. As of `1.6.0`, this channel will not stop dialplan execution on hangup inside of this application. Dialplan execution will continue normally, even upon hangup until the AGI application signals a desire to stop (either by exiting or, in the case of a net script, by closing the connection). A locally executed AGI script will receive SIGHUP on hangup from the channel except when using DeadAGI. A fast AGI server will correspondingly receive a HANGUP inline with the command dialog. Both of theses signals may be disabled by setting the `AGISIGHUP` channel variable to `no` before executing the AGI application. Alternatively, if you would like the AGI application to exit immediately after a channel hangup is detected, set the `AGIEXITONHANGUP` variable to `yes`.

Use the CLI command `agi show commands` to list available agi commands.

This application sets the following channel variable upon completion:

- `AGISTATUS` - The status of the attempt to the run the AGI script text string, one of:
    - SUCCESS
    - FAILURE
    - NOTFOUND
    - HANGUP

Syntax

```
DeadAGI(commandarg1arg2[...])
```

Arguments

- `command`
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 11 Application_AGI
- Asterisk 11 Application_EAGI

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Dial

Dial()

Synopsis

Attempt to connect to another device or endpoint and bridge the call.

Description

This application will place calls to one or more specified channels. As soon as one of the requested channels answers, the originating channel will be answered, if it has not already been answered. These two channels will then be active in a bridged call. All other channels that were requested will then be hung up.

Unless there is a timeout specified, the Dial application will wait indefinitely until one of the called channels answers, the user hangs up, or if all of the called channels are busy or unavailable. Dialplan executing will continue if no requested channels can be called, or if the timeout expires. This application will report normal termination if the originating channel hangs up, or if the call is bridged and either of the parties in the bridge ends the call.

If the `OUTBOUND_GROUP` variable is set, all peer channels created by this application will be put into that group (as in Set(GROUP()=...). If the `OUTBOUND_GROUP_ONCE` variable is set, all peer channels created by this application will be put into that group (as in Set(GROUP()=...). Unlike `OUTBOUND_GROUP`, however, the variable will be unset after use.

This application sets the following channel variables:

- `DIALEDTIME` - This is the time from dialing a channel until when it is disconnected.
- `ANSWEREDTIME` - This is the amount of time for actual call.
- `DIALSTATUS` - This is the status of the call
    - CHANUNAVAIL
    - CONGESTION
    - NOANSWER
    - BUSY
    - ANSWER
    - CANCEL
    - DONTCALL - For the Privacy and Screening Modes. Will be set if the called party chooses to send the calling party to the 'Go Away' script.
    - TORTURE - For the Privacy and Screening Modes. Will be set if the called party chooses to send the calling party to the 'torture' script.
    - INVALIDARGS

Syntax

```
Dial(Technology/Resource[&Technology2/Resource2[&...]][,timeout[,options,URL]])
```

Arguments

- `Technology/Resource`
    - `Technology/Resource` - Specification of the device(s) to dial. These must be in the format of `Technology/Resource`, where *Technology* represents a particular channel driver, and *Resource* represents a resource available to that particular channel driver.
    - `Technology2/Resource2` - Optional extra devices to dial in parallel
      If you need more then one enter them as Technology2/Resource2&Technology3/Resourse3&.....
- `timeout` - Specifies the number of seconds we attempt to dial the specified devices
  If not specified, this defaults to 136 years.
- `options`
    - A - Play an announcement to the called party, where *x* is the prompt to be played
        - x - The file to play to the called party
    - a - Immediately answer the calling channel when the called channel answers in all cases. Normally, the calling channel is answered when the called channel answers, but when options such as A() and M() are used, the calling channel is not answered until all actions on the called channel (such as playing an announcement) are completed. This option can be used to answer the calling channel before doing anything on the called channel. You will rarely need to use this option, the default behavior is adequate in most cases.
    - b - Before initiating an outgoing call, Gosub to the specified location using the newly created channel. The Gosub will be

executed for each destination channel.

    - `context`
    - `exten`
    - `priority`
        - `arg1`
        - `argN`
- `B` - Before initiating the outgoing call(s), Gosub to the specified location using the current channel.
    - `context`
    - `exten`
    - `priority`
        - `arg1`
        - `argN`
- `C` - Reset the call detail record (CDR) for this call.
- `c` - If the Dial() application cancels this call, always set HANGUPCAUSE to 'answered elsewhere'
- `d` - Allow the calling user to dial a 1 digit extension while waiting for a call to be answered. Exit to that extension if it exists in the current context, or the context defined in the `EXITCONTEXT` variable, if it exists.
- `D` - Send the specified DTMF strings **after** the called party has answered, but before the call gets bridged. The *called* DTMF string is sent to the called party, and the *calling* DTMF string is sent to the calling party. Both arguments can be used alone. If *pro gress* is specified, its DTMF is sent immediately after receiving a PROGRESS message.
    - `called`
    - `calling`
    - `progress`
- `e` - Execute the `h` extension for peer after the call ends
- `f` - If *x* is not provided, force the CallerID sent on a call-forward or deflection to the dialplan extension of this Dial() using a dialplan `hint`. For example, some PSTNs do not allow CallerID to be set to anything other than the numbers assigned to you. If *x* is provided, force the CallerID sent to *x*.
    - `x`
- `F` - When the caller hangs up, transfer the **called** party to the specified destination and **start** execution at that location.
    - `context`
    - `exten`
    - `priority`
- `F` - When the caller hangs up, transfer the **called** party to the next priority of the current extension and **start** execution at that location.
- `g` - Proceed with dialplan execution at the next priority in the current extension if the destination channel hangs up.
- `G` - If the call is answered, transfer the calling party to the specified *priority* and the called party to the specified *priority* plus one.
    - `context`
    - `exten`
    - `priority`
- `h` - Allow the called party to hang up by sending the DTMF sequence defined for disconnect in `features.conf`.
- `H` - Allow the calling party to hang up by sending the DTMF sequence defined for disconnect in `features.conf`.
- `i` - Asterisk will ignore any forwarding requests it may receive on this dial attempt.
- `I` - Asterisk will ignore any connected line update requests or any redirecting party update requests it may receive on this dial attempt.
- `k` - Allow the called party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.con f`.
- `K` - Allow the calling party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.con f`.
- `L` - Limit the call to *x* milliseconds. Play a warning when *y* milliseconds are left. Repeat the warning every *z* milliseconds until time expires.
  This option is affected by the following variables:
    - `LIMIT_PLAYAUDIO_CALLER` - If set, this variable causes Asterisk to play the prompts to the caller.
        - YES default: (true)
        - NO
    - `LIMIT_PLAYAUDIO_CALLEE` - If set, this variable causes Asterisk to play the prompts to the callee.
        - YES
        - NO default: (true)
    - `LIMIT_TIMEOUT_FILE` - If specified, *filename* specifies the sound prompt to play when the timeout is reached. If not set, the time remaining will be announced.
        - FILENAME

        - `LIMIT_CONNECT_FILE` - If specified, *filename* specifies the sound prompt to play when the call begins. If not set, the time remaining will be announced.
            - FILENAME
        - `LIMIT_WARNING_FILE` - If specified, *filename* specifies the sound prompt to play as a warning when time *x* is reached. If not set, the time remaining will be announced.
            - FILENAME
        - `x` - Maximum call time, in milliseconds
        - `y` - Warning time, in milliseconds
        - `z` - Repeat time, in milliseconds
- `m` - Provide hold music to the calling party until a requested channel answers. A specific music on hold *class* (as defined in `musiconhold.conf`) can be specified.
    - `class`
- `M` - Execute the specified *macro* for the **called** channel before connecting to the calling channel. Arguments can be specified to the Macro using `^` as a delimiter. The macro can set the variable `MACRO_RESULT` to specify the following actions after the macro is finished executing:
    - `MACRO_RESULT` - If set, this action will be taken after the macro finished executing.
        - ABORT - Hangup both legs of the call
        - CONGESTION - Behave as if line congestion was encountered
        - BUSY - Behave as if a busy signal was encountered
        - CONTINUE - Hangup the called party and allow the calling party to continue dialplan execution at the next priority
        - GOTO:[[<CONTEXT>^]<EXTEN>^]<PRIORITY> - Transfer the call to the specified destination.
    - `macro` - Name of the macro that should be executed.
    - `arg` - Macro arguments
- `n` - This option is a modifier for the call screening/privacy mode. (See the `p` and `P` options.) It specifies that no introductions are to be saved in the `priv-callerintros` directory.
    - `delete` - With *delete* either not specified or set to `0`, the recorded introduction will not be deleted if the caller hangs up while the remote party has not yet answered.
      With *delete* set to `1`, the introduction will always be deleted.
- `N` - This option is a modifier for the call screening/privacy mode. It specifies that if Caller*ID is present, do not screen the call.
- `o` - If *x* is not provided, specify that the CallerID that was present on the **calling** channel be stored as the CallerID on the **called** channel. This was the behavior of Asterisk 1.0 and earlier. If *x* is provided, specify the CallerID stored on the **called** channel. Note that o(${CALLERID(all)}) is similar to option o without the parameter.
    - `x`
- `O` - Enables **operator services** mode. This option only works when bridging a DAHDI channel to another DAHDI channel only. if specified on non-DAHDI interfaces, it will be ignored. When the destination answers (presumably an operator services station), the originator no longer has control of their line. They may hang up, but the switch will not release their line until the destination party (the operator) hangs up.
    - `mode` - With *mode* either not specified or set to `1`, the originator hanging up will cause the phone to ring back immediately.
      With *mode* set to `2`, when the operator flashes the trunk, it will ring their phone back.
- `p` - This option enables screening mode. This is basically Privacy mode without memory.
- `P` - Enable privacy mode. Use *x* as the family/key in the AstDB database if it is provided. The current extension is used if a database family/key is not specified.
    - `x`
- `r` - Default: Indicate ringing to the calling party, even if the called party isn't actually ringing. Pass no audio to the calling party until the called channel has answered.
    - `tone` - Indicate progress to calling party. Send audio 'tone' from indications.conf
- `S` - Hang up the call *x* seconds **after** the called party has answered the call.
    - `x`
- `s` - Force the outgoing callerid tag parameter to be set to the string *x*.
  Works with the `f` option.
    - `x`
- `t` - Allow the called party to transfer the calling party by sending the DTMF sequence defined in `features.conf`. This setting does not perform policy enforcement on transfers initiated by other methods.
- `T` - Allow the calling party to transfer the called party by sending the DTMF sequence defined in `features.conf`. This setting does not perform policy enforcement on transfers initiated by other methods.
- `U` - Execute via Gosub the routine *x* for the **called** channel before connecting to the calling channel. Arguments can be specified to the Gosub using `^` as a delimiter. The Gosub routine can set the variable `GOSUB_RESULT` to specify the following actions after

the Gosub returns.

        - `GOSUB_RESULT`
            - ABORT - Hangup both legs of the call.
            - CONGESTION - Behave as if line congestion was encountered.
            - BUSY - Behave as if a busy signal was encountered.
            - CONTINUE - Hangup the called party and allow the calling party to continue dialplan execution at the next priority.
            - GOTO:[[<CONTEXT>^]<EXTEN>^]<PRIORITY> - Transfer the call to the specified destination.
        - `x` - Name of the subroutine to execute via Gosub
        - `arg` - Arguments for the Gosub routine
    - u - Works with the f option.
        - `x` - Force the outgoing callerid presentation indicator parameter to be set to one of the values passed in *x*: `allowed_not_screened` `allowed_passed_screen` `allowed_failed_screen` `allowed` `prohib_not_screened` `prohib_passed_screen` `prohib_failed_screen` `prohib` `unavailable`
    - `w` - Allow the called party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
    - `W` - Allow the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in `features.conf`.
    - `x` - Allow the called party to enable recording of the call by sending the DTMF sequence defined for one-touch automixmonitor in `features.conf`.
    - `X` - Allow the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch automixmonitor in `features.conf`.
    - `z` - On a call forward, cancel any dial timeout which has been set for this call.
- `URL` - The optional URL will be sent to the called party if the channel driver supports it.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Dictate

Dictate()

Synopsis

Virtual Dictation Machine.

Description

Start dictation machine using optional *base_dir* for files.

Syntax

```
Dictate(base_dir,filename)
```

Arguments

- `base_dir`
- `filename`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Directory

Directory()

Synopsis

Provide directory of voicemail extensions.

Description

This application will present the calling channel with a directory of extensions from which they can search by name. The list of names and corresponding extensions is retrieved from the voicemail configuration file, `voicemail.conf`.

This application will immediately exit if one of the following DTMF digits are received and the extension to jump to exists:

0 - Jump to the 'o' extension, if it exists.

- - Jump to the 'a' extension, if it exists.

Syntax

```
Directory(vm-context[,dial-context[,options]])
```

Arguments

- `vm-context` - This is the context within voicemail.conf to use for the Directory. If not specified and `searchcontexts=no` in `voicemail.conf`, then `default` will be assumed.
- `dial-context` - This is the dialplan context to use when looking for an extension that the user has selected, or when jumping to the `o` or `a` extension. If not specified, the current context will be used.
- `options`
    - `e` - In addition to the name, also read the extension number to the caller before presenting dialing options.
    - `f` - Allow the caller to enter the first name of a user in the directory instead of using the last name. If specified, the optional number argument will be used for the number of characters the user should enter.
        - `n`
    - `l` - Allow the caller to enter the last name of a user in the directory. This is the default. If specified, the optional number argument will be used for the number of characters the user should enter.
        - `n`
    - `b` - Allow the caller to enter either the first or the last name of a user in the directory. If specified, the optional number argument will be used for the number of characters the user should enter.
        - `n`
    - `m` - Instead of reading each name sequentially and asking for confirmation, create a menu of up to 8 names.
    - `n` - Read digits even if the channel is not answered.
    - `p` - Pause for n milliseconds after the digits are typed. This is helpful for people with cellphones, who are not holding the receiver to their ear while entering DTMF.
        - `n`

> **Note**
> Only one of the *f*, *l*, or *b* options may be specified. **If more than one is specified**, then Directory will act as if *b* was specified. The number of characters for the user to type defaults to 3.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DISA

DISA()

Synopsis

Direct Inward System Access.

Description

The DISA, Direct Inward System Access, application allows someone from outside the telephone switch (PBX) to obtain an **internal** system dialtone and to place calls from it as if they were placing a call from within the switch. DISA plays a dialtone. The user enters their numeric passcode, followed by the pound sign `#`. If the passcode is correct, the user is then given system dialtone within *context* on which a call may be placed. If the user enters an invalid extension and extension `i` exists in the specified *context*, it will be used.

Be aware that using this may compromise the security of your PBX.

The arguments to this application (in `extensions.conf`) allow either specification of a single global *passcode* (that everyone uses), or individual passcodes contained in a file (*filename*).

The file that contains the passcodes (if used) allows a complete specification of all of the same arguments available on the command line, with the sole exception of the options. The file may contain blank lines, or comments starting with `#` or `;`.

Syntax

```
DISA(passcode|filename,context,cidmailbox[@context],options)
```

Arguments

- `passcode|filename` - If you need to present a DISA dialtone without entering a password, simply set *passcode* to `no-password` You may specified a *filename* instead of a *passcode*, this filename must contain individual passcodes
- `context` - Specifies the dialplan context in which the user-entered extension will be matched. If no context is specified, the DISA application defaults to the `disa` context. Presumably a normal system will have a special context set up for DISA use with some or a lot of restrictions.
- `cid` - Specifies a new (different) callerid to be used for this call.
- `mailbox` - Will cause a stutter-dialtone (indication **dialrecall**) to be used, if the specified mailbox contains any new messages.
    - `mailbox`
    - `context`
- `options`
    - `n` - The DISA application will not answer initially.
    - `p` - The extension entered will be considered complete when a `#` is entered.

See Also

- Asterisk 11 Application_Authenticate
- Asterisk 11 Application_VMAuthenticate

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_DumpChan

DumpChan()

Synopsis

Dump Info About The Calling Channel.

Description

Displays information on channel and listing of all channel variables. If *level* is specified, output is only displayed when the verbose level is currently set to that number or greater.

Syntax

```
DumpChan(level)
```

Arguments

- `level` - Minimun verbose level

See Also

- Asterisk 11 Application_NoOp
- Asterisk 11 Application_Verbose

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_EAGI

EAGI()

Synopsis

Executes an EAGI compliant application.

Description

Using 'EAGI' provides enhanced AGI, with incoming audio available out of band on file descriptor 3.

Executes an Asterisk Gateway Interface compliant program on a channel. AGI allows Asterisk to launch external programs written in any language to control a telephony channel, play audio, read DTMF digits, etc. by communicating with the AGI protocol on **stdin** and **stdout**. As of `1.6.0`, this channel will not stop dialplan execution on hangup inside of this application. Dialplan execution will continue normally, even upon hangup until the AGI application signals a desire to stop (either by exiting or, in the case of a net script, by closing the connection). A locally executed AGI script will receive SIGHUP on hangup from the channel except when using DeadAGI. A fast AGI server will correspondingly receive a HANGUP inline with the command dialog. Both of theses signals may be disabled by setting the `AGISIGHUP` channel variable to `no` before executing the AGI application. Alternatively, if you would like the AGI application to exit immediately after a channel hangup is detected, set the `AGIEXITONHANGUP` variable to `yes`.

Use the CLI command `agi show commands` to list available agi commands.

This application sets the following channel variable upon completion:

- `AGISTATUS` - The status of the attempt to the run the AGI script text string, one of:
    - SUCCESS
    - FAILURE
    - NOTFOUND
    - HANGUP

Syntax

```
EAGI(commandarg1arg2[...])
```

Arguments

- `command`
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 11 Application_AGI
- Asterisk 11 Application_DeadAGI

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Echo

Echo()

Synopsis

Echo media, DTMF back to the calling party

Description

Echos back any media or DTMF frames read from the calling channel back to itself. This will not echo CONTROL, MODEM, or NULL frames. Note: If '#' detected application exits.

This application does not automatically answer and should be preceeded by an application such as Answer() or Progress().

Syntax

```
Echo()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_EndWhile

EndWhile()

Synopsis

End a while loop.

Description

Return to the previous called `While()`.

Syntax

```
EndWhile()
```

Arguments

See Also

- Asterisk 11 Application_While
- Asterisk 11 Application_ExitWhile
- Asterisk 11 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Exec

Exec()

Synopsis

Executes dialplan application.

Description

Allows an arbitrary application to be invoked even when not hard coded into the dialplan. If the underlying application terminates the dialplan, or if the application cannot be found, Exec will terminate the dialplan.

To invoke external applications, see the application System. If you would like to catch any error instead, see TryExec.

Syntax

```
Exec(arguments)
```

Arguments

- `appname` - Application name and arguments of the dialplan application to execute.
    - `arguments`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ExecIf

ExecIf()

Synopsis

Executes dialplan application, conditionally.

Description

If *expr* is true, execute and return the result of *appiftrue(args)*.

If *expr* is true, but *appiftrue* is not found, then the application will return a non-zero value.

Syntax

```
ExecIf(expressionappiftrue[:appiffalse])
```

Arguments

- `expression`
- `execapp`
    - `appiftrue`
        - `args`
    - `appiffalse`
        - `args`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ExecIfTime

ExecIfTime()

Synopsis

Conditional application execution based on the current time.

Description

This application will execute the specified dialplan application, with optional arguments, if the current time matches the given time specification.

Syntax

```
ExecIfTime(timesweekdaysmdaysmonths[timezone]appargs)
```

Arguments

- `day_condition`
    - `times`
    - `weekdays`
    - `mdays`
    - `months`
    - `timezone`
- `appname`
    - `appargs`

See Also

- Asterisk 11 Application_Exec
- Asterisk 11 Application_ExecIf
- Asterisk 11 Application_TryExec
- Asterisk 11 Application_GotoIfTime

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ExitWhile

ExitWhile()

Synopsis

End a While loop.

Description

Exits a `While()` loop, whether or not the conditional has been satisfied.

Syntax

```
ExitWhile()
```

Arguments

See Also

- Asterisk 11 Application_While
- Asterisk 11 Application_EndWhile
- Asterisk 11 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ExtenSpy

ExtenSpy()

Synopsis

Listen to a channel, and optionally whisper into it.

Description

This application is used to listen to the audio from an Asterisk channel. This includes the audio coming in and out of the channel being spied on. Only channels created by outgoing calls for the specified extension will be selected for spying. If the optional context is not supplied, the current channel's context will be used.

While spying, the following actions may be performed:

- Dialing # cycles the volume level.
- Dialing `*` will stop spying and look for another channel to spy on.

> **Note**
> The *X* option supersedes the three features above in that if a valid single digit extension exists in the correct context ChanSpy will exit to it. This also disables choosing a channel based on `chanprefix` and a digit sequence.

Syntax

```
ExtenSpy(exten@context,options)
```

Arguments

- `exten`
    - `exten` - Specify extension.
    - `context` - Optionally specify a context, defaults to `default`.
- `options`
    - `b` - Only spy on channels involved in a bridged call.
    - `B` - Instead of whispering on a single channel barge in on both channels involved in the call.
    - `c`
        - `digit` - Specify a DTMF digit that can be used to spy on the next available channel.
    - `d` - Override the typical numeric DTMF functionality and instead use DTMF to switch between spy modes.
        - `4` - spy mode
        - `5` - whisper mode
        - `6` - barge mode
    - `e` - Enable **enforced** mode, so the spying channel can only monitor extensions whose name is in the *ext* : delimited list.
        - `ext`
    - `E` - Exit when the spied-on channel hangs up.
    - `g`
        - `grp` - Only spy on channels in which one or more of the groups listed in *grp* matches one or more groups from the `SPYGROUP` variable set on the channel to be spied upon.
    - `n` - Say the name of the person being spied on if that person has recorded his/her name. If a context is specified, then that voicemail context will be searched when retrieving the name, otherwise the `default` context be used when searching for the name (i.e. if SIP/1000 is the channel being spied on and no mailbox is specified, then `1000` will be used when searching for the name).
        - `mailbox`
        - `context`
    - `o` - Only listen to audio coming from this channel.
    - `q` - Don't play a beep when beginning to spy on a channel, or speak the selected channel name.
    - `r` - Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is `chanspy`.
        - `basename`

- s - Skip the playback of the channel type (i.e. SIP, IAX, etc) when speaking the selected channel name.
- S - Stop when there are no more extensions left to spy on.
- v - Adjust the initial volume in the range from -4 to 4. A negative value refers to a quieter setting.
    - value
- w - Enable whisper mode, so the spying channel can talk to the spied-on channel.
- W - Enable private whisper mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.
- x
    - digit - Specify a DTMF digit that can be used to exit the application.
- X - Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the SPY_EXIT_CONTEXT channel variable. The name of the last channel that was spied on will be stored in the SPY_CHANNEL variable.

See Also

- Asterisk 11 Application_ChanSpy

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_ExternalIVR

ExternalIVR()

Synopsis

Interfaces with an external IVR application.

Description

Either forks a process to run given command or makes a socket to connect to given host and starts a generator on the channel. The generator's play list is controlled by the external application, which can add and clear entries via simple commands issued over its stdout. The external application will receive all DTMF events received on the channel, and notification if the channel is hung up. The received on the channel, and notification if the channel is hung up. The application will not be forcibly terminated when the channel is hung up. For more information see `doc/AST.pdf`.

Syntax

```
ExternalIVR(arg1arg2[...],options)
```

Arguments

- `command|ivr://host`
    - `arg1`
    - `arg2`
- `options`
    - `n` - Tells ExternalIVR() not to answer the channel.
    - `i` - Tells ExternalIVR() not to send a hangup and exit when the channel receives a hangup, instead it sends an `I` informative message meaning that the external application MUST hang up the call with an `H` command.
    - `d` - Tells ExternalIVR() to run on a channel that has been hung up and will not look for hangups. The external application must exit with an `E` command.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Festival

Festival()

Synopsis

Say text to the user.

Description

Connect to Festival, send the argument, get back the waveform, play it to the user, allowing any given interrupt keys to immediately terminate and return the value, or `any` to allow any number back (useful in dialplan).

Syntax

```
Festival(text,intkeys)
```

Arguments

- `text`
- `intkeys`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Flash

Flash()

Synopsis

Flashes a DAHDI Trunk.

Description

Performs a flash on a DAHDI trunk. This can be used to access features provided on an incoming analogue circuit such as conference and call waiting. Use with SendDTMF() to perform external transfers.

Syntax

```
Flash()
```

Arguments

See Also

- Asterisk 11 Application_SendDTMF

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_FollowMe

FollowMe()

Synopsis

Find-Me/Follow-Me application.

Description

This application performs Find-Me/Follow-Me functionality for the caller as defined in the profile matching the *followmeid* parameter in `followme.conf`. If the specified *followmeid* profile doesn't exist in `followme.conf`, execution will be returned to the dialplan and call execution will continue at the next priority.

Returns -1 on hangup.

Syntax

```
FollowMe(followmeid,options)
```

Arguments

- `followmeid`
- `options`
    - `a` - Record the caller's name so it can be announced to the callee on each step.
    - `B` - Before initiating the outgoing call(s), Gosub to the specified location using the current channel.
        - `context`
        - `exten`
        - `priority`
            - `arg1`
            - `argN`
    - `b` - Before initiating an outgoing call, Gosub to the specified location using the newly created channel. The Gosub will be executed for each destination channel.
        - `context`
        - `exten`
        - `priority`
            - `arg1`
            - `argN`
    - `d` - Disable the 'Please hold while we try to connect your call' announcement.
    - `I` - Asterisk will ignore any connected line update requests it may receive on this dial attempt.
    - `l` - Disable local call optimization so that applications with audio hooks between the local bridge don't get dropped when the calls get joined directly.
    - `N` - Don't answer the incoming call until we're ready to connect the caller or give up.
    - `n` - Playback the unreachable status message if we've run out of steps or the callee has elected not to be reachable.
    - `s` - Playback the incoming status message prior to starting the follow-me step(s)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ForkCDR

ForkCDR()

Synopsis

Forks the Call Data Record.

Description

Causes the Call Data Record to fork an additional cdr record starting from the time of the fork call. This new cdr record will be linked to end of the list of cdr records attached to the channel.The original CDR has a LOCKED flag set, which forces most cdr operations to skip it, except for the functions that set the answer and end times, which ignore the LOCKED flag. This allows all the cdr records in the channel to be 'ended' together when the channel is closed.

The CDR() func (when setting CDR values) normally ignores the LOCKED flag also, but has options to vary its behavior. The 'T' option (described below), can override this behavior, but beware the risks.

First, this app finds the last cdr record in the list, and makes a copy of it. This new copy will be the newly forked cdr record. Next, this new record is linked to the end of the cdr record list. Next, The new cdr record is RESET (unless you use an option to prevent this)

This means that:

1. All flags are unset on the cdr record

2. the start, end, and answer times are all set to zero.

3. the billsec and duration fields are set to zero.

4. the start time is set to the current time.

5. the disposition is set to NULL.

Next, unless you specified the `v` option, all variables will be removed from the original cdr record. Thus, the `v` option allows any CDR variables to be replicated to all new forked cdr records. Without the `v` option, the variables on the original are effectively moved to the new forked cdr record.

Next, if the `s` option is set, the provided variable and value are set on the original cdr record.

Next, if the `a` option is given, and the original cdr record has an answer time set, then the new forked cdr record will have its answer time set to its start time. If the old answer time were carried forward, the answer time would be earlier than the start time, giving strange duration and billsec times.

If the `d` option was specified, the disposition is copied from the original cdr record to the new forked cdr. If the `D` option was specified, the destination channel field in the new forked CDR is erased. If the `e` option was specified, the 'end' time for the original cdr record is set to the current time. Future hang-up or ending events will not override this time stamp. If the `A` option is specified, the original cdr record will have it ANS_LOCKED flag set, which prevent future answer events from updating the original cdr record's disposition. Normally, an `ANSWERED` event would mark all cdr records in the chain as `ANSWERED`. If the `T` option is specified, the original cdr record will have its `DONT_TOUCH` flag set, which will force the cdr_answer, cdr_end, and cdr_setvar functions to leave that cdr record alone.

And, last but not least, the original cdr record has its LOCKED flag set. Almost all internal CDR functions (except for the funcs that set the end, and answer times, and set a variable) will honor this flag and leave a LOCKED cdr record alone. This means that the newly created forked cdr record will be affected by events transpiring within Asterisk, with the previously noted exceptions.

Syntax

```
ForkCDR(options)
```

Arguments

- `options`
    - `a` - Update the answer time on the NEW CDR just after it's been inited. The new CDR may have been answered already. The reset that forkcdr does will erase the answer time. This will bring it back, but the answer time will be a copy of the fork/start time. It will only do this if the initial cdr was indeed already answered.
    - `A` - Lock the original CDR against the answer time being updated. This will allow the disposition on the original CDR to remain the same.
    - `d` - Copy the disposition forward from the old cdr, after the init.
    - `D` - Clear the `dstchannel` on the new CDR after reset.
    - `e` - End the original CDR. Do this after all the necessary data is copied from the original CDR to the new forked CDR.

- `r` - Do **NOT** reset the new cdr.
- `s(name=val)` - Set the CDR var *name* in the original CDR, with value *val*.
- `T` - Mark the original CDR with a DONT_TOUCH flag. setvar, answer, and end cdr funcs will obey this flag; normally they don't honor the LOCKED flag set on the original CDR record.
- `v` - When the new CDR is forked, it gets a copy of the vars attached to the current CDR. The vars attached to the original CDR are removed unless this option is specified.

See Also

- Asterisk 11 Function_CDR
- Asterisk 11 Application_NoCDR
- Asterisk 11 Application_ResetCDR

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_GetCPEID

GetCPEID()

Synopsis

Get ADSI CPE ID.

Description

Obtains and displays ADSI CPE ID and other information in order to properly setup `dahdi.conf` for on-hook operations.

Syntax

```
GetCPEID()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Gosub

Gosub()

Synopsis

Jump to label, saving return address.

Description

Jumps to the label specified, saving the return address.

Syntax

```
Gosub(context,extenarg1[...]argN)
```

Arguments

- `context`
- `exten`
- `priority`
    - `arg1`
    - `argN`

See Also

- Asterisk 11 Application_GosubIf
- Asterisk 11 Application_Macro
- Asterisk 11 Application_Goto
- Asterisk 11 Application_Return
- Asterisk 11 Application_StackPop

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_GosubIf

GosubIf()

Synopsis

Conditionally jump to label, saving return address.

Description

If the condition is true, then jump to labeliftrue. If false, jumps to labeliffalse, if specified. In either case, a jump saves the return point in the dialplan, to be returned to with a Return.

Syntax

```
GosubIf(conditionlabeliftrue:labeliffalse)
```

Arguments

- `condition`
- `destination`
    - `labeliftrue` - Continue at *labeliftrue* if the condition is true. Takes the form similar to Goto() of [[context,]extension,]priority.
        - `arg1`
        - `argN`
    - `labeliffalse` - Continue at *labeliffalse* if the condition is false. Takes the form similar to Goto() of [[context,]extension,]priority.
        - `arg1`
        - `argN`

See Also

- Asterisk 11 Application_Gosub
- Asterisk 11 Application_Return
- Asterisk 11 Application_MacroIf
- Asterisk 11 Function_IF
- Asterisk 11 Application_GotoIf
- Asterisk 11 Application_Goto

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Goto

Goto()

Synopsis

Jump to a particular priority, extension, or context.

Description

This application will set the current context, extension, and priority in the channel structure. After it completes, the pbx engine will continue dialplan execution at the specified location. If no specific *extension*, or *extension* and *context*, are specified, then this application will just set the specified *priority* of the current extension.

At least a *priority* is required as an argument, or the goto will return a `-1`,and the channel and call will be terminated.

If the location that is put into the channel information is bogus, and asterisk cannot find that location in the dialplan, then the execution engine will try to find and execute the code in the `i` (invalid) extension in the current context. If that does not exist, it will try to execute the `h` extension. If neither the `h` nor `i` exte nsions have been defined, the channel is hung up, and the execution of instructions on the channel is terminated. What this means is that, for example, you specify a context that does not exist, then it will not be possible to find the `h` or `i` extensions, and the call will terminate!

Syntax

```
Goto(context,extensions,priority)
```

Arguments

- `context`
- `extensions`
- `priority`

See Also

- Asterisk 11 Application_GotoIf
- Asterisk 11 Application_GotoIfTime
- Asterisk 11 Application_Gosub
- Asterisk 11 Application_Macro

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_GotoIf

GotoIf()

Synopsis

Conditional goto.

Description

This application will set the current context, extension, and priority in the channel structure based on the evaluation of the given condition. After this application completes, the pbx engine will continue dialplan execution at the specified location in the dialplan. The labels are specified with the same syntax as used within the Goto application. If the label chosen by the condition is omitted, no jump is performed, and the execution passes to the next instruction. If the target location is bogus, and does not exist, the execution engine will try to find and execute the code in the `i` (invalid) extension in the current context. If that does not exist, it will try to execute the `h` extension. If neither the `h` nor `i` extensions have been defined, the channel is hung up, and the execution of instructions on the channel is terminated. Remember that this command can set the current context, and if the context specified does not exist, then it will not be able to find any 'h' or 'i' extensions there, and the channel and call will both be terminated!.

Syntax

```
GotoIf(conditionlabeliftrue:labeliffalse)
```

Arguments

- `condition`
- `destination`
    - `labeliftrue` - Continue at *labeliftrue* if the condition is true. Takes the form similar to Goto() of [[context,]extension,]priority.
    - `labeliffalse` - Continue at *labeliffalse* if the condition is false. Takes the form similar to Goto() of [[context,]extension,]priority.

See Also

- Asterisk 11 Application_Goto
- Asterisk 11 Application_GotoIfTime
- Asterisk 11 Application_GosubIf
- Asterisk 11 Application_MacroIf

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_GotoIfTime

GotoIfTime()

Synopsis

Conditional Goto based on the current time.

Description

This application will set the context, extension, and priority in the channel structure based on the evaluation of the given time specification. After this application completes, the pbx engine will continue dialplan execution at the specified location in the dialplan. If the current time is within the given time specification, the channel will continue at *labeliftrue*. Otherwise the channel will continue at *labeliffalse*. If the label chosen by the condition is omitted, no jump is performed, and execution passes to the next instruction. If the target jump location is bogus, the same actions would be taken as for `Goto`. Further information on the time specification can be found in examples illustrating how to do time-based context includes in the dialplan.

Syntax

```
GotoIfTime(timesweekdaysmdaysmonths[timezone]labeliftrue:labeliffalse)
```

Arguments

- `condition`
    - `times`
    - `weekdays`
    - `mdays`
    - `months`
    - `timezone`
- `destination`
    - `labeliftrue` - Continue at *labeliftrue* if the condition is true. Takes the form similar to Goto() of [[context,]extension,]priority.
    - `labeliffalse` - Continue at *labeliffalse* if the condition is false. Takes the form similar to Goto() of [[context,]extension,]priority.

See Also

- Asterisk 11 Application_GotoIf
- Asterisk 11 Application_Goto
- Asterisk 11 Function_IFTIME
- Asterisk 11 Function_TESTTIME

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Hangup

Hangup()

Synopsis

Hang up the calling channel.

Description

This application will hang up the calling channel.

Syntax

```
Hangup(causecode)
```

Arguments

- `causecode` - If a *causecode* is given the channel's hangup cause will be set to the given value.

See Also

- Asterisk 11 Application_Answer
- Asterisk 11 Application_Busy
- Asterisk 11 Application_Congestion

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_HangupCauseClear

HangupCauseClear()

Synopsis

Clears hangup cause information from the channel that is available through HANGUPCAUSE.

Description

Clears all channel-specific hangup cause information from the channel. This is never done automatically (i.e. for new Dial()s).

See Also

- Asterisk 11 Function_HANGUPCAUSE
- Asterisk 11 Function_HANGUPCAUSE_KEYS

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_IAX2Provision

IAX2Provision()

Synopsis

Provision a calling IAXy with a given template.

Description

Provisions the calling IAXy (assuming the calling entity is in fact an IAXy) with the given *template*. Returns `-1` on error or `0` on success.

Syntax

```
IAX2Provision(template)
```

Arguments

- `template` - If not specified, defaults to `default`.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ICES

ICES()

Synopsis

Encode and stream using 'ices'.

Description

Streams to an icecast server using ices (available separately). A configuration file must be supplied for ices (see contrib/asterisk-ices.xml).

> **Note**
> ICES version 2 client and server required.

Syntax

```
ICES(config)
```

Arguments

- `config` - ICES configuration file.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ImportVar

ImportVar()

Synopsis

Import a variable from a channel into a new variable.

Description

This application imports a *variable* from the specified *channel* (as opposed to the current one) and stores it as a variable (*newvar*) in the current channel (the channel that is calling this application). Variables created by this application have the same inheritance properties as those created with the `Set` application.

Syntax

```
ImportVar(newvarchannelnamevariable)
```

Arguments

- `newvar`
- `vardata`
    - `channelname`
    - `variable`

See Also

- Asterisk 11 Application_Set

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Incomplete

Incomplete()

Synopsis

Returns AST_PBX_INCOMPLETE value.

Description

Signals the PBX routines that the previous matched extension is incomplete and that further input should be allowed before matching can be considered to be complete. Can be used within a pattern match when certain criteria warrants a longer match.

Syntax

```
Incomplete(n)
```

Arguments

- n - If specified, then Incomplete will not attempt to answer the channel first.

> **Note**
> Most channel types need to be in Answer state in order to receive DTMF.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_IVRDemo

IVRDemo()

Synopsis

IVR Demo Application.

Description

This is a skeleton application that shows you the basic structure to create your own asterisk applications and demonstrates the IVR demo.

Syntax

```
IVRDemo(filename)
```

Arguments

- `filename`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberJoin

Moved to Asterisk 11 Application_JabberJoin_res_xmpp.

Asterisk 11 Application_JabberJoin_res_jabber

JabberJoin() - [res_jabber]

Synopsis

Join a chat room

Description

Allows Asterisk to join a chat room.

Syntax

```
JabberJoin(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk will use in the chat room.

> **Note**
> If a different nickname is supplied to an already joined room, the old nick will be changed to the new one.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberJoin_res_xmpp

JabberJoin() - [res_xmpp]

Synopsis

Join a chat room

Description

Allows Asterisk to join a chat room.

Syntax

```
JabberJoin(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk will use in the chat room.

> **Note**
> If a different nickname is supplied to an already joined room, the old nick will be changed to the new one.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberLeave

Moved to Asterisk 11 Application_JabberLeave_res_xmpp.

Asterisk 11 Application_JabberLeave_res_jabber

JabberLeave() - [res_jabber]

Synopsis

Leave a chat room

Description

Allows Asterisk to leave a chat room.

Syntax

```
JabberLeave(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk uses in the chat room.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberLeave_res_xmpp

JabberLeave() - [res_xmpp]

Synopsis

Leave a chat room

Description

Allows Asterisk to leave a chat room.

Syntax

```
JabberLeave(Jabber,RoomJID[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Nickname` - The nickname Asterisk uses in the chat room.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberSend

Moved to Asterisk 11 Application_JabberSend_res_xmpp.

Asterisk 11 Application_JabberSend_res_jabber

JabberSend() - [res_jabber]

Synopsis

Sends an XMPP message to a buddy.

Description

Sends the content of *message* as text message from the given *account* to the buddy identified by *jid*

Example: JabberSend(asterisk,bob@domain.com,Hello world) sends "Hello world" to *bob@domain.com* as an XMPP message from the account *asterisk*, configured in jabber.conf.

Syntax

```
JabberSend(account,jid,message)
```

Arguments

- `account` - The local named account to listen on (specified in jabber.conf)
- `jid` - Jabber ID of the buddy to send the message to. It can be a bare JID (username@domain) or a full JID (username@domain/resource).
- `message` - The message to send.

See Also

- Asterisk 11 Function_JABBER_STATUS_res_jabber
- Asterisk 11 Function_JABBER_RECEIVE_res_jabber

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_JabberSend_res_xmpp

JabberSend() - [res_xmpp]

Synopsis

Sends an XMPP message to a buddy.

Description

Sends the content of *message* as text message from the given *account* to the buddy identified by *jid*

Example: JabberSend(asterisk,bob@domain.com,Hello world) sends "Hello world" to *bob@domain.com* as an XMPP message from the account *asterisk*, configured in xmpp.conf.

Syntax

```
JabberSend(account,jid,message)
```

Arguments

- `account` - The local named account to listen on (specified in xmpp.conf)
- `jid` - Jabber ID of the buddy to send the message to. It can be a bare JID (username@domain) or a full JID (username@domain/resource).
- `message` - The message to send.

See Also

- Asterisk 11 Function_JABBER_STATUS_res_xmpp
- Asterisk 11 Function_JABBER_RECEIVE_res_xmpp

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_JabberSendGroup

Moved to Asterisk 11 Application_JabberSendGroup_res_xmpp.

Asterisk 11 Application_JabberSendGroup_res_jabber

JabberSendGroup() - [res_jabber]

Synopsis

Send a Jabber Message to a specified chat room

Description

Allows user to send a message to a chat room via XMPP.

> **Note**
> To be able to send messages to a chat room, a user must have previously joined it. Use the *JabberJoin* function to do so.

Syntax

```
JabberSendGroup(Jabber,RoomJID,Message[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Message` - Message to be sent to the chat room.
- `Nickname` - The nickname Asterisk uses in the chat room.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberSendGroup_res_xmpp

JabberSendGroup() - [res_xmpp]

Synopsis

Send a Jabber Message to a specified chat room

Description

Allows user to send a message to a chat room via XMPP.

> **Note**
> To be able to send messages to a chat room, a user must have previously joined it. Use the *JabberJoin* function to do so.

Syntax

```
JabberSendGroup(Jabber,RoomJID,Message[,Nickname])
```

Arguments

- `Jabber` - Client or transport Asterisk uses to connect to Jabber.
- `RoomJID` - XMPP/Jabber JID (Name) of chat room.
- `Message` - Message to be sent to the chat room.
- `Nickname` - The nickname Asterisk uses in the chat room.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberStatus

Moved to Asterisk 11 Application_JabberStatus_res_xmpp.

Asterisk 11 Application_JabberStatus_res_jabber

JabberStatus() - [res_jabber]

Synopsis

Retrieve the status of a jabber list member

Description

This application is deprecated. Please use the JABBER_STATUS() function instead.

Retrieves the numeric status associated with the specified buddy *JID*. The return value in the _Variable_will be one of the following.

- `1` - Online.
- `2` - Chatty.
- `3` - Away.
- `4` - Extended Away.
- `5` - Do Not Disturb.
- `6` - Offline.
- `7` - Not In Roster.

Syntax

```
JabberStatus(Jabber,JID,Variable)
```

Arguments

- `Jabber` - Client or transport Asterisk users to connect to Jabber.
- `JID` - XMPP/Jabber JID (Name) of recipient.
- `Variable` - Variable to store the status of requested user.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JabberStatus_res_xmpp

JabberStatus() - [res_xmpp]

Synopsis

Retrieve the status of a jabber list member

Description

This application is deprecated. Please use the JABBER_STATUS() function instead.

Retrieves the numeric status associated with the specified buddy *JID*. The return value in the _Variable_will be one of the following.

- `1` - Online.
- `2` - Chatty.
- `3` - Away.
- `4` - Extended Away.
- `5` - Do Not Disturb.
- `6` - Offline.
- `7` - Not In Roster.

Syntax

```
JabberStatus(Jabber,JID,Variable)
```

Arguments

- `Jabber` - Client or transport Asterisk users to connect to Jabber.
- `JID` - XMPP/Jabber JID (Name) of recipient.
- `Variable` - Variable to store the status of requested user.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_JACK

JACK()

Synopsis

Jack Audio Connection Kit

Description

When executing this application, two jack ports will be created; one input and one output. Other applications can be hooked up to these ports to access audio coming from, or being send to the channel.

Syntax

```
JACK([options])
```

Arguments

- `options`
    - `s`
        - `name` - Connect to the specified jack server name
    - `i`
        - `name` - Connect the output port that gets created to the specified jack input port
    - `o`
        - `name` - Connect the input port that gets created to the specified jack output port
    - `c`
        - `name` - By default, Asterisk will use the channel name for the jack client name.
          Use this option to specify a custom client name.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Log

Log()

Synopsis

Send arbitrary text to a selected log level.

Description

Sends an arbitrary text message to a selected log level.

Syntax

```
Log(level,message)
```

Arguments

- `level` - Level must be one of `ERROR`, `WARNING`, `NOTICE`, `DEBUG`, `VERBOSE` or `DTMF`.
- `message` - Output text message.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Macro

Macro()

Synopsis

Macro Implementation.

Description

Executes a macro using the context macro- *name*, jumping to the `s` extension of that context and executing each step, then returning when the steps end.

The calling extension, context, and priority are stored in `MACRO_EXTEN`, `MACRO_CONTEXT` and `MACRO_PRIORITY` respectively. Arguments become `ARG1`, `ARG2`, etc in the macro context.

If you Goto out of the Macro context, the Macro will terminate and control will be returned at the location of the Goto.

If `MACRO_OFFSET` is set at termination, Macro will attempt to continue at priority MACRO_OFFSET + N + 1 if such a step exists, and N + 1 otherwise.

> **Warning**
> Because of the way Macro is implemented (it executes the priorities contained within it via sub-engine), and a fixed per-thread memory stack allowance, macros are limited to 7 levels of nesting (macro calling macro calling macro, etc.); It may be possible that stack-intensive applications in deeply nested macros could cause asterisk to crash earlier than this limit. It is advised that if you need to deeply nest macro calls, that you use the Gosub application (now allows arguments like a Macro) with explict Return() calls instead.

> **Warning**
> Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
Macro(namearg1arg2[...])
```

Arguments

- `name` - The name of the macro
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 11 Application_MacroExit
- Asterisk 11 Application_Goto
- Asterisk 11 Application_Gosub

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MacroExclusive

MacroExclusive()

Synopsis

Exclusive Macro Implementation.

Description

Executes macro defined in the context macro- *name*. Only one call at a time may run the macro. (we'll wait if another call is busy executing in the Macro)

Arguments and return values as in application Macro()

> **Warning**
> Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
MacroExclusive(name,arg1,arg2[,...])
```

Arguments

- `name` - The name of the macro
- `arg1`
- `arg2`

See Also

- Asterisk 11 Application_Macro

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MacroExit

MacroExit()

Synopsis

Exit from Macro.

Description

Causes the currently running macro to exit as if it had ended normally by running out of priorities to execute. If used outside a macro, will likely cause unexpected behavior.

Syntax

```
MacroExit()
```

Arguments

See Also

- Asterisk 11 Application_Macro

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MacroIf

MacroIf()

Synopsis

Conditional Macro implementation.

Description

Executes macro defined in *macroiftrue* if *expr* is true (otherwise *macroiffalse* if provided)

Arguments and return values as in application Macro()

> **Warning**
> Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
MacroIf(exprmacroiftrue:macroiffalse)
```

Arguments

- `expr`
- `destination`
    - `macroiftrue`
        - `macroiftrue`
        - `arg1`
    - `macroiffalse`
        - `macroiffalse`
        - `arg1`

See Also

- Asterisk 11 Application_GotoIf
- Asterisk 11 Application_GosubIf
- Asterisk 11 Function_IF

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MailboxExists

MailboxExists()

Synopsis

Check to see if Voicemail mailbox exists.

Description

> **Note**
> DEPRECATED. Use VM_INFO(mailbox[@context],exists) instead.

Check to see if the specified *mailbox* exists. If no voicemail *context* is specified, the `default` context will be used.

This application will set the following channel variable upon completion:

- `VMBOXEXISTSSTATUS` - This will contain the status of the execution of the MailboxExists application. Possible values include:
    - SUCCESS
    - FAILED

Syntax

```
MailboxExists(mailbox@context,options)
```

Arguments

- `mailbox`
    - `mailbox`
    - `context`
- `options` - None options.

See Also

- Asterisk 11 Function_VM_INFO

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_MeetMe

MeetMe()

Synopsis

MeetMe conference bridge.

Description

Enters the user into a specified MeetMe conference. If the *confno* is omitted, the user will be prompted to enter one. User can exit the conference by hangup, or if the p option is specified, by pressing #.

> **Note**
> The DAHDI kernel modules and a functional DAHDI timing source (see dahdi_test) must be present for conferencing to operate properly. In addition, the chan_dahdi channel driver must be loaded for the `i` and `r` options to operate at all.

Syntax

```
MeetMe(confno,options,pin)
```

Arguments

- `confno` - The conference number
- `options`
  - `a` - Set admin mode.
  - `A` - Set marked mode.
  - `b` - Run AGI script specified in `MEETME_AGI_BACKGROUND` Default: `conf-background.agi`.
  - `c` - Announce user(s) count on joining a conference.
  - `C` - Continue in dialplan when kicked out of conference.
  - `d` - Dynamically add conference.
  - `D` - Dynamically add conference, prompting for a PIN.
  - `e` - Select an empty conference.
  - `E` - Select an empty pinless conference.
  - `F` - Pass DTMF through the conference.
  - `G` - Play an intro announcement in conference.
    - `x` - The file to playback
  - `i` - Announce user join/leave with review.
  - `I` - Announce user join/leave without review.
  - `k` - Close the conference if there's only one active participant left at exit.
  - `l` - Set listen only mode (Listen only, no talking).
  - `m` - Set initially muted.
  - `M` - Enable music on hold when the conference has a single caller. Optionally, specify a musiconhold class to use. If one is not provided, it will use the channel's currently set music class, or `default`.
    - `class`
  - `n` - Disable the denoiser. By default, if `func_speex` is loaded, Asterisk will apply a denoiser to channels in the MeetMe conference. However, channel drivers that present audio with a varying rate will experience degraded performance with a denoiser attached. This parameter allows a channel joining the conference to choose not to have a denoiser attached without having to unload `func_speex`.
  - `o` - Set talker optimization - treats talkers who aren't speaking as being muted, meaning (a) No encode is done on transmission and (b) Received audio that is not registered as talking is omitted causing no buildup in background noise.
  - `p` - Allow user to exit the conference by pressing `#` (default) or any of the defined keys. Dial plan execution will continue at the next priority following MeetMe. The key used is set to channel variable `MEETME_EXIT_KEY`.
    - `keys`
  - `P` - Always prompt for the pin even if it is specified.
  - `q` - Quiet mode (don't play enter/leave sounds).
  - `r` - Record conference (records as `MEETME_RECORDINGFILE` using format `MEETME_RECORDINGFORMAT`. Default filename is `meetme-conf-rec-${CONFNO}-${UNIQUEID}` and the default format is wav.

        - `s` - Present menu (user or admin) when `*` is received (send to menu).
        - `t` - Set talk only mode. (Talk only, no listening).
        - `T` - Set talker detection (sent to manager interface and meetme list).
        - `v` - Announce when a user is joining or leaving the conference. Use the voicemail greeting as the announcement. If the i or I options are set, the application will fall back to them if no voicemail greeting can be found.
            - `mailbox@context` - The mailbox and voicemail context to play from. If no context provided, assumed context is default.
        - `w` - Wait until the marked user enters the conference.
            - `secs`
        - `x` - Leave the conference when the last marked user leaves.
        - `X` - Allow user to exit the conference by entering a valid single digit extension `MEETME_EXIT_CONTEXT` or the current context if that variable is not defined.
        - `1` - Do not play message when first person enters
        - `S` - Kick the user *x* seconds **after** he entered into the conference.
            - `x`
        - `L` - Limit the conference to *x* ms. Play a warning when *y* ms are left. Repeat the warning every *z* ms. The following special variables can be used with this option:
            - `CONF_LIMIT_TIMEOUT_FILE` - File to play when time is up.
            - `CONF_LIMIT_WARNING_FILE` - File to play as warning if *y* is defined. The default is to say the time remaining.
            - `x`
            - `y`
            - `z`
- `pin`

See Also

- Asterisk 11 Application_MeetMeCount
- Asterisk 11 Application_MeetMeAdmin
- Asterisk 11 Application_MeetMeChannelAdmin

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r382322

Asterisk 11 Application_MeetMeAdmin

MeetMeAdmin()

Synopsis

MeetMe conference administration.

Description

Run admin *command* for conference *confno*.

Will additionally set the variable `MEETMEADMINSTATUS` with one of the following values:

- `MEETMEADMINSTATUS`
  - NOPARSE - Invalid arguments.
  - NOTFOUND - User specified was not found.
  - FAILED - Another failure occurred.
  - OK - The operation was completed successfully.

Syntax

```
MeetMeAdmin(confno,command,user)
```

Arguments

- `confno`
- `command`
  - `e` - Eject last user that joined.
  - `E` - Extend conference end time, if scheduled.
  - `k` - Kick one user out of conference.
  - `K` - Kick all users out of conference.
  - `l` - Unlock conference.
  - `L` - Lock conference.
  - `m` - Unmute one user.
  - `M` - Mute one user.
  - `n` - Unmute all users in the conference.
  - `N` - Mute all non-admin users in the conference.
  - `r` - Reset one user's volume settings.
  - `R` - Reset all users volume settings.
  - `s` - Lower entire conference speaking volume.
  - `S` - Raise entire conference speaking volume.
  - `t` - Lower one user's talk volume.
  - `T` - Raise one user's talk volume.
  - `u` - Lower one user's listen volume.
  - `U` - Raise one user's listen volume.
  - `v` - Lower entire conference listening volume.
  - `V` - Raise entire conference listening volume.
- `user`

See Also

- Asterisk 11 Application_MeetMe

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MeetMeChannelAdmin

MeetMeChannelAdmin()

Synopsis

MeetMe conference Administration (channel specific).

Description

Run admin *command* for a specific *channel* in any conference.

Syntax

```
MeetMeChannelAdmin(channel,command)
```

Arguments

- `channel`
- `command`
    - `k` - Kick the specified user out of the conference he is in.
    - `m` - Unmute the specified user.
    - `M` - Mute the specified user.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MeetMeCount

MeetMeCount()

Synopsis

MeetMe participant count.

Description

Plays back the number of users in the specified MeetMe conference. If *var* is specified, playback will be skipped and the value will be returned in the variable. Upon application completion, MeetMeCount will hangup the channel, unless priority `n+1` exists, in which case priority progress will continue.

Syntax

```
MeetMeCount(confno,var)
```

Arguments

- `confno` - Conference number.
- `var`

See Also

- Asterisk 11 Application_MeetMe

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MessageSend

MessageSend()

Synopsis

Send a text message.

Description

Send a text message. The body of the message that will be sent is what is currently set to `MESSAGE(body)`. The technology chosen for sending the message is determined based on a prefix to the `to` parameter.

This application sets the following channel variables:

- `MESSAGE_SEND_STATUS` - This is the message delivery status returned by this application.
  - INVALID_PROTOCOL - No handler for the technology part of the URI was found.
  - INVALID_URI - The protocol handler reported that the URI was not valid.
  - SUCCESS - Successfully passed on to the protocol handler, but delivery has not necessarily been guaranteed.
  - FAILURE - The protocol handler reported that it was unabled to deliver the message for some reason.

Syntax

```
MessageSend(to[,from])
```

Arguments

- `to` - A To URI for the message.

Technology: SIP
Specifying a prefix of `sip:` will send the message as a SIP MESSAGE request.

Technology: XMPP
Specifying a prefix of `xmpp:` will send the message as an XMPP chat message.

- `from` - A From URI for the message if needed for the message technology being used to send this message.

Technology: SIP
The `from` parameter can be a configured peer name or in the form of "display-name" <URI>.

Technology: XMPP
Specifying a prefix of `xmpp:` will specify the account defined in `xmpp.conf` to send the message from. Note that this field is required for XMPP messages.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Milliwatt

Milliwatt()

Synopsis

Generate a Constant 1004Hz tone at 0dbm (mu-law).

Description

Previous versions of this application generated the tone at 1000Hz. If for some reason you would prefer that behavior, supply the o option to get the old behavior.

Syntax

```
Milliwatt(options)
```

Arguments

- options
    - o - Generate the tone at 1000Hz like previous version.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MinivmAccMess

MinivmAccMess()

Synopsis

Record account specific messages.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`.

Use this application to record account specific audio/video messages for busy, unavailable and temporary messages.

Account specific directories will be created if they do not exist.

- `MVM_ACCMESS_STATUS` - This is the result of the attempt to record the specified greeting.
  FAILED is set if the file can't be created.
    - SUCCESS
    - FAILED

Syntax

```
MinivmAccMess(username@domain[,options])
```

Arguments

- `mailbox`
    - `username` - Voicemail username
    - `domain` - Voicemail domain
- `options`
    - `u` - Record the `unavailable` greeting.
    - `b` - Record the `busy` greeting.
    - `t` - Record the temporary greeting.
    - `n` - Account name.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MinivmDelete

MinivmDelete()

Synopsis

Delete Mini-Voicemail voicemail messages.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`.

It deletes voicemail file set in MVM_FILENAME or given filename.

- `MVM_DELETE_STATUS` - This is the status of the delete operation.
  - SUCCESS
  - FAILED

Syntax

```
MinivmDelete(filename)
```

Arguments

- `filename` - File to delete

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MinivmGreet

MinivmGreet()

Synopsis

Play Mini-Voicemail prompts.

Description

This application is part of the Mini-Voicemail system, configured in minivm.conf.

MinivmGreet() plays default prompts or user specific prompts for an account.

Busy and unavailable messages can be choosen, but will be overridden if a temporary message exists for the account.

- `MVM_GREET_STATUS` - This is the status of the greeting playback.
    - SUCCESS
    - USEREXIT
    - FAILED

Syntax

```
MinivmGreet(username@domain[,options])
```

Arguments

- `mailbox`
    - `username` - Voicemail username
    - `domain` - Voicemail domain
- `options`
    - `b` - Play the `busy` greeting to the calling party.
    - `s` - Skip the playback of instructions for leaving a message to the calling party.
    - `u` - Play the `unavailable` greeting.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MinivmMWI

MinivmMWI()

Synopsis

Send Message Waiting Notification to subscriber(s) of mailbox.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`.

MinivmMWI is used to send message waiting indication to any devices whose channels have subscribed to the mailbox passed in the first parameter.

Syntax

```
MinivmMWI(username@domain,urgent,new,old)
```

Arguments

- `mailbox`
    - `username` - Voicemail username
    - `domain` - Voicemail domain
- `urgent` - Number of urgent messages in mailbox.
- `new` - Number of new messages in mailbox.
- `old` - Number of old messages in mailbox.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MinivmNotify

MinivmNotify()

Synopsis

Notify voicemail owner about new messages.

Description

This application is part of the Mini-Voicemail system, configured in minivm.conf.

MiniVMnotify forwards messages about new voicemail to e-mail and pager. If there's no user account for that address, a temporary account will be used with default options (set in `minivm.conf`).

If the channel variable `MVM_COUNTER` is set, this will be used in the message file name and available in the template for the message.

If no template is given, the default email template will be used to send email and default pager template to send paging message (if the user account is configured with a paging address.

- `MVM_NOTIFY_STATUS` - This is the status of the notification attempt
    - SUCCESS
    - FAILED

Syntax

```
MinivmNotify(username@domain[,options])
```

Arguments

- `mailbox`
    - `username` - Voicemail username
    - `domain` - Voicemail domain
- `options`
    - `template` - E-mail template to use for voicemail notification

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MinivmRecord

MinivmRecord()

Synopsis

Receive Mini-Voicemail and forward via e-mail.

Description

This application is part of the Mini-Voicemail system, configured in `minivm.conf`

MiniVM records audio file in configured format and forwards message to e-mail and pager.

If there's no user account for that address, a temporary account will be used with default options.

The recorded file name and path will be stored in `MVM_FILENAME` and the duration of the message will be stored in `MVM_DURATION`

> **Note**
> If the caller hangs up after the recording, the only way to send the message and clean up is to execute in the `h` extension. The application will exit if any of the following DTMF digits are received and the requested extension exist in the current context.

- `MVM_RECORD_STATUS` - This is the status of the record operation
  - SUCCESS
  - USEREXIT
  - FAILED

Syntax

```
MinivmRecord(username@domain[,options])
```

Arguments

- `mailbox`
  - `username` - Voicemail username
  - `domain` - Voicemail domain
- `options`
  - `0` - Jump to the `o` extension in the current dialplan context.
  - `*` - Jump to the `a` extension in the current dialplan context.
  - `g` - Use the specified amount of gain when recording the voicemail message. The units are whole-number decibels (dB).
    - `gain` - Amount of gain to use

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MixMonitor

MixMonitor()

Synopsis

Record a call and mix the audio during the recording. Use of StopMixMonitor is required to guarantee the audio file is available for processing during dialplan execution.

Description

Records the audio on the current channel to the specified file.

This application does not automatically answer and should be preceeded by an application such as Answer or Progress().

> **Note**
> MixMonitor runs as an audiohook. In order to keep it running through a transfer, AUDIOHOOK_INHERIT must be set for the channel which ran mixmonitor. For more information, including dialplan configuration set for using AUDIOHOOK_INHERIT with MixMonitor, see the function documentation for AUDIOHOOK_INHERIT.

- `MIXMONITOR_FILENAME` - Will contain the filename used to record.

Syntax

```
MixMonitor(filename.extension,options,command)
```

Arguments

- `file`
  - `filename` - If *filename* is an absolute path, uses that path, otherwise creates the file in the configured monitoring directory from `asterisk.conf.`
  - `extension`
- `options`
  - `a` - Append to the file instead of overwriting it.
  - `b` - Only save audio to the file while the channel is bridged.
  - `v` - Adjust the **heard** volume by a factor of *x* (range `-4` to `4`)
    - `x`
  - `V` - Adjust the **spoken** volume by a factor of *x* (range `-4` to `4`)
    - `x`
  - `W` - Adjust both, **heard and spoken** volumes by a factor of *x* (range `-4` to `4`)
    - `x`
  - `r` - Use the specified file to record the **receive** audio feed. Like with the basic filename argument, if an absolute path isn't given, it will create the file in the configured monitoring directory.
    - `file`
  - `t` - Use the specified file to record the **transmit** audio feed. Like with the basic filename argument, if an absolute path isn't given, it will create the file in the configured monitoring directory.
    - `file`
  - `i` - Stores the MixMonitor's ID on this channel variable.
    - `chanvar`
  - `m` - Create a copy of the recording as a voicemail in the indicated **mailbox**(es) separated by commas eg. m(1111default,...). Folders can be optionally specified using the syntax: mailbox@context/folder
    - `mailbox`
- `command` - Will be executed when the recording is over.
  Any strings matching `^{X}` will be unescaped to `X`.
  All variables will be evaluated at the time MixMonitor is called.

See Also

- Asterisk 11 Application_Monitor
- Asterisk 11 Application_StopMixMonitor

* Asterisk 11 Application_PauseMonitor
* Asterisk 11 Application_UnpauseMonitor
* Asterisk 11 Function_AUDIOHOOK_INHERIT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Monitor

Monitor()

Synopsis

Monitor a channel.

Description

Used to start monitoring a channel. The channel's input and output voice packets are logged to files until the channel hangs up or monitoring is stopped by the StopMonitor application.

By default, files are stored to `/var/spool/asterisk/monitor/`. Returns `-1` if monitor files can't be opened or if the channel is already monitored, otherwise `0`.

Syntax

```
Monitor(file_format:urlbase,fname_base,options)
```

Arguments

- `file_format`
    - `file_format` - optional, if not set, defaults to `wav`
    - `urlbase`
- `fname_base` - if set, changes the filename used to the one specified.
- `options`
    - `m` - when the recording ends mix the two leg files into one and delete the two leg files. If the variable `MONITOR_EXEC` is set, the application referenced in it will be executed instead of soxmix/sox and the raw leg files will NOT be deleted automatically. soxmix/sox or `MONITOR_EXEC` is handed 3 arguments, the two leg files and a target mixed file name which is the same as the leg file names only without the in/out designator.
      If `MONITOR_EXEC_ARGS` is set, the contents will be passed on as additional arguments to `MONITOR_EXEC`. Both `MONITOR_EXEC` and the Mix flag can be set from the administrator interface.
    - `b` - Don't begin recording unless a call is bridged to another channel.
    - `i` - Skip recording of input stream (disables `m` option).
    - `o` - Skip recording of output stream (disables `m` option).

See Also

- Asterisk 11 Application_StopMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Morsecode

Morsecode()

Synopsis

Plays morse code.

Description

Plays the Morse code equivalent of the passed string.

This application does not automatically answer and should be preceeded by an application such as Answer() or Progress().

This application uses the following variables:

- `MORSEDITLEN` - Use this value in (ms) for length of dit
- `MORSETONE` - The pitch of the tone in (Hz), default is 800

Syntax

```
Morsecode(string)
```

Arguments

- `string` - String to playback as morse code to channel

See Also

- Asterisk 11 Application_SayAlpha
- Asterisk 11 Application_SayPhonetic

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_MP3Player

MP3Player()

Synopsis

Play an MP3 file or M3U playlist file or stream.

Description

Executes mpg123 to play the given location, which typically would be a mp3 filename or m3u playlist filename or a URL. Please read http://en.wikipedia.org/wiki/M3U to see how M3U playlist file format is like, Example usage would be exten => 1234,1,MP3Player(/var/lib/asterisk/playlist.m3u) User can exit by pressing any key on the dialpad, or by hanging up.

This application does not automatically answer and should be preceeded by an application such as Answer() or Progress().

Syntax

```
MP3Player(Location)
```

Arguments

- `Location` - Location of the file to be played. (argument passed to mpg123)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_MSet

MSet()

Synopsis

Set channel variable(s) or function value(s).

Description

This function can be used to set the value of channel variables or dialplan functions. When setting variables, if the variable name is prefixed with _, the variable will be inherited into channels created from the current channel If the variable name is prefixed with __, the variable will be inherited into channels created from the current channel and all children channels. MSet behaves in a similar fashion to the way Set worked in 1.2/1.4 and is thus prone to doing things that you may not expect. For example, it strips surrounding double-quotes from the right-hand side (value). If you need to put a separator character (comma or vert-bar), you will need to escape them by inserting a backslash before them. Avoid its use if possible.

Syntax

```
MSet(name1=value1name2=value2)
```

Arguments

- `set1`
  - `name1`
  - `value1`
- `set2`
  - `name2`
  - `value2`

See Also

- Asterisk 11 Application_Set

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_MusicOnHold

MusicOnHold()

Synopsis

Play Music On Hold indefinitely.

Description

Plays hold music specified by class. If omitted, the default music source for the channel will be used. Change the default class with Set(CHANNEL(musicclass)=...). If duration is given, hold music will be played specified number of seconds. If duration is ommited, music plays indefinitely. Returns `0` when done, `-1` on hangup.

This application does not automatically answer and should be preceeded by an application such as Answer() or Progress().

Syntax

```
MusicOnHold(class,duration)
```

Arguments

- `class`
- `duration`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_NBScat

NBScat()

Synopsis

Play an NBS local stream.

Description

Executes nbscat to listen to the local NBS stream. User can exit by pressing any key.

Syntax

```
NBScat()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_NoCDR

NoCDR()

Synopsis

Tell Asterisk to not maintain a CDR for the current call

Description

This application will tell Asterisk not to maintain a CDR for the current call.

Syntax

```
NoCDR()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_NoOp

NoOp()

Synopsis

Do Nothing (No Operation).

Description

This application does nothing. However, it is useful for debugging purposes.

This method can be used to see the evaluations of variables or functions without having any effect.

Syntax

```
NoOp(text)
```

Arguments

- `text` - Any text provided can be viewed at the Asterisk CLI.

See Also

- Asterisk 11 Application_Verbose
- Asterisk 11 Application_Log

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ODBC_Commit

ODBC_Commit()

Synopsis

Commits a currently open database transaction.

Description

Commits the database transaction specified by *transaction ID* or the current active transaction, if not specified.

Syntax

```
ODBC_Commit([transaction ID])
```

Arguments

- `transaction ID`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ODBC_Rollback

ODBC_Rollback()

Synopsis

Rollback a currently open database transaction.

Description

Rolls back the database transaction specified by *transaction ID* or the current active transaction, if not specified.

Syntax

```
ODBC_Rollback([transaction ID])
```

Arguments

- `transaction ID`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ODBCFinish

ODBCFinish()

Synopsis

Clear the resultset of a sucessful multirow query.

Description

For queries which are marked as mode=multirow, this will clear any remaining rows of the specified resultset.

Syntax

```
ODBCFinish(result-id)
```

Arguments

- `result-id`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Originate

Originate()

Synopsis

Originate a call.

Description

This application originates an outbound call and connects it to a specified extension or application. This application will block until the outgoing call fails or gets answered. At that point, this application will exit with the status variable set and dialplan processing will continue.

This application sets the following channel variable before exiting:

- `ORIGINATE_STATUS` - This indicates the result of the call origination.
  - FAILED
  - SUCCESS
  - BUSY
  - CONGESTION
  - HANGUP
  - RINGING
  - UNKNOWN - In practice, you should never see this value. Please report it to the issue tracker if you ever see it.

Syntax

```
Originate(tech_data,type,arg1[,arg2[,arg3[,timeout]]])
```

Arguments

- `tech_data` - Channel technology and data for creating the outbound channel. For example, SIP/1234.
- `type` - This should be `app` or `exten`, depending on whether the outbound channel should be connected to an application or extension.
- `arg1` - If the type is `app`, then this is the application name. If the type is `exten`, then this is the context that the channel will be sent to.
- `arg2` - If the type is `app`, then this is the data passed as arguments to the application. If the type is `exten`, then this is the extension that the channel will be sent to.
- `arg3` - If the type is `exten`, then this is the priority that the channel is sent to. If the type is `app`, then this parameter is ignored.
- `timeout` - Timeout in seconds. Default is 30 seconds.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_OSPAuth

OSPAuth()

Synopsis

OSP Authentication.

Description

Authenticate a call by OSP.

Input variables:

- `OSPINPEERIP` - The last hop IP address.
- `OSPINTOKEN` - The inbound OSP token.
  Output variables:

- `OSPINHANDLE` - The inbound call OSP transaction handle.
- `OSPINTIMELIMIT` - The inbound call duration limit in seconds.
  This application sets the following channel variable upon completion:

- `OSPAUTHSTATUS` - The status of OSPAuth attempt as a text string, one of
  - SUCCESS
  - FAILED
  - ERROR

Syntax

```
OSPAuth(provider,options)
```

Arguments

- `provider` - The name of the provider that authenticates the call.
- `options` - Reserverd.

See Also

- Asterisk 11 Application_OSPLookup
- Asterisk 11 Application_OSPNext
- Asterisk 11 Application_OSPFinish

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_OSPFinish

OSPFinish()

Synopsis

Report OSP entry.

Description

Report call state.

Input variables:

- OSPINHANDLE - The inbound call OSP transaction handle.
- OSPOUTHANDLE - The outbound call OSP transaction handle.
- OSPAUTHSTATUS - The OSPAuth status.
- OSPLOOKUPSTATUS - The OSPLookup status.
- OSPNEXTSTATUS - The OSPNext status.
- OSPINAUDIOQOS - The inbound call leg audio QoS string.
- OSPOUTAUDIOQOS - The outbound call leg audio QoS string.
  This application sets the following channel variable upon completion:

- OSPFINISHSTATUS - The status of the OSPFinish attempt as a text string, one of
  - SUCCESS
  - FAILED
  - ERROR

Syntax

```
OSPFinish(cause,options)
```

Arguments

- cause - Hangup cause.
- options - Reserved.

See Also

- Asterisk 11 Application_OSPAuth
- Asterisk 11 Application_OSPLookup
- Asterisk 11 Application_OSPNext

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_OSPLookup

OSPLookup()

Synopsis

Lookup destination by OSP.

Description

Looks up destination via OSP.

Input variables:

- `OSPINACTUALSRC` - The actual source device IP address in indirect mode.
- `OSPINPEERIP` - The last hop IP address.
- `OSPINTECH` - The inbound channel technology for the call.
- `OSPINHANDLE` - The inbound call OSP transaction handle.
- `OSPINTIMELIMIT` - The inbound call duration limit in seconds.
- `OSPINNETWORKID` - The inbound source network ID.
- `OSPINNPRN` - The inbound routing number.
- `OSPINNPCIC` - The inbound carrier identification code.
- `OSPINNPDI` - The inbound number portability database dip indicator.
- `OSPINSPID` - The inbound service provider identity.
- `OSPINOCN` - The inbound operator company number.
- `OSPINSPN` - The inbound service provider name.
- `OSPINALTSPN` - The inbound alternate service provider name.
- `OSPINMCC` - The inbound mobile country code.
- `OSPINMNC` - The inbound mobile network code.
- `OSPINTOHOST` - The inbound To header host part.
- `OSPINRPIDUSER` - The inbound Remote-Party-ID header user part.
- `OSPINPAIUSER` - The inbound P-Asserted-Identify header user part.
- `OSPINDIVUSER` - The inbound Diversion header user part.
- `OSPINDIVHOST` - The inbound Diversion header host part.
- `OSPINPCIUSER` - The inbound P-Charge-Info header user part.
- `OSPINCUSTOMINFON` - The inbound custom information, where `n` is the index beginning with `1` upto `8`.
  Output variables:

- `OSPOUTHANDLE` - The outbound call OSP transaction handle.
- `OSPOUTTECH` - The outbound channel technology for the call.
- `OSPDESTINATION` - The outbound destination IP address.
- `OSPOUTCALLING` - The outbound calling number.
- `OSPOUTCALLED` - The outbound called number.
- `OSPOUTNETWORKID` - The outbound destination network ID.
- `OSPOUTNPRN` - The outbound routing number.
- `OSPOUTNPCIC` - The outbound carrier identification code.
- `OSPOUTNPDI` - The outbound number portability database dip indicator.
- `OSPOUTSPID` - The outbound service provider identity.
- `OSPOUTOCN` - The outbound operator company number.
- `OSPOUTSPN` - The outbound service provider name.
- `OSPOUTALTSPN` - The outbound alternate service provider name.
- `OSPOUTMCC` - The outbound mobile country code.
- `OSPOUTMNC` - The outbound mobile network code.
- `OSPOUTTOKEN` - The outbound OSP token.
- `OSPDESTREMAILS` - The number of remained destinations.
- `OSPOUTTIMELIMIT` - The outbound call duration limit in seconds.
- `OSPOUTCALLIDTYPES` - The outbound Call-ID types.
- `OSPOUTCALLID` - The outbound Call-ID. Only for H.323.
- `OSPDIALSTR` - The outbound Dial command string.

This application sets the following channel variable upon completion:

- `OSPLOOKUPSTATUS` - The status of OSPLookup attempt as a text string, one of
  - SUCCESS
  - FAILED
  - ERROR

Syntax

```
OSPLookup(exten,provider,options)
```

Arguments

- `exten` - The exten of the call.
- `provider` - The name of the provider that is used to route the call.
- `options`
  - `h` - generate H323 call id for the outbound call
  - `s` - generate SIP call id for the outbound call. Have not been implemented
  - `i` - generate IAX call id for the outbound call. Have not been implemented

See Also

- Asterisk 11 Application_OSPAuth
- Asterisk 11 Application_OSPNext
- Asterisk 11 Application_OSPFinish

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_OSPNext

OSPNext()

Synopsis

Lookup next destination by OSP.

Description

Looks up the next destination via OSP.

Input variables:

- `OSPINHANDLE` - The inbound call OSP transaction handle.
- `OSPOUTHANDLE` - The outbound call OSP transaction handle.
- `OSPINTIMELIMIT` - The inbound call duration limit in seconds.
- `OSPOUTCALLIDTYPES` - The outbound Call-ID types.
- `OSPDESTREMAILS` - The number of remained destinations.
  Output variables:

- `OSPOUTTECH` - The outbound channel technology.
- `OSPDESTINATION` - The destination IP address.
- `OSPOUTCALLING` - The outbound calling number.
- `OSPOUTCALLED` - The outbound called number.
- `OSPOUTNETWORKID` - The outbound destination network ID.
- `OSPOUTNPRN` - The outbound routing number.
- `OSPOUTNPCIC` - The outbound carrier identification code.
- `OSPOUTNPDI` - The outbound number portability database dip indicator.
- `OSPOUTSPID` - The outbound service provider identity.
- `OSPOUTOCN` - The outbound operator company number.
- `OSPOUTSPN` - The outbound service provider name.
- `OSPOUTALTSPN` - The outbound alternate service provider name.
- `OSPOUTMCC` - The outbound mobile country code.
- `OSPOUTMNC` - The outbound mobile network code.
- `OSPOUTTOKEN` - The outbound OSP token.
- `OSPDESTREMAILS` - The number of remained destinations.
- `OSPOUTTIMELIMIT` - The outbound call duration limit in seconds.
- `OSPOUTCALLID` - The outbound Call-ID. Only for H.323.
- `OSPDIALSTR` - The outbound Dial command string.
  This application sets the following channel variable upon completion:

- `OSPNEXTSTATUS` - The status of the OSPNext attempt as a text string, one of
  - SUCCESS
  - FAILED
  - ERROR

See Also

- Asterisk 11 Application_OSPAuth
- Asterisk 11 Application_OSPLookup
- Asterisk 11 Application_OSPFinish

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Page

Page()

Synopsis

Page series of phones

Description

Places outbound calls to the given *technology/resource* and dumps them into a conference bridge as muted participants. The original caller is dumped into the conference as a speaker and the room is destroyed when the original caller leaves.

Syntax

```
Page(Technology/Resource&Technology2/Resource2[&...],options,timeout)
```

Arguments

- `Technology/Resource`
    - `Technology/Resource` - Specification of the device(s) to dial. These must be in the format of `Technology/Resource`, where *Technology* represents a particular channel driver, and *Resource* represents a resource available to that particular channel driver.
    - `Technology2/Resource2` - Optional extra devices to dial in parallel
      If you need more than one, enter them as Technology2/Resource2& Technology3/Resourse3&.....
- `options`
    - `d` - Full duplex audio
    - `i` - Ignore attempts to forward the call
    - `q` - Quiet, do not play beep to caller
    - `r` - Record the page into a file (`CONFBRIDGE(bridge,record_conference)`)
    - `s` - Only dial a channel if its device state says that it is `NOT_INUSE`
    - `A` - Play an announcement to all paged participants
        - `x` - The announcement to playback to all devices
    - `n` - Do not play announcement to caller (alters A behavior)
- `timeout` - Specify the length of time that the system will attempt to connect a call. After this duration, any page calls that have not been answered will be hung up by the system.

See Also

- Asterisk 11 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r380942

Asterisk 11 Application_Park

Park()

Synopsis

Park yourself.

Description

Used to park yourself (typically in combination with a supervised transfer to know the parking space).

If you set the `PARKINGEXTEN` variable to a parking space extension in the parking lot, Park() will attempt to park the call on that extension. If the extension is already is in use then execution will continue at the next priority.

If the `parkeddynamic` option is enabled in `features.conf` the following variables can be used to dynamically create new parking lots.

If you set the `PARKINGDYNAMIC` variable and this parking lot exists then it will be used as a template for the newly created dynamic lot. Otherwise, the default parking lot will be used.

If you set the `PARKINGDYNCONTEXT` variable then the newly created dynamic parking lot will use this context.

If you set the `PARKINGDYNEXTEN` variable then the newly created dynamic parking lot will use this extension to access the parking lot.

If you set the `PARKINGDYNPOS` variable then the newly created dynamic parking lot will use those parking postitions.

> **Note**
> This application must be used as the first extension priority to be recognized as a parking access extension. DTMF transfers and some channel drivers need this distinction to operate properly. The parking access extension in this case is treated like a dialplan hint.

> **Note**
> Parking lots automatically create and manage dialplan extensions in the parking lot context. You do not need to explicitly use this application in your dialplan. Instead, all you should do is include the parking lot context in your dialplan.

Syntax

```
Park(timeout,return_context,return_exten,return_priority,options,parking_lot_name)
```

Arguments

- `timeout` - A custom parking timeout for this parked call. Value in milliseconds.
- `return_context` - The context to return the call to after it times out.
- `return_exten` - The extension to return the call to after it times out.
- `return_priority` - The priority to return the call to after it times out.
- `options` - A list of options for this parked call.
    - `r` - Send ringing instead of MOH to the parked call.
    - `R` - Randomize the selection of a parking space.
    - `s` - Silence announcement of the parking space number.
- `parking_lot_name` - Specify in which parking lot to park a call.
  The parking lot used is selected in the following order:
  1) parking_lot_name option
  2) `PARKINGLOT` variable
  3) `CHANNEL(parkinglot)` function (Possibly preset by the channel driver.)
  4) Default parking lot.

See Also

- Asterisk 11 Application_ParkAndAnnounce
- Asterisk 11 Application_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ParkAndAnnounce

ParkAndAnnounce()

Synopsis

Park and Announce.

Description

Park a call into the parkinglot and announce the call to another channel.

The variable `PARKEDAT` will contain the parking extension into which the call was placed. Use with the Local channel to allow the dialplan to make use of this information.

Syntax

```
ParkAndAnnounce(announce:announce1[:...],timeout,dial,return_context)
```

Arguments

- `announce_template`
  - `announce` - Colon-separated list of files to announce. The word `PARKED` will be replaced by a say_digits of the extension in which the call is parked.
  - `announce1`
- `timeout` - Time in seconds before the call returns into the return context.
- `dial` - The app_dial style resource to call to make the announcement. Console/dsp calls the console.
- `return_context` - The goto-style label to jump the call back into after timeout. Default `priority+1`.

See Also

- Asterisk 11 Application_Park
- Asterisk 11 Application_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ParkedCall

ParkedCall()

Synopsis

Retrieve a parked call.

Description

Used to retrieve a parked call from a parking lot.

> **Note**
> Parking lots automatically create and manage dialplan extensions in the parking lot context. You do not need to explicitly use this application in your dialplan. Instead, all you should do is include the parking lot context in your dialplan.

Syntax

```
ParkedCall(exten,parking_lot_name)
```

Arguments

- `exten` - Parking space extension to retrieve a parked call. If not provided then the first available parked call in the parking lot will be retrieved.
- `parking_lot_name` - Specify from which parking lot to retrieve a parked call.
  The parking lot used is selected in the following order:
  1) parking_lot_name option
  2) `PARKINGLOT` variable
  3) `CHANNEL(parkinglot)` function (Possibly preset by the channel driver.)
  4) Default parking lot.

See Also

- Asterisk 11 Application_Park
- Asterisk 11 Application_ParkAndAnnounce

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_PauseMonitor

PauseMonitor()

Synopsis

Pause monitoring of a channel.

Description

Pauses monitoring of a channel until it is re-enabled by a call to UnpauseMonitor.

Syntax

```
PauseMonitor()
```

Arguments

See Also

- Asterisk 11 Application_UnpauseMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_PauseQueueMember

PauseQueueMember()

Synopsis

Pauses a queue member.

Description

Pauses (blocks calls for) a queue member. The given interface will be paused in the given queue. This prevents any calls from being sent from the queue to the interface until it is unpaused with UnpauseQueueMember or the manager interface. If no queuename is given, the interface is paused in every queue it is a member of. The application will fail if the interface is not found.

This application sets the following channel variable upon completion:

- `PQMSTATUS` - The status of the attempt to pause a queue member as a text string.
    - PAUSED
    - NOTFOUND
      Example: PauseQueueMember(,SIP/3000)

Syntax

```
PauseQueueMember(queuename,interface,options,reason)
```

Arguments

- `queuename`
- `interface`
- `options`
- `reason` - Is used to add extra information to the appropriate queue_log entries and manager events.

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Pickup

Pickup()

Synopsis

Directed extension call pickup.

Description

This application can pickup a specified ringing channel. The channel to pickup can be specified in the following ways.

1) If no *extension* targets are specified, the application will pickup a channel matching the pickup group of the requesting channel.

2) If the *extension* is specified with a *context* of the special string `PICKUPMARK` (for example 10@PICKUPMARK), the application will pickup a channel which has defined the channel variable `PICKUPMARK` with the same value as *extension* (in this example, `10`).

3) If the *extension* is specified with or without a *context*, the channel with a matching *extension* and *context* will be picked up. If no *context* is specified, the current context will be used.

> **Note**
> The *extension* is typically set on matching channels by the dial application that created the channel. The *context* is set on matching channels by the channel driver for the device.

Syntax

```
Pickup(extension&extension2[&...])
```

Arguments

- `targets`
  - `extension` - Specification of the pickup target.
    - `extension`
    - `context`
  - `extension2` - Additional specifications of pickup targets.
    - `extension2`
    - `context2`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_PickupChan

PickupChan()

Synopsis

Pickup a ringing channel.

Description

This will pickup a specified *channel* if ringing.

Syntax

```
PickupChan(Technology/Resource[&Technology2/Resource2[&...]][,options])
```

Arguments

- `Technology/Resource`
  - `Technology/Resource`
  - `Technology2/Resource2`
- `options`
  - p - Channel name specified partial name. Used when find channel by callid.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Playback

Playback()

Synopsis

Play a file.

Description

Plays back given filenames (do not put extension of wav/alaw etc). The playback command answer the channel if no options are specified. If the file is non-existant it will fail

This application sets the following channel variable upon completion:

- `PLAYBACKSTATUS` - The status of the playback attempt as a text string.
    - SUCCESS
    - FAILED
      See Also: Background (application) – for playing sound files that are interruptible

WaitExten (application) – wait for digits from caller, optionally play music on hold

Syntax

```
Playback(filename&filename2[&...],options)
```

Arguments

- `filenames`
    - `filename`
    - `filename2`
- `options` - Comma separated list of options
    - `skip` - Do not play if not answered
    - `noanswer` - Playback without answering, otherwise the channel will be answered before the sound is played.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_PlayTones

PlayTones()

Synopsis

Play a tone list.

Description

Plays a tone list. Execution will continue with the next step in the dialplan immediately while the tones continue to play.

See the sample `indications.conf` for a description of the specification of a tonelist.

Syntax

```
PlayTones(arg)
```

Arguments

- `arg` - Arg is either the tone name defined in the `indications.conf` configuration file, or a directly specified list of frequencies and durations.

See Also

- Asterisk 11 Application_StopPlayTones

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_PrivacyManager

PrivacyManager()

Synopsis

Require phone number to be entered, if no CallerID sent

Description

If no Caller*ID is sent, PrivacyManager answers the channel and asks the caller to enter their phone number. The caller is given *maxretries* attempts to do so. The application does **nothing** if Caller*ID was received on the channel.

The application sets the following channel variable upon completion:

- `PRIVACYMGRSTATUS` - The status of the privacy manager's attempt to collect a phone number from the user.
    - SUCCESS
    - FAILED

Syntax

```
PrivacyManager(maxretries,minlength,options,context)
```

Arguments

- `maxretries` - Total tries caller is allowed to input a callerid. Defaults to `3`.
- `minlength` - Minimum allowable digits in the input callerid number. Defaults to `10`.
- `options` - Position reserved for options.
- `context` - Context to check the given callerid against patterns.

See Also

- Asterisk 11 Application_Zapateller

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Proceeding

Proceeding()

Synopsis

Indicate proceeding.

Description

This application will request that a proceeding message be provided to the calling channel.

Syntax

```
Proceeding()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Progress

Progress()

Synopsis

Indicate progress.

Description

This application will request that in-band progress information be provided to the calling channel.

Syntax

```
Progress()
```

Arguments

See Also

- Asterisk 11 Application_Busy
- Asterisk 11 Application_Congestion
- Asterisk 11 Application_Ringing
- Asterisk 11 Application_Playtones

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Queue

Queue()

Synopsis

Queue a call for a call queue.

Description

In addition to transferring the call, a call may be parked and then picked up by another user.

This application will return to the dialplan if the queue does not exist, or any of the join options cause the caller to not enter the queue.

This application does not automatically answer and should be preceeded by an application such as Answer(), Progress(), or Ringing().

This application sets the following channel variable upon completion:

- `QUEUESTATUS` - The status of the call as a text string.
    - TIMEOUT
    - FULL
    - JOINEMPTY
    - LEAVEEMPTY
    - JOINUNAVAIL
    - LEAVEUNAVAIL
    - CONTINUE

Syntax

```
Queue(queuename,options,URL,announceoverride,timeout,AGI,macro,gosub,rule,position)
```

Arguments

- `queuename`
- `options`
    - `C` - Mark all calls as "answered elsewhere" when cancelled.
    - `c` - Continue in the dialplan if the callee hangs up.
    - `d` - data-quality (modem) call (minimum delay).
    - `F` - When the caller hangs up, transfer the **called member** to the specified destination and **start** execution at that location.
        - `context`
        - `exten`
        - `priority`
    - `F` - When the caller hangs up, transfer the **called member** to the next priority of the current extension and **start** execution at that location.
    - `h` - Allow **callee** to hang up by pressing `*`.
    - `H` - Allow **caller** to hang up by pressing `*`.
    - `n` - No retries on the timeout; will exit this application and go to the next step.
    - `i` - Ignore call forward requests from queue members and do nothing when they are requested.
    - `I` - Asterisk will ignore any connected line update requests or any redirecting party update requests it may receive on this dial attempt.
    - `r` - Ring instead of playing MOH. Periodic Announcements are still made, if applicable.
    - `R` - Ring instead of playing MOH when a member channel is actually ringing.
    - `t` - Allow the **called** user to transfer the calling user.
    - `T` - Allow the **calling** user to transfer the call.
    - `w` - Allow the **called** user to write the conversation to disk via Monitor.
    - `W` - Allow the **calling** user to write the conversation to disk via Monitor.
    - `k` - Allow the **called** party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
    - `K` - Allow the **calling** party to enable parking of the call by sending the DTMF sequence defined for call parking in `features.conf`.
    - `x` - Allow the **called** user to write the conversation to disk via MixMonitor.

    - `X` - Allow the **calling** user to write the conversation to disk via MixMonitor.
- `URL` - URL will be sent to the called party if the channel supports it.
- `announceoverride`
- `timeout` - Will cause the queue to fail out after a specified number of seconds, checked between each `queues.conf` *timeout* and *retry* cycle.
- `AGI` - Will setup an AGI script to be executed on the calling party's channel once they are connected to a queue member.
- `macro` - Will run a macro on the calling party's channel once they are connected to a queue member.
- `gosub` - Will run a gosub on the calling party's channel once they are connected to a queue member.
- `rule` - Will cause the queue's defaultrule to be overridden by the rule specified.
- `position` - Attempt to enter the caller into the queue at the numerical position specified. `1` would attempt to enter the caller at the head of the queue, and `3` would attempt to place the caller third in the queue.

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_QueueLog

QueueLog()

Synopsis

Writes to the queue_log file.

Description

Allows you to write your own events into the queue log.

Example: QueueLog(101,${UNIQUEID},${AGENT},WENTONBREAK,600)

Syntax

```
QueueLog(queuename,uniqueid,agent,event,additionalinfo)
```

Arguments

- `queuename`
- `uniqueid`
- `agent`
- `event`
- `additionalinfo`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_RaiseException

RaiseException()

Synopsis

Handle an exceptional condition.

Description

This application will jump to the e extension in the current context, setting the dialplan function EXCEPTION(). If the e extension does not exist, the call will hangup.

Syntax

```
RaiseException(reason)
```

Arguments

- `reason`

See Also

- Asterisk 11 Function_Exception

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Read

Read()

Synopsis

Read a variable.

Description

Reads a #-terminated string of digits a certain number of times from the user in to the given *variable*.

This application sets the following channel variable upon completion:

- `READSTATUS` - This is the status of the read operation.
    - OK
    - ERROR
    - HANGUP
    - INTERRUPTED
    - SKIPPED
    - TIMEOUT

Syntax

```
Read(variablefilename&filename2[&...],maxdigits,options,attempts,timeout)
```

Arguments

- `variable` - The input digits will be stored in the given *variable* name.
- `filenames`
    - `filename` - file(s) to play before reading digits or tone with option i
    - `filename2`
- `maxdigits` - Maximum acceptable number of digits. Stops reading after *maxdigits* have been entered (without requiring the user to press the `#` key).
  Defaults to `0` - no limit - wait for the user press the `#` key. Any value below `0` means the same. Max accepted value is `255`.
- `options`
    - `s` - to return immediately if the line is not up.
    - `i` - to play filename as an indication tone from your `indications.conf`.
    - `n` - to read digits even if the line is not up.
- `attempts` - If greater than `1`, that many *attempts* will be made in the event no data is entered.
- `timeout` - The number of seconds to wait for a digit response. If greater than `0`, that value will override the default timeout. Can be floating point.

See Also

- Asterisk 11 Application_SendDTMF

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ReadExten

ReadExten()

Synopsis

Read an extension into a variable.

Description

Reads a # terminated string of digits from the user into the given variable.

Will set READEXTENSTATUS on exit with one of the following statuses:

- `READEXTENSTATUS`
  - OK - A valid extension exists in ${variable}.
  - TIMEOUT - No extension was entered in the specified time. Also sets ${variable} to "t".
  - INVALID - An invalid extension, ${INVALID_EXTEN}, was entered. Also sets ${variable} to "i".
  - SKIP - Line was not up and the option 's' was specified.
  - ERROR - Invalid arguments were passed.

Syntax

```
ReadExten(variable,filename,context,option,timeout)
```

Arguments

- `variable`
- `filename` - File to play before reading digits or tone with option `i`
- `context` - Context in which to match extensions.
- `option`
  - `s` - Return immediately if the channel is not answered.
  - `i` - Play *filename* as an indication tone from your `indications.conf` or a directly specified list of frequencies and durations.
  - `n` - Read digits even if the channel is not answered.
- `timeout` - An integer number of seconds to wait for a digit response. If greater than `0`, that value will override the default timeout.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_ReadFile

ReadFile()

Synopsis

Read the contents of a text file into a channel variable.

Description

Read the contents of a text file into channel variable *varname*

> **Warning**
> ReadFile has been deprecated in favor of Set(varname=${FILE(file,0,length)})

Syntax

```
ReadFile(varnamefile[length])
```

Arguments

- `varname` - Result stored here.
- `fileparams`
  - `file` - The name of the file to read.
  - `length` - Maximum number of characters to capture.
    If not specified defaults to max.

See Also

- Asterisk 11 Application_System
- Asterisk 11 Application_Read

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ReceiveFAX_app_fax

ReceiveFAX() - [app_fax]

Synopsis

Receive a Fax

Description

Receives a FAX from the channel into the given filename overwriting the file if it already exists.

File created will be in TIFF format.

This application sets the following channel variables:

- `LOCALSTATIONID` - To identify itself to the remote end
- `LOCALHEADERINFO` - To generate a header line on each page
- `FAXSTATUS`
  - SUCCESS
  - FAILED
- `FAXERROR` - Cause of failure
- `REMOTESTATIONID` - The CSID of the remote side
- `FAXPAGES` - Number of pages sent
- `FAXBITRATE` - Transmission rate
- `FAXRESOLUTION` - Resolution of sent fax

Syntax

```
ReceiveFAX(filename[,c])
```

Arguments

- `filename` - Filename of TIFF file save incoming fax
- `c` - Makes the application behave as the calling machine
  (Default behavior is as answering machine)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ReceiveFAX_res_fax

ReceiveFAX() - [res_fax]

Synopsis

Receive a FAX and save as a TIFF/F file.

Description

This application is provided by res_fax, which is a FAX technology agnostic module that utilizes FAX technology resource modules to complete a FAX transmission.

Session arguments can be set by the FAXOPT function and to check results of the ReceiveFax() application.

Syntax

```
ReceiveFAX(filename,options)
```

Arguments

- `filename`
- `options`
    - `d` - Enable FAX debugging.
    - `f` - Allow audio fallback FAX transfer on T.38 capable channels.
    - `F` - Force usage of audio mode on T.38 capable channels.
    - `s` - Send progress Manager events (overrides statusevents setting in res_fax.conf).

See Also

- Asterisk 11 Function_FAXOPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_ReceiveFAX (app_fax)

Moved to Asterisk 11 Application_ReceiveFAX_app_fax.

Asterisk 11 Application_ReceiveFAX (res_fax)

Moved to Asterisk 11 Application_ReceiveFAX_res_fax.

Asterisk 11 Application_Record

Record()

Synopsis

Record to a file.

Description

If filename contains `%d`, these characters will be replaced with a number incremented by one each time the file is recorded. Use `core show file formats` to see the available formats on your system User can press `#` to terminate the recording and continue to the next priority. If the user hangs up during a recording, all data will be lost and the application will terminate.

- `RECORDED_FILE` - Will be set to the final filename of the recording.
- `RECORD_STATUS` - This is the final status of the command
    - DTMF - A terminating DTMF was received ('#' or '*', depending upon option 't')
    - SILENCE - The maximum silence occurred in the recording.
    - SKIP - The line was not yet answered and the 's' option was specified.
    - TIMEOUT - The maximum length was reached.
    - HANGUP - The channel was hung up.
    - ERROR - An unrecoverable error occurred, which resulted in a WARNING to the logs.

Syntax

```
Record(filename.format,silence,maxduration,options)
```

Arguments

- `filename`
    - `filename`
    - `format` - Is the format of the file type to be recorded (wav, gsm, etc).
- `silence` - Is the number of seconds of silence to allow before returning.
- `maxduration` - Is the maximum recording duration in seconds. If missing or 0 there is no maximum.
- `options`
    - `a` - Append to existing recording rather than replacing.
    - `n` - Do not answer, but record anyway if line not yet answered.
    - `q` - quiet (do not play a beep tone).
    - `s` - skip recording if the line is not yet answered.
    - `t` - use alternate '*' terminator key (DTMF) instead of default '#'
    - `x` - Ignore all terminator keys (DTMF) and keep recording until hangup.
    - `k` - Keep recorded file upon hangup.
    - `y` - Terminate recording if **any** DTMF digit is received.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_RemoveQueueMember

RemoveQueueMember()

Synopsis

Dynamically removes queue members.

Description

If the interface is **NOT** in the queue it will return an error.

This application sets the following channel variable upon completion:

- `RQMSTATUS`
    - REMOVED
    - NOTINQUEUE
    - NOSUCHQUEUE
    - NOTDYNAMIC
      Example: RemoveQueueMember(techsupport,SIP/3000)

Syntax

```
RemoveQueueMember(queuename,interface)
```

Arguments

- `queuename`
- `interface`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_ResetCDR

ResetCDR()

Synopsis

Resets the Call Data Record.

Description

This application causes the Call Data Record to be reset.

Syntax

```
ResetCDR(options)
```

Arguments

- `options`
    - `w` - Store the current CDR record before resetting it.
    - `a` - Store any stacked records.
    - `v` - Save CDR variables.
    - `e` - Enable CDR only (negate effects of NoCDR).

See Also

- Asterisk 11 Application_ForkCDR
- Asterisk 11 Application_NoCDR

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_RetryDial

RetryDial()

Synopsis

Place a call, retrying on failure allowing an optional exit extension.

Description

This application will attempt to place a call using the normal Dial application. If no channel can be reached, the *announce* file will be played. Then, it will wait *sleep* number of seconds before retrying the call. After *retries* number of attempts, the calling channel will continue at the next priority in the dialplan. If the *retries* setting is set to 0, this application will retry endlessly. While waiting to retry a call, a 1 digit extension may be dialed. If that extension exists in either the context defined in `EXITCONTEXT` or the current one, The call will jump to that extension immediately. The *dialargs* are specified in the same format that arguments are provided to the Dial application.

Syntax

```
RetryDial(announce,sleep,retries,dialargs)
```

Arguments

- `announce` - Filename of sound that will be played when no channel can be reached
- `sleep` - Number of seconds to wait after a dial attempt failed before a new attempt is made
- `retries` - Number of retries
  When this is reached flow will continue at the next priority in the dialplan
- `dialargs` - Same format as arguments provided to the Dial application

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Return

Return()

Synopsis

Return from gosub routine.

Description

Jumps to the last label on the stack, removing it. The return *value*, if any, is saved in the channel variable `GOSUB_RETVAL`.

Syntax

```
Return(value)
```

Arguments

- `value` - Return value.

See Also

- Asterisk 11 Application_Gosub
- Asterisk 11 Application_StackPop

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Ringing

Ringing()

Synopsis

Indicate ringing tone.

Description

This application will request that the channel indicate a ringing tone to the user.

Syntax

```
Ringing()
```

Arguments

See Also

- Asterisk 11 Application_Busy
- Asterisk 11 Application_Congestion
- Asterisk 11 Application_Progress
- Asterisk 11 Application_Playtones

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SayAlpha

SayAlpha()

Synopsis

Say Alpha.

Description

This application will play the sounds that correspond to the letters of the given *string*.

Syntax

```
SayAlpha(string)
```

Arguments

- `string`

See Also

- Asterisk 11 Application_SayDigits
- Asterisk 11 Application_SayNumber
- Asterisk 11 Application_SayPhonetic
- Asterisk 11 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SayCountedAdj

SayCountedAdj()

Synopsis

Say a adjective in declined form in order to count things

Description

Selects and plays the proper form of an adjective according to the gender and of the noun which it modifies and the number of objects named by the noun-verb combination which have been counted. Used when saying things such as "5 new messages". The various singular and plural forms of the adjective are selected by adding suffixes to *filename*.

If the channel language is English, then no suffix will ever be added (since, in English, adjectives are not declined). If the channel language is Russian or some other slavic language, then the suffix will the specified *gender* for nominative, and "x" for genative plural. (The genative singular is not used when counting things.) For example, SayCountedAdj(1,new,f) will play sound file "newa" (containing the word "novaya"), but SayCountedAdj(5,new,f) will play sound file "newx" (containing the word "novikh").

This application does not automatically answer and should be preceeded by an application such as Answer(), Progress(), or Proceeding().

Syntax

```
SayCountedAdj(number,filename,gender)
```

Arguments

- `number` - The number of things
- `filename` - File name stem for the adjective
- `gender` - The gender of the noun modified, one of 'm', 'f', 'n', or 'c'

See Also

- Asterisk 11 Application_SayCountedNoun
- Asterisk 11 Application_SayNumber

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_SayCountedNoun

SayCountedNoun()

Synopsis

Say a noun in declined form in order to count things

Description

Selects and plays the proper singular or plural form of a noun when saying things such as "five calls". English has simple rules for deciding when to say "call" and when to say "calls", but other languages have complicated rules which would be extremely difficult to implement in the Asterisk dialplan language.

The correct sound file is selected by examining the *number* and adding the appropriate suffix to *filename*. If the channel language is English, then the suffix will be either empty or "s". If the channel language is Russian or some other Slavic language, then the suffix will be empty for nominative, "x1" for genative singular, and "x2" for genative plural.

Note that combining *filename* with a suffix will not necessarily produce a correctly spelled plural form. For example, SayCountedNoun(2,man) will play the sound file "mans" rather than "men". This behavior is intentional. Since the file name is never seen by the end user, there is no need to implement complicated spelling rules. We simply record the word "men" in the sound file named "mans".

This application does not automatically answer and should be preceeded by an application such as Answer() or Progress.

Syntax

```
SayCountedNoun(number,filename)
```

Arguments

- `number` - The number of things
- `filename` - File name stem for the noun that is the the name of the things

See Also

- Asterisk 11 Application_SayCountedAdj
- Asterisk 11 Application_SayNumber

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_SayCountPL

SayCountPL()

Synopsis

Say Polish counting words.

Description

Polish grammar has some funny rules for counting words. for example 1 zloty, 2 zlote, 5 zlotych. This application will take the words for 1, 2-4 and 5 and decide based on grammar rules which one to use with the number you pass to it.

Example: SayCountPL(zloty,zlote,zlotych,122) will give: zlote

Syntax

```
SayCountPL(word1,word2,word5,number)
```

Arguments

- `word1`
- `word2`
- `word5`
- `number`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SayDigits

SayDigits()

Synopsis

Say Digits.

Description

This application will play the sounds that correspond to the digits of the given number. This will use the language that is currently set for the channel.

Syntax

```
SayDigits(digits)
```

Arguments

- `digits`

See Also

- Asterisk 11 Application_SayAlpha
- Asterisk 11 Application_SayNumber
- Asterisk 11 Application_SayPhonetic
- Asterisk 11 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SayNumber

SayNumber()

Synopsis

Say Number.

Description

This application will play the sounds that correspond to the given *digits*. Optionally, a *gender* may be specified. This will use the language that is currently set for the channel. See the CHANNEL() function for more information on setting the language for the channel.

Syntax

```
SayNumber(digits,gender)
```

Arguments

- `digits`
- `gender`

See Also

- Asterisk 11 Application_SayAlpha
- Asterisk 11 Application_SayDigits
- Asterisk 11 Application_SayPhonetic
- Asterisk 11 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SayPhonetic

SayPhonetic()

Synopsis

Say Phonetic.

Description

This application will play the sounds from the phonetic alphabet that correspond to the letters in the given *string*.

Syntax

```
SayPhonetic(string)
```

Arguments

- `string`

See Also

- Asterisk 11 Application_SayAlpha
- Asterisk 11 Application_SayDigits
- Asterisk 11 Application_SayNumber

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SayUnixTime

SayUnixTime()

Synopsis

Says a specified time in a custom format.

Description

Uses some of the sound files stored in `/var/lib/asterisk/sounds` to construct a phrase saying the specified date and/or time in the specified format.

Syntax

```
SayUnixTime([unixtime[,timezone[,format[,options]]]])
```

Arguments

- `unixtime` - time, in seconds since Jan 1, 1970. May be negative. Defaults to now.
- `timezone` - timezone, see `/usr/share/zoneinfo` for a list. Defaults to machine default.
- `format` - a format the time is to be said in. See `voicemail.conf`. Defaults to `ABdY "digits/at" IMp`
- `options`
    - `j` - Allow the calling user to dial digits to jump to that extension.

See Also

- Asterisk 11 Function_STRFTIME
- Asterisk 11 Function_STRPTIME
- Asterisk 11 Function_IFTIME

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_SendDTMF

SendDTMF()

Synopsis

Sends arbitrary DTMF digits

Description

It will send all digits or terminate if it encounters an error.

Syntax

```
SendDTMF(digits[,timeout_ms[,duration_ms[,channel]]])
```

Arguments

- `digits` - List of digits 0-9,*#,a-d,A-D to send also w for a half second pause, and f or F for a flash-hook if the channel supports flash-hook.
- `timeout_ms` - Amount of time to wait in ms between tones. (defaults to .25s)
- `duration_ms` - Duration of each digit
- `channel` - Channel where digits will be played

See Also

- Asterisk 11 Application_Read

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SendFAX_app_fax

SendFAX() - [app_fax]

Synopsis

Send a Fax

Description

Send a given TIFF file to the channel as a FAX.

This application sets the following channel variables:

- `LOCALSTATIONID` - To identify itself to the remote end
- `LOCALHEADERINFO` - To generate a header line on each page
- `FAXSTATUS`
    - SUCCESS
    - FAILED
- `FAXERROR` - Cause of failure
- `REMOTESTATIONID` - The CSID of the remote side
- `FAXPAGES` - Number of pages sent
- `FAXBITRATE` - Transmission rate
- `FAXRESOLUTION` - Resolution of sent fax

Syntax

```
SendFAX(filename[,a])
```

Arguments

- `filename` - Filename of TIFF file to fax
- a - Makes the application behave as the answering machine
  (Default behavior is as calling machine)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SendFAX_res_fax

SendFAX() - [res_fax]

Synopsis

Sends a specified TIFF/F file as a FAX.

Description

This application is provided by res_fax, which is a FAX technology agnostic module that utilizes FAX technology resource modules to complete a FAX transmission.

Session arguments can be set by the FAXOPT function and to check results of the SendFax() application.

Syntax

```
SendFAX(filename2[&...],options)
```

Arguments

- `filename`
  - `filename2` - TIFF file to send as a FAX.
- `options`
  - `d` - Enable FAX debugging.
  - `f` - Allow audio fallback FAX transfer on T.38 capable channels.
  - `F` - Force usage of audio mode on T.38 capable channels.
  - `s` - Send progress Manager events (overrides statusevents setting in res_fax.conf).
  - `z` - Initiate a T.38 reinvite on the channel if the remote end does not.

See Also

- Asterisk 11 Function_FAXOPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SendFAX (app_fax)

Moved to Asterisk 11 Application_SendFAX_app_fax.

Asterisk 11 Application_SendFAX (res_fax)

Moved to Asterisk 11 Application_SendFAX_res_fax.

Asterisk 11 Application_SendImage

SendImage()

Synopsis

Sends an image file.

Description

Send an image file on a channel supporting it.

Result of transmission will be stored in `SENDIMAGESTATUS`

- `SENDIMAGESTATUS`
  - SUCCESS - Transmission succeeded.
  - FAILURE - Transmission failed.
  - UNSUPPORTED - Image transmission not supported by channel.

Syntax

```
SendImage(filename)
```

Arguments

- `filename` - Path of the filename (image) to send.

See Also

- Asterisk 11 Application_SendText
- Asterisk 11 Application_SendURL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SendText

SendText()

Synopsis

Send a Text Message.

Description

Sends *text* to current channel (callee).

Result of transmission will be stored in the `SENDTEXTSTATUS`

- `SENDTEXTSTATUS`
  - SUCCESS - Transmission succeeded.
  - FAILURE - Transmission failed.
  - UNSUPPORTED - Text transmission not supported by channel.

    > **Note**
    > At this moment, text is supposed to be 7 bit ASCII in most channels.

Syntax

```
SendText(text)
```

Arguments

- `text`

See Also

- Asterisk 11 Application_SendImage
- Asterisk 11 Application_SendURL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SendURL

SendURL()

Synopsis

Send a URL.

Description

Requests client go to *URL* (IAX2) or sends the URL to the client (other channels).

Result is returned in the `SENDURLSTATUS` channel variable:

- `SENDURLSTATUS`
  - SUCCESS - URL successfully sent to client.
  - FAILURE - Failed to send URL.
  - NOLOAD - Client failed to load URL (wait enabled).
  - UNSUPPORTED - Channel does not support URL transport.
    SendURL continues normally if the URL was sent correctly or if the channel does not support HTML transport. Otherwise, the channel is hung up.

Syntax

```
SendURL(URL,option)
```

Arguments

- `URL`
- `option`
  - `w` - Execution will wait for an acknowledgement that the URL has been loaded before continuing.

See Also

- Asterisk 11 Application_SendImage
- Asterisk 11 Application_SendText

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Set

Set()

Synopsis

Set channel variable or function value.

Description

This function can be used to set the value of channel variables or dialplan functions. When setting variables, if the variable name is prefixed with _, the variable will be inherited into channels created from the current channel. If the variable name is prefixed with __, the variable will be inherited into channels created from the current channel and all children channels.

> **Note**
> If (and only if), in `/etc/asterisk/asterisk.conf`, you have a `[compat]` category, and you have `app_set = 1.4` under that, then the behavior of this app changes, and strips surrounding quotes from the right hand side as it did previously in 1.4. The advantages of not stripping out quoting, and not caring about the separator characters (comma and vertical bar) were sufficient to make these changes in 1.6. Confusion about how many backslashes would be needed to properly protect separators and quotes in various database access strings has been greatly reduced by these changes.

Syntax

```
Set(name=value)
```

Arguments

- `name`
- `value`

See Also

- Asterisk 11 Application_MSet
- Asterisk 11 Function_GLOBAL
- Asterisk 11 Function_SET
- Asterisk 11 Function_ENV

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_SetAMAFlags

SetAMAFlags()

Synopsis

Set the AMA Flags.

Description

This application will set the channel's AMA Flags for billing purposes.

Syntax

```
SetAMAFlags(flag)
```

Arguments

- `flag`

See Also

- Asterisk 11 Function_CDR

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SetCallerPres

SetCallerPres()

Synopsis

Set CallerID Presentation.

Description

Set Caller*ID presentation on a call.

Syntax

```
SetCallerPres(presentation)
```

Arguments

- `presentation`
    - `allowed_not_screened` - Presentation Allowed, Not Screened.
    - `allowed_passed_screen` - Presentation Allowed, Passed Screen.
    - `allowed_failed_screen` - Presentation Allowed, Failed Screen.
    - `allowed` - Presentation Allowed, Network Number.
    - `prohib_not_screened` - Presentation Prohibited, Not Screened.
    - `prohib_passed_screen` - Presentation Prohibited, Passed Screen.
    - `prohib_failed_screen` - Presentation Prohibited, Failed Screen.
    - `prohib` - Presentation Prohibited, Network Number.
    - `unavailable` - Number Unavailable.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SetMusicOnHold

SetMusicOnHold()

Synopsis

Set default Music On Hold class.

Description

!!! DEPRECATED. USe Set(CHANNEL(musicclass)=...) instead !!!

Sets the default class for music on hold for a given channel. When music on hold is activated, this class will be used to select which music is played.

!!! DEPRECATED. USe Set(CHANNEL(musicclass)=...) instead !!!

Syntax

```
SetMusicOnHold(class)
```

Arguments

- `class`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SIPAddHeader

SIPAddHeader()

Synopsis

Add a SIP header to the outbound call.

Description

Adds a header to a SIP call placed with DIAL.

Remember to use the X-header if you are adding non-standard SIP headers, like `X-Asterisk-Accountcode:`. Use this with care. Adding the wrong headers may jeopardize the SIP dialog.

Always returns `0`.

Syntax

```
SIPAddHeader(Header:Content)
```

Arguments

- `Header`
- `Content`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SIPDtmfMode

SIPDtmfMode()

Synopsis

Change the dtmfmode for a SIP call.

Description

Changes the dtmfmode for a SIP call.

Syntax

```
SIPDtmfMode(mode)
```

Arguments

- `mode`
    - `inband`
    - `info`
    - `rfc2833`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SIPRemoveHeader

SIPRemoveHeader()

Synopsis

Remove SIP headers previously added with SIPAddHeader

Description

SIPRemoveHeader() allows you to remove headers which were previously added with SIPAddHeader(). If no parameter is supplied, all previously added headers will be removed. If a parameter is supplied, only the matching headers will be removed.

For example you have added these 2 headers:

SIPAddHeader(P-Asserted-Identity: sip:foo@bar);

SIPAddHeader(P-Preferred-Identity: sip:bar@foo);

// remove all headers

SIPRemoveHeader();

// remove all P- headers

SIPRemoveHeader(P-);

// remove only the PAI header (note the : at the end)

SIPRemoveHeader(P-Asserted-Identity;

Always returns `0`.

Syntax

```
SIPRemoveHeader([Header])
```

Arguments

- `Header`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SIPSendCustomINFO

SIPSendCustomINFO()

Synopsis

Send a custom INFO frame on specified channels.

Description

SIPSendCustomINFO() allows you to send a custom INFO message on all active SIP channels or on channels with the specified User Agent. This application is only available if TEST_FRAMEWORK is defined.

Syntax

```
SIPSendCustomINFO(Data[,UserAgent])
```

Arguments

- `Data`
- `UserAgent`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SkelGuessNumber

SkelGuessNumber()

Synopsis

An example number guessing game

Description

This simple number guessing application is a template to build other applications from. It shows you the basic structure to create your own Asterisk applications.

Syntax

```
SkelGuessNumber(level,options)
```

Arguments

- `level`
- `options`
    - c - The computer should cheat
    - n - How many games to play before hanging up

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SLAStation

SLAStation()

Synopsis

Shared Line Appearance Station.

Description

This application should be executed by an SLA station. The argument depends on how the call was initiated. If the phone was just taken off hook, then the argument *station* should be just the station name. If the call was initiated by pressing a line key, then the station name should be preceded by an underscore and the trunk name associated with that line button.

For example: `station1_line1`

On exit, this application will set the variable `SLASTATION_STATUS` to one of the following values:

- `SLASTATION_STATUS`
  - FAILURE
  - CONGESTION
  - SUCCESS

Syntax

```
SLAStation(station)
```

Arguments

- `station` - Station name

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SLATrunk

SLATrunk()

Synopsis

Shared Line Appearance Trunk.

Description

This application should be executed by an SLA trunk on an inbound call. The channel calling this application should correspond to the SLA trunk with the name *trunk* that is being passed as an argument.

On exit, this application will set the variable `SLATRUNK_STATUS` to one of the following values:

- `SLATRUNK_STATUS`
    - FAILURE
    - SUCCESS
    - UNANSWERED
    - RINGTIMEOUT

Syntax

```
SLATrunk(trunk,options)
```

Arguments

- `trunk` - Trunk name
- `options`
    - `M` - Play back the specified MOH *class* instead of ringing
        - `class`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SMS

SMS()

Synopsis

Communicates with SMS service centres and SMS capable analogue phones.

Description

SMS handles exchange of SMS data with a call to/from SMS capable phone or SMS PSTN service center. Can send and/or receive SMS messages. Works to ETSI ES 201 912; compatible with BT SMS PSTN service in UK and Telecom Italia in Italy.

Typical usage is to use to handle calls from the SMS service centre CLI, or to set up a call using `outgoing` or manager interface to connect service centre to SMS().

"Messages are processed as per text file message queues. smsq (a separate software) is a command to generate message queues and send messages.

> **Note**
> The protocol has tight delay bounds. Please use short frames and disable/keep short the jitter buffer on the ATA to make sure that respones (ACK etc.) are received in time.

Syntax

```
SMS(name,options,addr,body)
```

Arguments

- `name` - The name of the queue used in `/var/spool/asterisk/sms`
- `options`
  - `a` - Answer, i.e. send initial FSK packet.
  - `s` - Act as service centre talking to a phone.
  - `t` - Use protocol 2 (default used is protocol 1).
  - `p` - Set the initial delay to N ms (default is `300`). addr and body are a deprecated format to send messages out.
  - `r` - Set the Status Report Request (SRR) bit.
  - `o` - The body should be coded as octets not 7-bit symbols.
- `addr`
- `body`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_SoftHangup

SoftHangup()

Synopsis

Hangs up the requested channel.

Description

Hangs up the requested channel. If there are no channels to hangup, the application will report it.

Syntax

```
SoftHangup(Technology/Resource,options)
```

Arguments

- `Technology/Resource`
- `options`
    - a - Hang up all channels on a specified device instead of a single resource

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechActivateGrammar

SpeechActivateGrammar()

Synopsis

Activate a grammar.

Description

This activates the specified grammar to be recognized by the engine. A grammar tells the speech recognition engine what to recognize, and how to portray it back to you in the dialplan. The grammar name is the only argument to this application.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechActivateGrammar(grammar_name)
```

Arguments

- `grammar_name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechBackground

SpeechBackground()

Synopsis

Play a sound file and wait for speech to be recognized.

Description

This application plays a sound file and waits for the person to speak. Once they start speaking playback of the file stops, and silence is heard. Once they stop talking the processing sound is played to indicate the speech recognition engine is working. Once results are available the application returns and results (score and text) are available using dialplan functions.

The first text and score are ${SPEECH_TEXT(0)} AND ${SPEECH_SCORE(0)} while the second are ${SPEECH_TEXT(1)} and ${SPEECH_SCORE(1)}.

The first argument is the sound file and the second is the timeout integer in seconds.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechBackground(sound_file,timeout,options)
```

Arguments

- `sound_file`
- `timeout` - Timeout integer in seconds. Note the timeout will only start once the sound file has stopped playing.
- `options`
    - n - Don't answer the channel if it has not already been answered.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechCreate

SpeechCreate()

Synopsis

Create a Speech Structure.

Description

This application creates information to be used by all the other applications. It must be called before doing any speech recognition activities such as activating a grammar. It takes the engine name to use as the argument, if not specified the default engine will be used.

Sets the ERROR channel variable to 1 if the engine cannot be used.

Syntax

```
SpeechCreate(engine_name)
```

Arguments

- `engine_name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechDeactivateGrammar

SpeechDeactivateGrammar()

Synopsis

Deactivate a grammar.

Description

This deactivates the specified grammar so that it is no longer recognized.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechDeactivateGrammar(grammar_name)
```

Arguments

- `grammar_name` - The grammar name to deactivate

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechDestroy

SpeechDestroy()

Synopsis

End speech recognition.

Description

This destroys the information used by all the other speech recognition applications. If you call this application but end up wanting to recognize more speech, you must call SpeechCreate() again before calling any other application.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechDestroy()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechLoadGrammar

SpeechLoadGrammar()

Synopsis

Load a grammar.

Description

Load a grammar only on the channel, not globally.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechLoadGrammar(grammar_name,path)
```

Arguments

- `grammar_name`
- `path`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechProcessingSound

SpeechProcessingSound()

Synopsis

Change background processing sound.

Description

This changes the processing sound that SpeechBackground plays back when the speech recognition engine is processing and working to get results.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechProcessingSound(sound_file)
```

Arguments

- `sound_file`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechStart

SpeechStart()

Synopsis

Start recognizing voice in the audio stream.

Description

Tell the speech recognition engine that it should start trying to get results from audio being fed to it.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechStart()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_SpeechUnloadGrammar

SpeechUnloadGrammar()

Synopsis

Unload a grammar.

Description

Unload a grammar.

Hangs up the channel on failure. If this is not desired, use TryExec.

Syntax

```
SpeechUnloadGrammar(grammar_name)
```

Arguments

- `grammar_name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_StackPop

StackPop()

Synopsis

Remove one address from gosub stack.

Description

Removes last label on the stack, discarding it.

Syntax

```
StackPop()
```

Arguments

See Also

- Asterisk 11 Application_Return
- Asterisk 11 Application_Gosub

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_StartMusicOnHold

StartMusicOnHold()

Synopsis

Play Music On Hold.

Description

Starts playing music on hold, uses default music class for channel. Starts playing music specified by class. If omitted, the default music source for the channel will be used. Always returns `0`.

Syntax

```
StartMusicOnHold(class)
```

Arguments

- `class`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_StopMixMonitor

StopMixMonitor()

Synopsis

Stop recording a call through MixMonitor, and free the recording's file handle.

Description

Stops the audio recording that was started with a call to `MixMonitor()` on the current channel.

Syntax

```
StopMixMonitor([MixMonitorID])
```

Arguments

- `MixMonitorID` - If a valid ID is provided, then this command will stop only that specific MixMonitor.

See Also

- Asterisk 11 Application_MixMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_StopMonitor

StopMonitor()

Synopsis

Stop monitoring a channel.

Description

Stops monitoring a channel. Has no effect if the channel is not monitored.

Syntax

```
StopMonitor()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_StopMusicOnHold

StopMusicOnHold()

Synopsis

Stop playing Music On Hold.

Description

Stops playing music on hold.

Syntax

```
StopMusicOnHold()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_StopPlayTones

StopPlayTones()

Synopsis

Stop playing a tone list.

Description

Stop playing a tone list, initiated by PlayTones().

Syntax

```
StopPlayTones()
```

Arguments

See Also

- Asterisk 11 Application_PlayTones

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_System

System()

Synopsis

Execute a system command.

Description

Executes a command by using system(). If the command fails, the console should report a fallthrough.

Result of execution is returned in the `SYSTEMSTATUS` channel variable:

- `SYSTEMSTATUS`
    - FAILURE - Could not execute the specified command.
    - SUCCESS - Specified command successfully executed.

Syntax

```
System(command)
```

Arguments

- `command` - Command to execute

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_TestClient

TestClient()

Synopsis

Execute Interface Test Client.

Description

Executes test client with given *testid*. Results stored in `/var/log/asterisk/testreports/<testid>-client.txt`

Syntax

```
TestClient(testid)
```

Arguments

- `testid` - An ID to identify this test.

See Also

- Asterisk 11 Application_TestServer

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_TestServer

TestServer()

Synopsis

Execute Interface Test Server.

Description

Perform test server function and write call report. Results stored in `/var/log/asterisk/testreports/<testid>-server.txt`

Syntax

```
TestServer()
```

Arguments

See Also

- Asterisk 11 Application_TestClient

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Transfer

Transfer()

Synopsis

Transfer caller to remote extension.

Description

Requests the remote caller be transferred to a given destination. If TECH (SIP, IAX2, LOCAL etc) is used, only an incoming call with the same channel technology will be transfered. Note that for SIP, if you transfer before call is setup, a 302 redirect SIP message will be returned to the caller.

The result of the application will be reported in the `TRANSFERSTATUS` channel variable:

- `TRANSFERSTATUS`
    - SUCCESS - Transfer succeeded.
    - FAILURE - Transfer failed.
    - UNSUPPORTED - Transfer unsupported by channel driver.

Syntax

```
Transfer(Tech/destination)
```

Arguments

- `dest`
    - `Tech/`
    - `destination`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_TryExec

TryExec()

Synopsis

Executes dialplan application, always returning.

Description

Allows an arbitrary application to be invoked even when not hard coded into the dialplan. To invoke external applications see the application System. Always returns to the dialplan. The channel variable TRYSTATUS will be set to one of:

- `TRYSTATUS`
    - SUCCESS - If the application returned zero.
    - FAILED - If the application returned non-zero.
    - NOAPP - If the application was not found or was not specified.

Syntax

```
TryExec(arguments)
```

Arguments

- `appname`
    - `arguments`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_TrySystem

TrySystem()

Synopsis

Try executing a system command.

Description

Executes a command by using system().

Result of execution is returned in the `SYSTEMSTATUS` channel variable:

- `SYSTEMSTATUS`
  - FAILURE - Could not execute the specified command.
  - SUCCESS - Specified command successfully executed.
  - APPERROR - Specified command successfully executed, but returned error code.

Syntax

```
TrySystem(command)
```

Arguments

- `command` - Command to execute

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_UnpauseMonitor

UnpauseMonitor()

Synopsis

Unpause monitoring of a channel.

Description

Unpauses monitoring of a channel on which monitoring had previously been paused with PauseMonitor.

Syntax

```
UnpauseMonitor()
```

Arguments

See Also

- Asterisk 11 Application_PauseMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_UnpauseQueueMember

UnpauseQueueMember()

Synopsis

Unpauses a queue member.

Description

Unpauses (resumes calls to) a queue member. This is the counterpart to `PauseQueueMember()` and operates exactly the same way, except it unpauses instead of pausing the given interface.

This application sets the following channel variable upon completion:

- `UPQMSTATUS` - The status of the attempt to unpause a queue member as a text string.
    - UNPAUSED
    - NOTFOUND
      Example: UnpauseQueueMember(,SIP/3000)

Syntax

```
UnpauseQueueMember(queuename,interface,options,reason)
```

Arguments

- `queuename`
- `interface`
- `options`
- `reason` - Is used to add extra information to the appropriate queue_log entries and manager events.

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_UserEvent

UserEvent()

Synopsis

Send an arbitrary event to the manager interface.

Description

Sends an arbitrary event to the manager interface, with an optional *body* representing additional arguments. The *body* may be specified as a `,` delimited list of headers. Each additional argument will be placed on a new line in the event. The format of the event will be:

Event: UserEvent

UserEvent: <specified event name>

[body]

If no *body* is specified, only Event and UserEvent headers will be present.

Syntax

```
UserEvent(eventname,body)
```

Arguments

- `eventname`
- `body`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Application_Verbose

Verbose()

Synopsis

Send arbitrary text to verbose output.

Description

Sends an arbitrary text message to verbose output.

Syntax

```
Verbose(level,message)
```

Arguments

- `level` - Must be an integer value. If not specified, defaults to 0.
- `message` - Output text message.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_VMAuthenticate

VMAuthenticate()

Synopsis

Authenticate with Voicemail passwords.

Description

This application behaves the same way as the Authenticate application, but the passwords are taken from `voicemail.conf`. If the *mailbox* is specified, only that mailbox's password will be considered valid. If the *mailbox* is not specified, the channel variable `AUTH_MAILBOX` will be set with the authenticated mailbox.

The VMAuthenticate application will exit if the following DTMF digit is entered as Mailbox or Password, and the extension exists:

- `*` - Jump to the a extension in the current dialplan context.

Syntax

```
VMAuthenticate(mailbox@context,options)
```

Arguments

- `mailbox`
  - `mailbox`
  - `context`
- `options`
  - `s` - Skip playing the initial prompts.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_VMSayName

VMSayName()

Synopsis

Play the name of a voicemail user

Description

This application will say the recorded name of the voicemail user specified as the argument to this application. If no context is provided, `default` is assumed.

Syntax

```
VMSayName(mailbox@context)
```

Arguments

- `mailbox`
    - `mailbox`
    - `context`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_VoiceMail

VoiceMail()

Synopsis

Leave a Voicemail message.

Description

This application allows the calling party to leave a message for the specified list of mailboxes. When multiple mailboxes are specified, the greeting will be taken from the first mailbox specified. Dialplan execution will stop if the specified mailbox does not exist.

The Voicemail application will exit if any of the following DTMF digits are received:

- `0` - Jump to the o extension in the current dialplan context.
- `*` - Jump to the a extension in the current dialplan context.
  This application will set the following channel variable upon completion:

- `VMSTATUS` - This indicates the status of the execution of the VoiceMail application.
  - SUCCESS
  - USEREXIT
  - FAILED

Syntax

```
VoiceMail(mailbox1&mailbox2[&...],options)
```

Arguments

- `mailboxs`
  - `mailbox1`
    - `mailbox`
    - `context`
  - `mailbox2`
    - `mailbox`
    - `context`
- `options`
  - `b` - Play the `busy` greeting to the calling party.
  - `d` - Accept digits for a new extension in context *c*, if played during the greeting. Context defaults to the current context.
    - `c`
  - `g` - Use the specified amount of gain when recording the voicemail message. The units are whole-number decibels (dB). Only works on supported technologies, which is DAHDI only.
    - `#`
  - `s` - Skip the playback of instructions for leaving a message to the calling party.
  - `u` - Play the `unavailable` greeting.
  - `U` - Mark message as `URGENT`.
  - `P` - Mark message as `PRIORITY`.

See Also

- Asterisk 11 Application_VoiceMailMain

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_VoiceMailMain

VoiceMailMain()

Synopsis

Check Voicemail messages.

Description

This application allows the calling party to check voicemail messages. A specific *mailbox*, and optional corresponding *context*, may be specified. If a *mailbox* is not provided, the calling party will be prompted to enter one. If a *context* is not specified, the `default` context will be used.

The VoiceMailMain application will exit if the following DTMF digit is entered as Mailbox or Password, and the extension exists:

- `*` - Jump to the a extension in the current dialplan context.

Syntax

```
VoiceMailMain(mailbox@context,options)
```

Arguments

- `mailbox`
  - `mailbox`
  - `context`
- `options`
  - `p` - Consider the *mailbox* parameter as a prefix to the mailbox that is entered by the caller.
  - `g` - Use the specified amount of gain when recording a voicemail message. The units are whole-number decibels (dB).
    - `#`
  - `s` - Skip checking the passcode for the mailbox.
  - `a` - Skip folder prompt and go directly to *folder* specified. Defaults to `INBOX` (or `0`).
    - `folder`
    - `0` - INBOX
    - `1` - Old
    - `2` - Work
    - `3` - Family
    - `4` - Friends
    - `5` - Cust1
    - `6` - Cust2
    - `7` - Cust3
    - `8` - Cust4
    - `9` - Cust5

See Also

- Asterisk 11 Application_VoiceMail

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_VoiceMailPlayMsg

VoiceMailPlayMsg()

Synopsis

Play a single voice mail msg from a mailbox by msg id.

Description

This application sets the following channel variable upon completion:

- `VOICEMAIL_PLAYBACKSTATUS` - The status of the playback attempt as a text string.
    - SUCCESS
    - FAILED

Syntax

```
VoiceMailPlayMsg(mailbox@context,msg_id)
```

Arguments

- `mailbox`
    - `mailbox`
    - `context`
- `msg_id` - The msg id of the msg to play back.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Wait

Wait()

Synopsis

Waits for some time.

Description

This application waits for a specified number of *seconds*.

Syntax

```
Wait(seconds)
```

Arguments

- `seconds` - Can be passed with fractions of a second. For example, `1.5` will ask the application to wait for 1.5 seconds.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_WaitExten

WaitExten()

Synopsis

Waits for an extension to be entered.

Description

This application waits for the user to enter a new extension for a specified number of *seconds*.

> **Warning**
> Use of the application `WaitExten` within a macro will not function as expected. Please use the `Read` application in order to read DTMF from a channel currently executing a macro.

Syntax

```
WaitExten(seconds,options)
```

Arguments

- `seconds` - Can be passed with fractions of a second. For example, `1.5` will ask the application to wait for 1.5 seconds.
- `options`
    - `m` - Provide music on hold to the caller while waiting for an extension.
        - `x` - Specify the class for music on hold. **CHANNEL(musicclass) will be used instead if set**

See Also

- Asterisk 11 Application_Background
- Asterisk 11 Function_TIMEOUT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_WaitForNoise

WaitForNoise()

Synopsis

Waits for a specified amount of noise.

Description

Waits for up to *noiserequired* milliseconds of noise, *iterations* times. An optional *timeout* specified the number of seconds to return after, even if we do not receive the specified amount of noise. Use *timeout* with caution, as it may defeat the purpose of this application, which is to wait indefinitely until noise is detected on the line.

Syntax

```
WaitForNoise(noiserequired,iterations,timeout)
```

Arguments

- `noiserequired`
- `iterations` - If not specified, defaults to `1`.
- `timeout` - Is specified only to avoid an infinite loop in cases where silence is never achieved.

See Also

- Asterisk 11 Application_WaitForSilence

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_WaitForRing

WaitForRing()

Synopsis

Wait for Ring Application.

Description

Returns `0` after waiting at least *timeout* seconds, and only after the next ring has completed. Returns `0` on success or `-1` on hangup.

Syntax

```
WaitForRing(timeout)
```

Arguments

- `timeout`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_WaitForSilence

WaitForSilence()

Synopsis

Waits for a specified amount of silence.

Description

Waits for up to *silencerequired* milliseconds of silence, *iterations* times. An optional *timeout* specified the number of seconds to return after, even if we do not receive the specified amount of silence. Use *timeout* with caution, as it may defeat the purpose of this application, which is to wait indefinitely until silence is detected on the line. This is particularly useful for reverse-911-type call broadcast applications where you need to wait for an answering machine to complete its spiel before playing a message.

Typically you will want to include two or more calls to WaitForSilence when dealing with an answering machine; first waiting for the spiel to finish, then waiting for the beep, etc.

Examples:

WaitForSilence(500,2) will wait for 1/2 second of silence, twice

WaitForSilence(1000) will wait for 1 second of silence, once

WaitForSilence(300,3,10) will wait for 300ms silence, 3 times, and returns after 10 sec, even if silence is not detected

Sets the channel variable `WAITSTATUS` to one of these values:

- `WAITSTATUS`
    - SILENCE - if exited with silence detected.
    - TIMEOUT - if exited without silence detected after timeout.

Syntax

```
WaitForSilence(silencerequired,iterations,timeout)
```

Arguments

- `silencerequired`
- `iterations` - If not specified, defaults to `1`.
- `timeout` - Is specified only to avoid an infinite loop in cases where silence is never achieved.

See Also

- Asterisk 11 Application_WaitForNoise

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_WaitMusicOnHold

WaitMusicOnHold()

Synopsis

Wait, playing Music On Hold.

Description

!!! DEPRECATED. Use MusicOnHold instead !!!

Plays hold music specified number of seconds. Returns `0` when done, or `-1` on hangup. If no hold music is available, the delay will still occur with no sound.

!!! DEPRECATED. Use MusicOnHold instead !!!

Syntax

```
WaitMusicOnHold(delay)
```

Arguments

- `delay`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_WaitUntil

WaitUntil()

Synopsis

Wait (sleep) until the current time is the given epoch.

Description

Waits until the given *epoch*.

Sets `WAITUNTILSTATUS` to one of the following values:

- `WAITUNTILSTATUS`
    - OK - Wait succeeded.
    - FAILURE - Invalid argument.
    - HANGUP - Channel hungup before time elapsed.
    - PAST - Time specified had already past.

Syntax

```
WaitUntil(epoch)
```

Arguments

- `epoch`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_While

While()

Synopsis

Start a while loop.

Description

Start a While Loop. Execution will return to this point when `EndWhile()` is called until expr is no longer true.

Syntax

```
While(expr)
```

Arguments

- `expr`

See Also

- Asterisk 11 Application_EndWhile
- Asterisk 11 Application_ExitWhile
- Asterisk 11 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Application_Zapateller

Zapateller()

Synopsis

Block telemarketers with SIT.

Description

Generates special information tone to block telemarketers from calling you.

This application will set the following channel variable upon completion:

- `ZAPATELLERSTATUS` - This will contain the last action accomplished by the Zapateller application. Possible values include:
  - NOTHING
  - ANSWERED
  - ZAPPED

Syntax

```
Zapateller(options)
```

Arguments

- `options` - Comma delimited list of options.
  - `answer` - Causes the line to be answered before playing the tone.
  - `nocallerid` - Causes Zapateller to only play the tone if there is no callerid information available.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Dialplan Functions

Asterisk 11 Function_AES_DECRYPT

AES_DECRYPT()

Synopsis

Decrypt a string encoded in base64 with AES given a 16 character key.

Description

Returns the plain text string.

Syntax

```
AES_DECRYPT(key,string)
```

Arguments

- `key` - AES Key
- `string` - Input string.

See Also

- Asterisk 11 Function_AES_ENCRYPT
- Asterisk 11 Function_BASE64_ENCODE
- Asterisk 11 Function_BASE64_DECODE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_AES_ENCRYPT

AES_ENCRYPT()

Synopsis

Encrypt a string with AES given a 16 character key.

Description

Returns an AES encrypted string encoded in base64.

Syntax

```
AES_ENCRYPT(key,string)
```

Arguments

- `key` - AES Key
- `string` - Input string

See Also

- Asterisk 11 Function_AES_DECRYPT
- Asterisk 11 Function_BASE64_ENCODE
- Asterisk 11 Function_BASE64_DECODE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_AGC

AGC()

Synopsis

Apply automatic gain control to audio on a channel.

Description

The AGC function will apply automatic gain control to the audio on the channel that it is executed on. Using `rx` for audio received and `tx` for audio transmitted to the channel. When using this function you set a target audio level. It is primarily intended for use with analog lines, but could be useful for other channels as well. The target volume is set with a number between `1-32768`. The larger the number the louder (more gain) the channel will receive.

Examples:

exten => 1,1,Set(AGC(rx)=8000)

exten => 1,2,Set(AGC(tx)=off)

Syntax

```
AGC(channeldirection)
```

Arguments

- `channeldirection` - This can be either `rx` or `tx`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_AGENT

AGENT()

Synopsis

Gets information about an Agent

Description

Syntax

```
AGENT(agentid:item)
```

Arguments

- `agentid`
- `item` - The valid items to retrieve are:
    - `status` - (default) The status of the agent (LOGGEDIN | LOGGEDOUT)
    - `password` - The password of the agent
    - `name` - The name of the agent
    - `mohclass` - MusicOnHold class
    - `channel` - The name of the active channel for the Agent (AgentLogin)
    - `fullchannel` - The untruncated name of the active channel for the Agent (AgentLogin)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_AMI_CLIENT

AMI_CLIENT()

Synopsis

Checks attributes of manager accounts

Description

Currently, the only supported parameter is "sessions" which will return the current number of active sessions for this AMI account.

Syntax

```
AMI_CLIENT(loginname,field)
```

Arguments

- `loginname` - Login name, specified in manager.conf
- `field` - The manager account attribute to return
    - `sessions` - The number of sessions for this AMI account

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_ARRAY

ARRAY()

Synopsis

Allows setting multiple variables at once.

Description

The comma-delimited list passed as a value to which the function is set will be interpreted as a set of values to which the comma-delimited list of variable names in the argument should be set.

Example: Set(ARRAY(var1,var2)=1,2) will set var1 to 1 and var2 to 2

Syntax

```
ARRAY(var1[,var2[,...][,varN]])
```

Arguments

- `var1`
- `var2`
- `varN`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_AST_CONFIG

AST_CONFIG()

Synopsis

Retrieve a variable from a configuration file.

Description

This function reads a variable from an Asterisk configuration file.

Syntax

```
AST_CONFIG(config_file,category,variable_name)
```

Arguments

- `config_file`
- `category`
- `variable_name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_AUDIOHOOK_INHERIT

AUDIOHOOK_INHERIT()

Synopsis

Set whether an audiohook may be inherited to another channel

Description

By enabling audiohook inheritance on the channel, you are giving permission for an audiohook to be inherited by a descendent channel. Inheritance may be be disabled at any point as well.

Example scenario:

exten => 2000,1,MixMonitor(blah.wav)

exten => 2000,n,Set(AUDIOHOOK_INHERIT(MixMonitor)=yes)

exten => 2000,n,Dial(SIP/2000)

exten => 4000,1,Dial(SIP/4000)

exten => 5000,1,MixMonitor(blah2.wav)

exten => 5000,n,Dial(SIP/5000)

In this basic dialplan scenario, let's consider the following sample calls

Call 1: Caller dials 2000. The person who answers then executes an attended

transfer to 4000.

Result: Since extension 2000 set MixMonitor to be inheritable, after the

transfer to 4000 has completed, the call will continue to be recorded to blah.wav

Call 2: Caller dials 5000. The person who answers then executes an attended

transfer to 4000.

Result: Since extension 5000 did not set MixMonitor to be inheritable, the

recording will stop once the call has been transferred to 4000.

Syntax

```
AUDIOHOOK_INHERIT(source)
```

Arguments

- `source` - The built-in sources in Asterisk are
  - `MixMonitor`
  - `Chanspy`
  - `Volume`
  - `Speex`
  - `pitch_shift`
  - `JACK_HOOK`
  - `Mute`
    Note that the names are not case-sensitive

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_BASE64_DECODE

BASE64_DECODE()

Synopsis

Decode a base64 string.

Description

Returns the plain text string.

Syntax

```
BASE64_DECODE(string)
```

Arguments

- `string` - Input string.

See Also

- Asterisk 11 Function_BASE64_ENCODE
- Asterisk 11 Function_AES_DECRYPT
- Asterisk 11 Function_AES_ENCRYPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_BASE64_ENCODE

BASE64_ENCODE()

Synopsis

Encode a string in base64.

Description

Returns the base64 string.

Syntax

```
BASE64_ENCODE(string)
```

Arguments

- `string` - Input string

See Also

- Asterisk 11 Function_BASE64_DECODE
- Asterisk 11 Function_AES_DECRYPT
- Asterisk 11 Function_AES_ENCRYPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_BLACKLIST

BLACKLIST()

Synopsis

Check if the callerid is on the blacklist.

Description

Uses astdb to check if the Caller*ID is in family `blacklist`. Returns `1` or `0`.

Syntax

```
BLACKLIST()
```

Arguments

See Also

- Asterisk 11 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALENDAR_BUSY

CALENDAR_BUSY()

Synopsis

Determine if the calendar is marked busy at this time.

Description

Check the specified calendar's current busy status.

Syntax

```
CALENDAR_BUSY(calendar)
```

Arguments

- `calendar`

See Also

- Asterisk 11 Function_CALENDAR_EVENT
- Asterisk 11 Function_CALENDAR_QUERY
- Asterisk 11 Function_CALENDAR_QUERY_RESULT
- Asterisk 11 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALENDAR_EVENT

CALENDAR_EVENT()

Synopsis

Get calendar event notification data from a notification call.

Description

Whenever a calendar event notification call is made, the event data may be accessed with this function.

Syntax

```
CALENDAR_EVENT(field)
```

Arguments

- `field`
  - `summary` - The VEVENT SUMMARY property or Exchange event 'subject'
  - `description` - The text description of the event
  - `organizer` - The organizer of the event
  - `location` - The location of the eventt
  - `categories` - The categories of the event
  - `priority` - The priority of the event
  - `calendar` - The name of the calendar associated with the event
  - `uid` - The unique identifier for this event
  - `start` - The start time of the event
  - `end` - The end time of the event
  - `busystate` - The busy state of the event 0=FREE, 1=TENTATIVE, 2=BUSY

See Also

- Asterisk 11 Function_CALENDAR_BUSY
- Asterisk 11 Function_CALENDAR_QUERY
- Asterisk 11 Function_CALENDAR_QUERY_RESULT
- Asterisk 11 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALENDAR_QUERY

CALENDAR_QUERY()

Synopsis

Query a calendar server and store the data on a channel

Description

Get a list of events in the currently accessible timeframe of the *calendar* The function returns the id for accessing the result with CALENDAR_QUERY_RESULT()

Syntax

```
CALENDAR_QUERY(calendar[,start[,end]])
```

Arguments

- `calendar` - The calendar that should be queried
- `start` - The start time of the query (in seconds since epoch)
- `end` - The end time of the query (in seconds since epoch)

See Also

- Asterisk 11 Function_CALENDAR_BUSY
- Asterisk 11 Function_CALENDAR_EVENT
- Asterisk 11 Function_CALENDAR_QUERY_RESULT
- Asterisk 11 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALENDAR_QUERY_RESULT

CALENDAR_QUERY_RESULT()

Synopsis

Retrieve data from a previously run CALENDAR_QUERY() call

Description

After running CALENDAR_QUERY and getting a result *id*, calling `CALENDAR_QUERY` with that *id* and a *field* will return the data for that field. If multiple events matched the query, and *entry* is provided, information from that event will be returned.

Syntax

```
CALENDAR_QUERY_RESULT(id,field[,entry])
```

Arguments

- `id` - The query ID returned by `CALENDAR_QUERY`
- `field`
  - `getnum` - number of events occurring during time range
  - `summary` - A summary of the event
  - `description` - The full event description
  - `organizer` - The event organizer
  - `location` - The event location
  - `categories` - The categories of the event
  - `priority` - The priority of the event
  - `calendar` - The name of the calendar associted with the event
  - `uid` - The unique identifier for the event
  - `start` - The start time of the event (in seconds since epoch)
  - `end` - The end time of the event (in seconds since epoch)
  - `busystate` - The busy status of the event 0=FREE, 1=TENTATIVE, 2=BUSY
- `entry` - Return data from a specific event returned by the query

See Also

- Asterisk 11 Function_CALENDAR_BUSY
- Asterisk 11 Function_CALENDAR_EVENT
- Asterisk 11 Function_CALENDAR_QUERY
- Asterisk 11 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALENDAR_WRITE

CALENDAR_WRITE()

Synopsis

Write an event to a calendar

Description

Example: CALENDAR_WRITE(calendar,field1,field2,field3)=val1,val2,val3

The field and value arguments can easily be set/passed using the HASHKEYS() and HASH() functions

- `CALENDAR_SUCCESS` - The status of the write operation to the calendar
    - `1` - The event was successfully written to the calendar.
    - `0` - The event was not written to the calendar due to network issues, permissions, etc.

Syntax

```
CALENDAR_WRITE(calendar,field[,...])
```

Arguments

- `calendar` - The calendar to write to
- `field`
    - `summary` - A summary of the event
    - `description` - The full event description
    - `organizer` - The event organizer
    - `location` - The event location
    - `categories` - The categories of the event
    - `priority` - The priority of the event
    - `uid` - The unique identifier for the event
    - `start` - The start time of the event (in seconds since epoch)
    - `end` - The end time of the event (in seconds since epoch)
    - `busystate` - The busy status of the event 0=FREE, 1=TENTATIVE, 2=BUSY

See Also

- Asterisk 11 Function_CALENDAR_BUSY
- Asterisk 11 Function_CALENDAR_EVENT
- Asterisk 11 Function_CALENDAR_QUERY
- Asterisk 11 Function_CALENDAR_QUERY_RESULT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALLCOMPLETION

CALLCOMPLETION()

Synopsis

Get or set a call completion configuration parameter for a channel.

Description

The CALLCOMPLETION function can be used to get or set a call completion configuration parameter for a channel. Note that setting a configuration parameter will only change the parameter for the duration of the call. For more information see `doc/AST.pdf`. For more information on call completion parameters, see `configs/ccss.conf.sample`.

Syntax

```
CALLCOMPLETION(option)
```

Arguments

- `option` - The allowable options are:
    - `cc_agent_policy`
    - `cc_monitor_policy`
    - `cc_offer_timer`
    - `ccnr_available_timer`
    - `ccbs_available_timer`
    - `cc_recall_timer`
    - `cc_max_agents`
    - `cc_max_monitors`
    - `cc_callback_macro`
    - `cc_agent_dialstring`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALLERID

CALLERID()

Synopsis

Gets or sets Caller*ID data on the channel.

Description

Gets or sets Caller*ID data on the channel. Uses channel callerid by default or optional callerid, if specified.

The allowable values for the *name-charset* field are the following:

- `unknown` - Unknown
- `iso8859-1` - ISO8859-1
- `withdrawn` - Withdrawn
- `iso8859-2` - ISO8859-2
- `iso8859-3` - ISO8859-3
- `iso8859-4` - ISO8859-4
- `iso8859-5` - ISO8859-5
- `iso8859-7` - ISO8859-7
- `bmp` - ISO10646 Bmp String
- `utf8` - ISO10646 UTF-8 String

Syntax

```
CALLERID(datatype,CID)
```

Arguments

- `datatype` - The allowable datatypes are:
    - `all`
    - `name`
    - `name-valid`
    - `name-charset`
    - `name-pres`
    - `num`
    - `num-valid`
    - `num-plan`
    - `num-pres`
    - `subaddr`
    - `subaddr-valid`
    - `subaddr-type`
    - `subaddr-odd`
    - `tag`
    - `priv-all`
    - `priv-name`
    - `priv-name-valid`
    - `priv-name-charset`
    - `priv-name-pres`
    - `priv-num`
    - `priv-num-valid`
    - `priv-num-plan`
    - `priv-num-pres`
    - `priv-subaddr`
    - `priv-subaddr-valid`
    - `priv-subaddr-type`
    - `priv-subaddr-odd`
    - `priv-tag`
    - `ANI-all`

    - `ANI-name`
    - `ANI-name-valid`
    - `ANI-name-charset`
    - `ANI-name-pres`
    - `ANI-num`
    - `ANI-num-valid`
    - `ANI-num-plan`
    - `ANI-num-pres`
    - `ANI-tag`
    - `RDNIS`
    - `DNID`
    - `dnid-num-plan`
    - `dnid-subaddr`
    - `dnid-subaddr-valid`
    - `dnid-subaddr-type`
    - `dnid-subaddr-odd`
- `CID` - Optional Caller*ID to parse instead of using the Caller*ID from the channel. This parameter is only optional when reading the Caller*ID.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CALLERPRES

CALLERPRES()

Synopsis

Gets or sets Caller*ID presentation on the channel.

Description

Gets or sets Caller*ID presentation on the channel. This function is deprecated in favor of CALLERID(num-pres) and CALLERID(name-pres). The following values are valid:

- `allowed_not_screened` - Presentation Allowed, Not Screened.
- `allowed_passed_screen` - Presentation Allowed, Passed Screen.
- `allowed_failed_screen` - Presentation Allowed, Failed Screen.
- `allowed` - Presentation Allowed, Network Number.
- `prohib_not_screened` - Presentation Prohibited, Not Screened.
- `prohib_passed_screen` - Presentation Prohibited, Passed Screen.
- `prohib_failed_screen` - Presentation Prohibited, Failed Screen.
- `prohib` - Presentation Prohibited, Network Number.
- `unavailable` - Number Unavailable.

Syntax

```
CALLERPRES()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CDR

CDR()

Synopsis

Gets or sets a CDR variable.

Description

All of the CDR field names are read-only, except for `accountcode`, `userfield`, and `amaflags`. You may, however, supply a name not on the above list, and create your own variable, whose value can be changed with this function, and this variable will be stored on the cdr.

> **Note**
> For setting CDR values, the `l` flag does not apply to setting the `accountcode`, `userfield`, or `amaflags`.
>
> CDRs can only be modified before the bridge between two channels is torn down. For example, CDRs may not be modified after the `Dial` application has returned.

Raw values for `disposition`:

- `0` - NO ANSWER
- `1` - NO ANSWER (NULL record)
- `2` - FAILED
- `4` - BUSY
- `8` - ANSWERED
  Raw values for `amaflags`:

- `1` - OMIT
- `2` - BILLING
- `3` - DOCUMENTATION
  Example: exten => 1,1,Set(CDR(userfield)=test)

Syntax

```
CDR(name[,options])
```

Arguments

- `name` - CDR field name:
  - `clid` - Caller ID.
  - `lastdata` - Last application arguments.
  - `disposition` - ANSWERED, NO ANSWER, BUSY, FAILED.
  - `src` - Source.
  - `start` - Time the call started.
  - `amaflags` - DOCUMENTATION, BILL, IGNORE, etc.
  - `dst` - Destination.
  - `answer` - Time the call was answered.
  - `accountcode` - The channel's account code.
  - `dcontext` - Destination context.
  - `end` - Time the call ended.
  - `uniqueid` - The channel's unique id.
  - `dstchannel` - Destination channel.
  - `duration` - Duration of the call.
  - `userfield` - The channel's user specified field.
  - `lastapp` - Last application.
  - `billsec` - Duration of the call once it was answered.
  - `channel` - Channel name.
  - `sequence` - CDR sequence number.

- `options`
  - `f` - Returns billsec or duration fields as floating point values.
  - `l` - Uses the most recent CDR on a channel with multiple records
  - `r` - Searches the entire stack of CDRs on the channel.
  - `s` - Skips any CDR's that are marked 'LOCKED' due to forkCDR() calls. (on setting/writing CDR vars only)
  - `u` - Retrieves the raw, unprocessed value.
    For example, 'start', 'answer', and 'end' will be retrieved as epoch values, when the `u` option is passed, but formatted as YYYY-MM-DD HH:MM:SS otherwise. Similarly, disposition and amaflags will return their raw integral values.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CHANNEL

CHANNEL()

Synopsis

Gets/sets various pieces of information about the channel.

Description

Gets/sets various pieces of information about the channel, additional *item* may be available from the channel driver; see its documentation for details. Any *it em* requested that is not available on the current channel will return an empty string.

Syntax

```
CHANNEL(item)
```

Arguments

- `item` - Standard items (provided by all channel technologies) are:
    - `amaflags` - R/W the Automatic Message Accounting (AMA) flags on the channel. When read from a channel, the integer value will always be returned. When written to a channel, both the string format or integer value is accepted.
        - `1` - OMIT
        - `2` - BILLING
        - `3` - DOCUMENTATION
    - `accountcode` - R/W the channel's account code.
    - `audioreadformat` - R/O format currently being read.
    - `audionativeformat` - R/O format used natively for audio.
    - `audiowriteformat` - R/O format currently being written.
    - `callgroup` - R/W numeric call pickup groups that this channel is a member.
    - `pickupgroup` - R/W numeric call pickup groups this channel can pickup.
    - `namedcallgroup` - R/W named call pickup groups that this channel is a member.
    - `namedpickupgroup` - R/W named call pickup groups this channel can pickup.
    - `channeltype` - R/O technology used for channel.
    - `checkhangup` - R/O Whether the channel is hanging up (1/0)
    - `hangup_handler_pop` - W/O Replace the most recently added hangup handler with a new hangup handler on the channel if supplied. The assigned string is passed to the Gosub application when the channel is hung up. Any optionally omitted context and exten are supplied by the channel pushing the handler before it is pushed.
    - `hangup_handler_push` - W/O Push a hangup handler onto the channel hangup handler stack. The assigned string is passed to the Gosub application when the channel is hung up. Any optionally omitted context and exten are supplied by the channel pushing the handler before it is pushed.
    - `hangup_handler_wipe` - W/O Wipe the entire hangup handler stack and replace with a new hangup handler on the channel if supplied. The assigned string is passed to the Gosub application when the channel is hung up. Any optionally omitted context and exten are supplied by the channel pushing the handler before it is pushed.
    - `language` - R/W language for sounds played.
    - `musicclass` - R/W class (from musiconhold.conf) for hold music.
    - `name` - The name of the channel
    - `parkinglot` - R/W parkinglot for parking.
    - `rxgain` - R/W set rxgain level on channel drivers that support it.
    - `secure_bridge_signaling` - Whether or not channels bridged to this channel require secure signaling
    - `secure_bridge_media` - Whether or not channels bridged to this channel require secure media
    - `state` - R/O state for channel
    - `tonezone` - R/W zone for indications played
    - `transfercapability` - R/W ISDN Transfer Capability, one of:
        - `SPEECH`
        - `DIGITAL`
        - `RESTRICTED_DIGITAL`
        - `3K1AUDIO`
        - `DIGITAL_W_TONES`

    - VIDEO
- txgain - R/W set txgain level on channel drivers that support it.
- videonativeformat - R/O format used natively for video
- trace - R/W whether or not context tracing is enabled, only available **if CHANNEL_TRACE is defined**.
  chan_sip provides the following additional options:
- peerip - R/O Get the IP address of the peer.
- recvip - R/O Get the source IP address of the peer.
- from - R/O Get the URI from the From: header.
- uri - R/O Get the URI from the Contact: header.
- useragent - R/O Get the useragent.
- peername - R/O Get the name of the peer.
- t38passthrough - R/O 1 if T38 is offered or enabled in this channel, otherwise 0
- rtpqos - R/O Get QOS information about the RTP stream
  This option takes two additional arguments:
  Argument 1:
  audio Get data about the audio stream
  video Get data about the video stream
  text Get data about the text stream
  Argument 2:
  local_ssrc Local SSRC (stream ID)
  local_lostpackets Local lost packets
  local_jitter Local calculated jitter
  local_maxjitter Local calculated jitter (maximum)
  local_minjitter Local calculated jitter (minimum)
  {{local_normdevjitter}}Local calculated jitter (normal deviation)
  local_stdevjitter Local calculated jitter (standard deviation)
  local_count Number of received packets
  remote_ssrc Remote SSRC (stream ID)
  {{remote_lostpackets}}Remote lost packets
  remote_jitter Remote reported jitter
  remote_maxjitter Remote calculated jitter (maximum)
  remote_minjitter Remote calculated jitter (minimum)
  {{remote_normdevjitter}}Remote calculated jitter (normal deviation)
  {{remote_stdevjitter}}Remote calculated jitter (standard deviation)
  remote_count Number of transmitted packets
  rtt Round trip time
  maxrtt Round trip time (maximum)
  minrtt Round trip time (minimum)
  normdevrtt Round trip time (normal deviation)
  stdevrtt Round trip time (standard deviation)
  all All statistics (in a form suited to logging, but not for parsing)
- rtpdest - R/O Get remote RTP destination information.
  This option takes one additional argument:
  Argument 1:
  audio Get audio destination
  video Get video destination
  text Get text destination
  Defaults to audio if unspecified.
- rtpsource - R/O Get source RTP destination information.
  This option takes one additional argument:
  Argument 1:
  audio Get audio destination
  video Get video destination
  text Get text destination
  Defaults to audio if unspecified.
  chan_iax2 provides the following additional options:
- osptoken - R/O Get the peer's osptoken.
- peerip - R/O Get the peer's ip address.

- `peername` - R/O Get the peer's username.
- `secure_signaling` - R/O Get the if the IAX channel is secured.
- `secure_media` - R/O Get the if the IAX channel is secured.
  chan_dahdi provides the following additional options:
- `dahdi_channel` - R/O DAHDI channel related to this channel.
- `dahdi_span` - R/O DAHDI span related to this channel.
- `dahdi_type` - R/O DAHDI channel type, one of:
  - `analog`
  - `mfc/r2`
  - `pri`
  - `pseudo`
  - `ss7`
- `keypad_digits` - R/O PRI Keypad digits that came in with the SETUP message.
- `reversecharge` - R/O PRI Reverse Charging Indication, one of:
  - `-1` - None
  - {{ 1}} - Reverse Charging Requested
- `no_media_path` - R/O PRI Nonzero if the channel has no B channel. The channel is either on hold or a call waiting call.
- `buffers` - W/O Change the channel's buffer policy (for the current call only)
  This option takes two arguments:
  Number of buffers,
  Buffer policy being one of:
  `full`
  `immediate`
  `half`
- `echocan_mode` - W/O Change the configuration of the active echo canceller on the channel (if any), for the current call only.
  Possible values are:
  {{on}}Normal mode (the echo canceller is actually reinitalized)
  {{off}}Disabled
  {{fax}}FAX/data mode (NLP disabled if possible, otherwise completely disabled)
  {{voice}}Voice mode (returns from FAX mode, reverting the changes that were made)
  chan_ooh323 provides the following additional options:
- `faxdetect` - R/W Fax Detect
  Returns 0 or 1
  Write yes or no
- `t38support` - R/W t38support
  Returns 0 or 1
  Write yes or no
- `h323id_url` - R/0 Returns caller URL
- `caller_h323id` - R/0 Returns caller h323id
- `caller_dialeddigits` - R/0 Returns caller dialed digits
- `caller_email` - R/0 Returns caller email
- `callee_email` - R/0 Returns callee email
- `callee_dialeddigits` - R/0 Returns callee dialed digits
- `caller_url` - R/0 Returns caller URL

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r395020

Asterisk 11 Function_CHANNELS

CHANNELS()

Synopsis

Gets the list of channels, optionally filtering by a regular expression.

Description

Gets the list of channels, optionally filtering by a *regular_expression*. If no argument is provided, all known channels are returned. The *regular_expression* must correspond to the POSIX.2 specification, as shown in **regex(7)**. The list returned will be space-delimited.

Syntax

```
CHANNELS(regular_expression)
```

Arguments

- `regular_expression`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CHECKSIPDOMAIN

CHECKSIPDOMAIN()

Synopsis

Checks if domain is a local domain.

Description

This function checks if the *domain* in the argument is configured as a local SIP domain that this Asterisk server is configured to handle. Returns the domain name if it is locally handled, otherwise an empty string. Check the `domain=` configuration in `sip.conf`.

Syntax

```
CHECKSIPDOMAIN(domain)
```

Arguments

- `domain`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CONFBRIDGE

CONFBRIDGE()

Synopsis

Set a custom dynamic bridge and user profile on a channel for the ConfBridge application using the same options defined in confbridge.conf.

Description

---- Example 1 ----

In this example the custom set user profile on this channel will automatically be used by the ConfBridge app.

exten => 1,1,Answer()

exten => 1,n,Set(CONFBRIDGE(user,announce_join_leave)=yes)

exten => 1,n,Set(CONFBRIDGE(user,startmuted)=yes)

exten => 1,n,ConfBridge(1)

---- Example 2 ----

This example shows how to use a predefined user or bridge profile in confbridge.conf as a template for a dynamic profile. Here we make a admin/marked user out of the default_user profile that is already defined in confbridge.conf.

exten => 1,1,Answer()

exten => 1,n,Set(CONFBRIDGE(user,template)=default_user)

exten => 1,n,Set(CONFBRIDGE(user,admin)=yes)

exten => 1,n,Set(CONFBRIDGE(user,marked)=yes)

exten => 1,n,ConfBridge(1)

Syntax

```
CONFBRIDGE(type,option)
```

Arguments

- `type` - Type refers to which type of profile the option belongs too. Type can be `bridge` or `user`.
- `option` - Option refers to `confbridge.conf` option that is being set dynamically on this channel.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CONFBRIDGE_INFO

CONFBRIDGE_INFO()

Synopsis

Get information about a ConfBridge conference.

Description

This function returns a non-negative integer for valid conference identifiers (0 or 1 for `locked`) and "" for invalid conference identifiers.

Syntax

```
CONFBRIDGE_INFO(type,conf)
```

Arguments

- `type` - Type can be `parties`, `admins`, `marked`, or `locked`.
- `conf` - Conf refers to the name of the conference being referenced.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_CONNECTEDLINE

CONNECTEDLINE()

Synopsis

Gets or sets Connected Line data on the channel.

Description

Gets or sets Connected Line data on the channel.

The allowable values for the *name-charset* field are the following:

- `unknown` - Unknown
- `iso8859-1` - ISO8859-1
- `withdrawn` - Withdrawn
- `iso8859-2` - ISO8859-2
- `iso8859-3` - ISO8859-3
- `iso8859-4` - ISO8859-4
- `iso8859-5` - ISO8859-5
- `iso8859-7` - ISO8859-7
- `bmp` - ISO10646 Bmp String
- `utf8` - ISO10646 UTF-8 String

Syntax

```
CONNECTEDLINE(datatype,i)
```

Arguments

- `datatype` - The allowable datatypes are:
    - `all`
    - `name`
    - `name-valid`
    - `name-charset`
    - `name-pres`
    - `num`
    - `num-valid`
    - `num-plan`
    - `num-pres`
    - `subaddr`
    - `subaddr-valid`
    - `subaddr-type`
    - `subaddr-odd`
    - `tag`
    - `priv-all`
    - `priv-name`
    - `priv-name-valid`
    - `priv-name-charset`
    - `priv-name-pres`
    - `priv-num`
    - `priv-num-valid`
    - `priv-num-plan`
    - `priv-num-pres`
    - `priv-subaddr`
    - `priv-subaddr-valid`
    - `priv-subaddr-type`
    - `priv-subaddr-odd`
    - `priv-tag`

- `i` - If set, this will prevent the channel from sending out protocol messages because of the value being set

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CSV_QUOTE

CSV_QUOTE()

Synopsis

Quotes a given string for use in a CSV file, escaping embedded quotes as necessary

Description

Example: ${CSV_QUOTE("a,b" 123)} will return """a,b"" 123"

Syntax

```
CSV_QUOTE(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_CURL

CURL()

Synopsis

Retrieve content from a remote web or ftp server

Description

Syntax

```
CURL(url,post-data)
```

Arguments

- `url`
- `post-data` - If specified, an `HTTP POST` will be performed with the content of *post-data*, instead of an `HTTP GET` (default).

See Also

- Asterisk 11 Function_CURLOPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CURLOPT

CURLOPT()

Synopsis

Sets various options for future invocations of CURL.

Description

Options may be set globally or per channel. Per-channel settings will override global settings.

Syntax

```
CURLOPT(key)
```

Arguments

- `key`
  - `cookie` - A cookie to send with the request. Multiple cookies are supported.
  - `conntimeout` - Number of seconds to wait for a connection to succeed
  - `dnstimeout` - Number of seconds to wait for DNS to be resolved
  - `ftptext` - For FTP URIs, force a text transfer (boolean)
  - `ftptimeout` - For FTP URIs, number of seconds to wait for a server response
  - `header` - Include header information in the result (boolean)
  - `httptimeout` - For HTTP(S) URIs, number of seconds to wait for a server response
  - `maxredirs` - Maximum number of redirects to follow
  - `proxy` - Hostname or IP address to use as a proxy server
  - `proxytype` - Type of `proxy`
    - `http`
    - `socks4`
    - `socks5`
  - `proxyport` - Port number of the `proxy`
  - `proxyuserpwd` - A *username*`:`*password* combination to use for authenticating requests through a `proxy`
  - `referer` - Referer URL to use for the request
  - `useragent` - UserAgent string to use for the request
  - `userpwd` - A *username*`:`*password* to use for authentication when the server response to an initial request indicates a 401 status code.
  - `ssl_verifypeer` - Whether to verify the server certificate against a list of known root certificate authorities (boolean).
  - `hashcompat` - Assuming the responses will be in `key1=value1&key2=value2` format, reformat the response such that it can be used by the `HASH` function.
    - `yes`
    - `no`
    - `legacy` - Also translate + to the space character, in violation of current RFC standards.

See Also

- Asterisk 11 Function_CURL
- Asterisk 11 Function_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_CUT

CUT()

Synopsis

Slices and dices strings, based upon a named delimiter.

Description

Cut out information from a string (*varname*), based upon a named delimiter.

Syntax

```
CUT(varname,char-delim,range-spec)
```

Arguments

- `varname` - Variable you want cut
- `char-delim` - Delimiter, defaults to `-`
- `range-spec` - Number of the field you want (1-based offset), may also be specified as a range (with `-`) or group of ranges and fields (with &)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DB

DB()

Synopsis

Read from or write to the Asterisk database.

Description

This function will read from or write a value to the Asterisk database. On a read, this function returns the corresponding value from the database, or blank if it does not exist. Reading a database value will also set the variable DB_RESULT. If you wish to find out if an entry exists, use the DB_EXISTS function.

Syntax

```
DB(family/key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 11 Application_DBdel
- Asterisk 11 Function_DB_DELETE
- Asterisk 11 Application_DBdeltree
- Asterisk 11 Function_DB_EXISTS

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DB_DELETE

DB_DELETE()

Synopsis

Return a value from the database and delete it.

Description

This function will retrieve a value from the Asterisk database and then remove that key from the database. `DB_RESULT` will be set to the key's value if it exists.

Syntax

```
DB_DELETE(family/key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 11 Application_DBdel
- Asterisk 11 Function_DB
- Asterisk 11 Application_DBdeltree

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DB_EXISTS

DB_EXISTS()

Synopsis

Check to see if a key exists in the Asterisk database.

Description

This function will check to see if a key exists in the Asterisk database. If it exists, the function will return `1`. If not, it will return `0`. Checking for existence of a database key will also set the variable DB_RESULT to the key's value if it exists.

Syntax

```
DB_EXISTS(family/key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 11 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DB_KEYS

DB_KEYS()

Synopsis

Obtain a list of keys within the Asterisk database.

Description

This function will return a comma-separated list of keys existing at the prefix specified within the Asterisk database. If no argument is provided, then a list of key families will be returned.

Syntax

```
DB_KEYS(prefix)
```

Arguments

- `prefix`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DEC

DEC()

Synopsis

Decrements the value of a variable, while returning the updated value to the dialplan

Description

Decrements the value of a variable, while returning the updated value to the dialplan

Example: DEC(MyVAR) - Decrements MyVar

Note: DEC(${MyVAR}) - Is wrong, as DEC expects the variable name, not its value

Syntax

```
DEC(variable)
```

Arguments

- `variable` - The variable name to be manipulated, without the braces.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DENOISE

DENOISE()

Synopsis

Apply noise reduction to audio on a channel.

Description

The DENOISE function will apply noise reduction to audio on the channel that it is executed on. It is very useful for noisy analog lines, especially when adjusting gains or using AGC. Use `rx` for audio received from the channel and `tx` to apply the filter to the audio being sent to the channel.

Examples:

exten => 1,1,Set(DENOISE(rx)=on)

exten => 1,2,Set(DENOISE(tx)=off)

Syntax

```
DENOISE(channeldirection)
```

Arguments

- `channeldirection` - This can be either `rx` or `tx` the values that can be set to this are either `on` and `off`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DEVICE_STATE

DEVICE_STATE()

Synopsis

Get or Set a device state.

Description

The DEVICE_STATE function can be used to retrieve the device state from any device state provider. For example:

NoOp(SIP/mypeer has state ${DEVICE_STATE(SIP/mypeer)})

NoOp(Conference number 1234 has state ${DEVICE_STATE(MeetMe:1234)})

The DEVICE_STATE function can also be used to set custom device state from the dialplan. The `Custom:` prefix must be used. For example:

Set(DEVICE_STATE(Custom:lamp1)=BUSY)

Set(DEVICE_STATE(Custom:lamp2)=NOT_INUSE)

You can subscribe to the status of a custom device state using a hint in the dialplan:

exten => 1234,hint,Custom:lamp1

The possible values for both uses of this function are:

UNKNOWN | NOT_INUSE | INUSE | BUSY | INVALID | UNAVAILABLE | RINGING | RINGINUSE | ONHOLD

Syntax

```
DEVICE_STATE(device)
```

Arguments

- `device`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DIALGROUP

DIALGROUP()

Synopsis

Manages a group of users for dialing.

Description

Presents an interface meant to be used in concert with the Dial application, by presenting a list of channels which should be dialled when referenced.

When DIALGROUP is read from, the argument is interpreted as the particular *group* for which a dial should be attempted. When DIALGROUP is written to with no arguments, the entire list is replaced with the argument specified.

Functionality is similar to a queue, except that when no interfaces are available, execution may continue in the dialplan. This is useful when you want certain people to be the first to answer any calls, with immediate fallback to a queue when the front line people are busy or unavailable, but you still want front line people to log in and out of that group, just like a queue.

Example:

exten => 1,1,Set(DIALGROUP(mygroup,add)=SIP/10)

exten => 1,n,Set(DIALGROUP(mygroup,add)=SIP/20)

exten => 1,n,Dial(${DIALGROUP(mygroup)})

Syntax

```
DIALGROUP(group,op)
```

Arguments

- `group`
- `op` - The operation name, possible values are:
  add - add a channel name or interface (write-only)
  del - remove a channel name or interface (write-only)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DIALPLAN_EXISTS

DIALPLAN_EXISTS()

Synopsis

Checks the existence of a dialplan target.

Description

This function returns `1` if the target exits. Otherwise, it returns `0`.

Syntax

```
DIALPLAN_EXISTS(context,extension,priority)
```

Arguments

- `context`
- `extension`
- `priority`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DUNDILOOKUP

DUNDILOOKUP()

Synopsis

Do a DUNDi lookup of a phone number.

Description

This will do a DUNDi lookup of the given phone number.

This function will return the Technology/Resource found in the first result in the DUNDi lookup. If no results were found, the result will be blank.

Syntax

```
DUNDILOOKUP(number,context,options)
```

Arguments

- `number`
- `context` - If not specified the default will be `e164`.
- `options`
    - b - Bypass the internal DUNDi cache

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DUNDIQUERY

DUNDIQUERY()

Synopsis

Initiate a DUNDi query.

Description

This will do a DUNDi lookup of the given phone number.

The result of this function will be a numeric ID that can be used to retrieve the results with the `DUNDIRESULT` function.

Syntax

```
DUNDIQUERY(number,context,options)
```

Arguments

- `number`
- `context` - If not specified the default will be `e164`.
- `options`
    - `b` - Bypass the internal DUNDi cache

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_DUNDIRESULT

DUNDIRESULT()

Synopsis

Retrieve results from a DUNDIQUERY.

Description

This function will retrieve results from a previous use\n" of the `DUNDIQUERY` function.

Syntax

```
DUNDIRESULT(id,resultnum)
```

Arguments

- `id` - The identifier returned by the `DUNDIQUERY` function.
- `resultnum`
    - `number` - The number of the result that you want to retrieve, this starts at `1`
    - `getnum` - The total number of results that are available.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_ENUMLOOKUP

ENUMLOOKUP()

Synopsis

General or specific querying of NAPTR records for ENUM or ENUM-like DNS pointers.

Description

For more information see `doc/AST.pdf`.

Syntax

```
ENUMLOOKUP(number,method-type,options,record#,zone-suffix)
```

Arguments

- `number`
- `method-type` - If no *method-type* is given, the default will be `sip`.
- `options`
    - `c` - Returns an integer count of the number of NAPTRs of a certain RR type.
      Combination of `c` and Method-type of `ALL` will return a count of all NAPTRs for the record or -1 on error.
    - `u` - Returns the full URI and does not strip off the URI-scheme.
    - `s` - Triggers ISN specific rewriting.
    - `i` - Looks for branches into an Infrastructure ENUM tree.
    - `d` - for a direct DNS lookup without any flipping of digits.
- `record#` - If no *record#* is given, defaults to `1`.
- `zone-suffix` - If no *zone-suffix* is given, the default will be `e164.arpa`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ENUMQUERY

ENUMQUERY()

Synopsis

Initiate an ENUM query.

Description

This will do a ENUM lookup of the given phone number.

Syntax

```
ENUMQUERY(number,method-type,zone-suffix)
```

Arguments

- `number`
- `method-type` - If no *method-type* is given, the default will be `sip`.
- `zone-suffix` - If no *zone-suffix* is given, the default will be `e164.arpa`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ENUMRESULT

ENUMRESULT()

Synopsis

Retrieve results from a ENUMQUERY.

Description

This function will retrieve results from a previous use of the ENUMQUERY function.

Syntax

```
ENUMRESULT(id,resultnum)
```

Arguments

- `id` - The identifier returned by the ENUMQUERY function.
- `resultnum` - The number of the result that you want to retrieve.
  Results start at `1`. If this argument is specified as `getnum`, then it will return the total number of results that are available or `-1` on error.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ENV

ENV()

Synopsis

Gets or sets the environment variable specified.

Description

Variables starting with AST_ are reserved to the system and may not be set.

Syntax

```
ENV(varname)
```

Arguments

- `varname` - Environment variable name

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_EVAL

EVAL()

Synopsis

Evaluate stored variables

Description

Using EVAL basically causes a string to be evaluated twice. When a variable or expression is in the dialplan, it will be evaluated at runtime. However, if the results of the evaluation is in fact another variable or expression, using EVAL will have it evaluated a second time.

Example: If the `MYVAR` contains `OTHERVAR`, then the result of ${EVAL(`MYVAR`)} in the dialplan will be the contents of `OTHERVAR`. Normally just putting `MYVAR` in the dialplan the result would be `OTHERVAR`.

Syntax

```
EVAL(variable)
```

Arguments

- `variable`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_EXCEPTION

EXCEPTION()

Synopsis

Retrieve the details of the current dialplan exception.

Description

Retrieve the details (specified *field*) of the current dialplan exception.

Syntax

```
EXCEPTION(field)
```

Arguments

- `field` - The following fields are available for retrieval:
  - `reason` - INVALID, ERROR, RESPONSETIMEOUT, ABSOLUTETIMEOUT, or custom value set by the RaiseException() application
  - `context` - The context executing when the exception occurred.
  - `exten` - The extension executing when the exception occurred.
  - `priority` - The numeric priority executing when the exception occurred.

See Also

- Asterisk 11 Application_RaiseException

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_EXISTS

EXISTS()

Synopsis

Test the existence of a value.

Description

Returns `1` if exists, `0` otherwise.

Syntax

```
EXISTS(data)
```

Arguments

- `data`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_EXTENSION_STATE

EXTENSION_STATE()

Synopsis

Get an extension's state.

Description

The EXTENSION_STATE function can be used to retrieve the state from any hinted extension. For example:

NoOp(1234@default has state ${EXTENSION_STATE(1234)})

NoOp(4567@home has state ${EXTENSION_STATE(4567@home)})

The possible values returned by this function are:

UNKNOWN | NOT_INUSE | INUSE | BUSY | INVALID | UNAVAILABLE | RINGING | RINGINUSE | HOLDINUSE | ONHOLD

Syntax

```
EXTENSION_STATE(extension@context)
```

Arguments

- `extension`
- `context` - If it is not specified defaults to `default`.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FAXOPT

FAXOPT()

Synopsis

Gets/sets various pieces of information about a fax session.

Description

FAXOPT can be used to override the settings for a FAX session listed in `res_fax.conf`, it can also be used to retreive information about a FAX session that has finished eg. pages/status.

Syntax

```
FAXOPT(item)
```

Arguments

- `item`
    - `ecm` - R/W Error Correction Mode (ECM) enable with 'yes', disable with 'no'.
    - `error` - R/O FAX transmission error code upon failure.
    - `filename` - R/O Filename of the first file of the FAX transmission.
    - `filenames` - R/O Filenames of all of the files in the FAX transmission (comma separated).
    - `headerinfo` - R/W FAX header information.
    - `localstationid` - R/W Local Station Identification.
    - `minrate` - R/W Minimum transfer rate set before transmission.
    - `maxrate` - R/W Maximum transfer rate set before transmission.
    - `modem` - R/W Modem type (v17/v27/v29).
    - `gateway` - R/W T38 fax gateway, with optional fax activity timeout in seconds (yes,timeout/no)
    - `faxdetect` - R/W Enable FAX detect with optional timeout in seconds (yes,t38,cng,timeout/no)
    - `pages` - R/O Number of pages transferred.
    - `rate` - R/O Negotiated transmission rate.
    - `remotestationid` - R/O Remote Station Identification after transmission.
    - `resolution` - R/O Negotiated image resolution after transmission.
    - `sessionid` - R/O Session ID of the FAX transmission.
    - `status` - R/O Result Status of the FAX transmission.
    - `statusstr` - R/O Verbose Result Status of the FAX transmission.

See Also

- Asterisk 11 Application_ReceiveFax
- Asterisk 11 Application_SendFax

Import Version

This documentation was imported from Asterisk Version SVN-trunk-r370328

Asterisk 11 Function_FAXOPT_res_fax

FAXOPT() - [res_fax]

Synopsis

Gets/sets various pieces of information about a fax session.

Description

FAXOPT can be used to override the settings for a FAX session listed in `res_fax.conf`, it can also be used to retreive information about a FAX session that has finished eg. pages/status.

Syntax

```
FAXOPT(item)
```

Arguments

- `item`
  - `ecm` - R/W Error Correction Mode (ECM) enable with 'yes', disable with 'no'.
  - `error` - R/O FAX transmission error code upon failure.
  - `filename` - R/O Filename of the first file of the FAX transmission.
  - `filenames` - R/O Filenames of all of the files in the FAX transmission (comma separated).
  - `headerinfo` - R/W FAX header information.
  - `localstationid` - R/W Local Station Identification.
  - `minrate` - R/W Minimum transfer rate set before transmission.
  - `maxrate` - R/W Maximum transfer rate set before transmission.
  - `modem` - R/W Modem type (v17/v27/v29).
  - `gateway` - R/W T38 fax gateway, with optional fax activity timeout in seconds (yes[,timeout]/no)
  - `faxdetect` - R/W Enable FAX detect with optional timeout in seconds (yes,t38,cng[,timeout]/no)
  - `pages` - R/O Number of pages transferred.
  - `rate` - R/O Negotiated transmission rate.
  - `remotestationid` - R/O Remote Station Identification after transmission.
  - `resolution` - R/O Negotiated image resolution after transmission.
  - `sessionid` - R/O Session ID of the FAX transmission.
  - `status` - R/O Result Status of the FAX transmission.
  - `statusstr` - R/O Verbose Result Status of the FAX transmission.

See Also

- Asterisk 11 Application_ReceiveFax
- Asterisk 11 Application_SendFax

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_FEATURE

FEATURE()

Synopsis

Get or set a feature option on a channel.

Description

When this function is used as a read, it will get the current value of the specified feature option for this channel. It will be the value of this option configured in features.conf if a channel specific value has not been set. This function can also be used to set a channel specific value for the supported feature options.

Syntax

```
FEATURE(option_name)
```

Arguments

- `option_name` - The allowed values are:
    - `parkingtime` - Specified in seconds.

See Also

- Asterisk 11 Function_FEATUREMAP

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FEATUREMAP

FEATUREMAP()

Synopsis

Get or set a feature map to a given value on a specific channel.

Description

When this function is used as a read, it will get the current digit sequence mapped to the specified feature for this channel. This value will be the one configured in features.conf if a channel specific value has not been set. This function can also be used to set a channel specific value for a feature mapping.

Syntax

```
FEATUREMAP(feature_name)
```

Arguments

- `feature_name` - The allowed values are:
    - `atxfer` - Attended Transfer
    - `blindxfer` - Blind Transfer
    - `automon` - Auto Monitor
    - `disconnect` - Call Disconnect
    - `parkcall` - Park Call
    - `automixmon` - Auto MixMonitor

See Also

- Asterisk 11 Function_FEATURE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FIELDNUM

FIELDNUM()

Synopsis

Return the 1-based offset of a field in a list

Description

Search the variable named *varname* for the string *value* delimited by *delim* and return a 1-based offset as to its location. If not found or an error occured, return `0`.

The delimiter may be specified as a special or extended ASCII character, by encoding it. The characters `\n`, `\r`, and `\t` are all recognized as the newline, carriage return, and tab characters, respectively. Also, octal and hexadecimal specifications are recognized by the patterns `\0nnn` and `\xHH`, respectively. For example, if you wanted to encode a comma as the delimiter, you could use either `\054` or `\x2C`.

Example: If ${example} contains `ex-amp-le`, then ${FIELDNUM(example,-,amp)} returns 2.

Syntax

```
FIELDNUM(varname,delim,value)
```

Arguments

- `varname`
- `delim`
- `value`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FIELDQTY

FIELDQTY()

Synopsis

Count the fields with an arbitrary delimiter

Description

The delimiter may be specified as a special or extended ASCII character, by encoding it. The characters `\n`, `\r`, and `\t` are all recognized as the newline, carriage return, and tab characters, respectively. Also, octal and hexadecimal specifications are recognized by the patterns `\0nnn` and `\xHH`, respectively. For example, if you wanted to encode a comma as the delimiter, you could use either `\054` or `\x2C`.

Example: If ${example} contains `ex-amp-le`, then ${FIELDQTY(example,-)} returns 3.

Syntax

```
FIELDQTY(varname,delim)
```

Arguments

- `varname`
- `delim`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FILE

FILE()

Synopsis

Read or write text file.

Description

Read and write text file in character and line mode.

Examples:

Read mode (byte):

;reads the entire content of the file.

Set(foo=${FILE(/tmp/test.txt)})

;reads from the 11th byte to the end of the file (i.e. skips the first 10).

Set(foo=${FILE(/tmp/test.txt,10)})

;reads from the 11th to 20th byte in the file (i.e. skip the first 10, then read 10 bytes).

Set(foo=${FILE(/tmp/test.txt,10,10)})

Read mode (line):

; reads the 3rd line of the file.

Set(foo=${FILE(/tmp/test.txt,3,1,l)})

; reads the 3rd and 4th lines of the file.

Set(foo=${FILE(/tmp/test.txt,3,2,l)})

; reads from the third line to the end of the file.

Set(foo=${FILE(/tmp/test.txt,3,,l)})

; reads the last three lines of the file.

Set(foo=${FILE(/tmp/test.txt,-3,,l)})

; reads the 3rd line of a DOS-formatted file.

Set(foo=${FILE(/tmp/test.txt,3,1,l,d)})

Write mode (byte):

; truncate the file and write "bar"

Set(FILE(/tmp/test.txt)=bar)

; Append "bar"

Set(FILE(/tmp/test.txt,,,a)=bar)

; Replace the first byte with "bar" (replaces 1 character with 3)

Set(FILE(/tmp/test.txt,0,1)=bar)

; Replace 10 bytes beginning at the 21st byte of the file with "bar"

Set(FILE(/tmp/test.txt,20,10)=bar)

; Replace all bytes from the 21st with "bar"

Set(FILE(/tmp/test.txt,20)=bar)

; Insert "bar" after the 4th character

Set(FILE(/tmp/test.txt,4,0)=bar)

Write mode (line):

; Replace the first line of the file with "bar"

Set(FILE(/tmp/foo.txt,0,1,l)=bar)

; Replace the last line of the file with "bar"

Set(FILE(/tmp/foo.txt,-1,,l)=bar)

; Append "bar" to the file with a newline

Set(FILE(/tmp/foo.txt,,,al)=bar)

Syntax

```
FILE(filename,offset,length,options,format)
```

Arguments

- `filename`
- `offset` - Maybe specified as any number. If negative, *offset* specifies the number of bytes back from the end of the file.
- `length` - If specified, will limit the length of the data read to that size. If negative, trims *length* bytes from the end of the file.
- `options`
    - `l` - Line mode: offset and length are assumed to be measured in lines, instead of byte offsets.
    - `a` - In write mode only, the append option is used to append to the end of the file, instead of overwriting the existing file.
    - `d` - In write mode and line mode only, this option does not automatically append a newline string to the end of a value. This is useful for deleting lines, instead of setting them to blank.
- `format` - The *format* parameter may be used to delimit the type of line terminators in line mode.
    - `u` - Unix newline format.
    - `d` - DOS newline format.
    - `m` - Macintosh newline format.

See Also

- Asterisk 11 Function_FILE_COUNT_LINE
- Asterisk 11 Function_FILE_FORMAT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_FILE_COUNT_LINE

FILE_COUNT_LINE()

Synopsis

Obtains the number of lines of a text file.

Description

Returns the number of lines, or `-1` on error.

Syntax

```
FILE_COUNT_LINE(filename,format)
```

Arguments

- `filename`
- `format` - Format may be one of the following:
    - `u` - Unix newline format.
    - `d` - DOS newline format.
    - `m` - Macintosh newline format.

> **Note**
> If not specified, an attempt will be made to determine the newline format type.

See Also

- Asterisk 11 Function_FILE
- Asterisk 11 Function_FILE_FORMAT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FILE_FORMAT

FILE_FORMAT()

Synopsis

Return the newline format of a text file.

Description

Return the line terminator type:

'u' - Unix "\n" format

'd' - DOS "\r\n" format

'm' - Macintosh "\r" format

'x' - Cannot be determined

Syntax

```
FILE_FORMAT(filename)
```

Arguments

- `filename`

See Also

- Asterisk 11 Function_FILE
- Asterisk 11 Function_FILE_COUNT_LINE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_FILTER

FILTER()

Synopsis

Filter the string to include only the allowed characters

Description

Permits all characters listed in *allowed-chars*, filtering all others outs. In addition to literally listing the characters, you may also use ranges of characters (delimited by a `-`

Hexadecimal characters started with a `\x`(i.e. \x20)

Octal characters started with a `\0` (i.e. \040)

Also `\t`,`\n` and `\r` are recognized.

> **Note**
> If you want the `-` character it needs to be prefixed with a {{}}

Syntax

```
FILTER(allowed-chars,string)
```

Arguments

- `allowed-chars`
- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_FRAME_TRACE

FRAME_TRACE()

Synopsis

View internal ast_frames as they are read and written on a channel.

Description

Examples:

exten => 1,1,Set(FRAME_TRACE(white)=DTMF_BEGIN,DTMF_END); view only DTMF frames.

exten => 1,1,Set(FRAME_TRACE()=DTMF_BEGIN,DTMF_END); view only DTMF frames.

exten => 1,1,Set(FRAME_TRACE(black)=DTMF_BEGIN,DTMF_END); view everything except DTMF frames.

Syntax

```
FRAME_TRACE(filter list type)
```

Arguments

- `filter list type` - A filter can be applied to the trace to limit what frames are viewed. This filter can either be a `white` or `black` list of frame types. When no filter type is present, `white` is used. If no arguments are provided at all, all frames will be output.
  Below are the different types of frames that can be filtered.
  - `DTMF_BEGIN`
  - `DTMF_END`
  - `VOICE`
  - `VIDEO`
  - `CONTROL`
  - `NULL`
  - `IAX`
  - `TEXT`
  - `IMAGE`
  - `HTML`
  - `CNG`
  - `MODEM`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_GLOBAL

GLOBAL()

Synopsis

Gets or sets the global variable specified.

Description

Set or get the value of a global variable specified in *varname*

Syntax

```
GLOBAL(varname)
```

Arguments

- `varname` - Global variable name

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_GROUP

GROUP()

Synopsis

Gets or sets the channel group.

Description

category can be employed for more fine grained group management. Each channel can only be member of exactly one group per category.

Syntax

```
GROUP(category)
```

Arguments

- `category` - Category name.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_GROUP_COUNT

GROUP_COUNT()

Synopsis

Counts the number of channels in the specified group.

Description

Calculates the group count for the specified group, or uses the channel's current group if not specifed (and non-empty).

Syntax

```
GROUP_COUNT(groupname@category)
```

Arguments

- `groupname` - Group name.
- `category` - Category name

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_GROUP_LIST

GROUP_LIST()

Synopsis

Gets a list of the groups set on a channel.

Description

Gets a list of the groups set on a channel.

Syntax

```
GROUP_LIST()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_GROUP_MATCH_COUNT

GROUP_MATCH_COUNT()

Synopsis

Counts the number of channels in the groups matching the specified pattern.

Description

Calculates the group count for all groups that match the specified pattern. Note: category matching is applied after matching based on group. Uses standard regular expression matching on both (see regex(7)).

Syntax

```
GROUP_MATCH_COUNT(groupmatch@category)
```

Arguments

- `groupmatch` - A standard regular expression used to match a group name.
- `category` - A standard regular expression used to match a category name.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_HANGUPCAUSE

HANGUPCAUSE()

Synopsis

Gets per-channel hangupcause information from the channel.

Description

Gets technology-specific or translated Asterisk cause code information from the channel for the specified channel that resulted from a dial.

Syntax

```
HANGUPCAUSE(channel,type)
```

Arguments

- `channel` - The name of the channel for which to retreive cause information.
- `type` - Parameter describing which type of information is requested. Types are:
  - `tech` - Technology-specific cause information
  - `ast` - Translated Asterisk cause code

See Also

- Asterisk 11 Function_HANGUPCAUSE_KEYS
- Asterisk 11 Application_HangupCauseClear

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_HANGUPCAUSE_KEYS

HANGUPCAUSE_KEYS()

Synopsis

Gets the list of channels for which hangup causes are available.

Description

Returns a comma-separated list of channel names to be used with the HANGUPCAUSE function.

See Also

- Asterisk 11 Function_HANGUPCAUSE
- Asterisk 11 Application_HangupCauseClear

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_HASH

HASH()

Synopsis

Implementation of a dialplan associative array

Description

In two arguments mode, gets and sets values to corresponding keys within a named associative array. The single-argument mode will only work when assigned to from a function defined by func_odbc

Syntax

```
HASH(hashname,hashkey)
```

Arguments

- `hashname`
- `hashkey`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_HASHKEYS

HASHKEYS()

Synopsis

Retrieve the keys of the HASH() function.

Description

Returns a comma-delimited list of the current keys of the associative array defined by the HASH() function. Note that if you iterate over the keys of the result, adding keys during iteration will cause the result of the HASHKEYS() function to change.

Syntax

```
HASHKEYS(hashname)
```

Arguments

- `hashname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_HINT

HINT()

Synopsis

Get the devices set for a dialplan hint.

Description

The HINT function can be used to retrieve the list of devices that are mapped to a dialplan hint. For example:

NoOp(Hint for Extension 1234 is ${HINT(1234)})

Syntax

```
HINT(extension,options)
```

Arguments

- `extension`
    - `extension`
    - `context`
- `options`
    - n - Retrieve name on the hint instead of list of devices.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_IAXPEER

IAXPEER()

Synopsis

Gets IAX peer information.

Description

Gets information associated with the specified IAX2 peer.

Syntax

```
IAXPEER(peername,item)
```

Arguments

- `peername`
    - `CURRENTCHANNEL` - If *peername* is specified to this value, return the IP address of the endpoint of the current channel
- `item` - If *peername* is specified, valid items are:
    - `ip` - (default) The IP address.
    - `status` - The peer's status (if `qualify=yes`)
    - `mailbox` - The configured mailbox.
    - `context` - The configured context.
    - `expire` - The epoch time of the next expire.
    - `dynamic` - Is it dynamic? (yes/no).
    - `callerid_name` - The configured Caller ID name.
    - `callerid_num` - The configured Caller ID number.
    - `codecs` - The configured codecs.
    - `codecx` - Preferred codec index number *x* (beginning with `0`)

See Also

- Asterisk 11 Function_SIPPEER

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_IAXVAR

IAXVAR()

Synopsis

Sets or retrieves a remote variable.

Description

Gets or sets a variable that is sent to a remote IAX2 peer during call setup.

Syntax

```
IAXVAR(varname)
```

Arguments

- `varname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ICONV

ICONV()

Synopsis

Converts charsets of strings.

Description

Converts string from *in-charset* into *out-charset*. For available charsets, use `iconv -l` on your shell command line.

> **Note**
> Due to limitations within the API, ICONV will not currently work with charsets with embedded NULLs. If found, the string will terminate.

Syntax

```
ICONV(in-charset,out-charset,string)
```

Arguments

- `in-charset` - Input charset
- `out-charset` - Output charset
- `string` - String to convert, from *in-charset* to *out-charset*

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_IF

IF()

Synopsis

Check for an expresion.

Description

Returns the data following `?` if true, else the data following `:`

Syntax

```
IF(expresion?retvalue)
```

Arguments

- `expresion`
- `retvalue`
  - `true`
  - `false`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_IFMODULE

IFMODULE()

Synopsis

Checks if an Asterisk module is loaded in memory.

Description

Checks if a module is loaded. Use the full module name as shown by the list in `module list`. Returns `1` if module exists in memory, otherwise `0`

Syntax

```
IFMODULE(modulename.so)
```

Arguments

- `modulename.so` - Module name complete with `.so`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_IFTIME

IFTIME()

Synopsis

Temporal Conditional.

Description

Returns the data following `?` if true, else the data following `:`

Syntax

```
IFTIME(timespec?retvalue)
```

Arguments

- `timespec`
- `retvalue`
    - `true`
    - `false`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_IMPORT

IMPORT()

Synopsis

Retrieve the value of a variable from another channel.

Description

Syntax

```
IMPORT(channel,variable)
```

Arguments

- `channel`
- `variable`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_INC

INC()

Synopsis

Increments the value of a variable, while returning the updated value to the dialplan

Description

Increments the value of a variable, while returning the updated value to the dialplan

Example: INC(MyVAR) - Increments MyVar

Note: INC(${MyVAR}) - Is wrong, as INC expects the variable name, not its value

Syntax

```
INC(variable)
```

Arguments

- `variable` - The variable name to be manipulated, without the braces.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ISNULL

ISNULL()

Synopsis

Check if a value is NULL.

Description

Returns `1` if NULL or `0` otherwise.

Syntax

```
ISNULL(data)
```

Arguments

- `data`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_JABBER_RECEIVE

Moved to Asterisk 11 Function_JABBER_RECEIVE_res_xmpp.

Asterisk 11 Function_JABBER_RECEIVE_res_jabber

JABBER_RECEIVE() - [res_jabber]

Synopsis

Reads XMPP messages.

Description

Receives a text message on the given *account* from the buddy identified by *jid* and returns the contents.

Example: ${JABBER_RECEIVE(asterisk,bob@domain.com)} returns an XMPP message sent from *bob@domain.com* (or nothing in case of a time out), to the *asterisk* XMPP account configured in jabber.conf.

Syntax

```
JABBER_RECEIVE(account,jid,timeout)
```

Arguments

- `account` - The local named account to listen on (specified in jabber.conf)
- `jid` - Jabber ID of the buddy to receive message from. It can be a bare JID (username@domain) or a full JID (username@domain/resource).
- `timeout` - In seconds, defaults to `20`.

See Also

- Asterisk 11 Function_JABBER_STATUS_res_jabber
- Asterisk 11 Application_JabberSend_res_jabber

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_JABBER_RECEIVE_res_xmpp

JABBER_RECEIVE() - [res_xmpp]

Synopsis

Reads XMPP messages.

Description

Receives a text message on the given *account* from the buddy identified by *jid* and returns the contents.

Example: ${JABBER_RECEIVE(asterisk,bob@domain.com)} returns an XMPP message sent from *bob@domain.com* (or nothing in case of a time out), to the *asterisk* XMPP account configured in xmpp.conf.

Syntax

```
JABBER_RECEIVE(account,jid,timeout)
```

Arguments

- `account` - The local named account to listen on (specified in xmpp.conf)
- `jid` - Jabber ID of the buddy to receive message from. It can be a bare JID (username@domain) or a full JID (username@domain/resource).
- `timeout` - In seconds, defaults to `20`.

See Also

- Asterisk 11 Function_JABBER_STATUS_res_xmpp
- Asterisk 11 Application_JabberSend_res_xmpp

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_JABBER_STATUS

Moved to Asterisk 11 Function_JABBER_STATUS_res_xmpp.

Asterisk 11 Function_JABBER_STATUS_res_jabber

JABBER_STATUS() - [res_jabber]

Synopsis

Retrieves a buddy's status.

Description

Retrieves the numeric status associated with the buddy identified by *jid*. If the buddy does not exist in the buddylist, returns 7.

Status will be 1-7.

1=Online, 2=Chatty, 3=Away, 4=XAway, 5=DND, 6=Offline

If not in roster variable will be set to 7.

Example: ${JABBER_STATUS(asterisk,bob@domain.com)} returns 1 if *bob@domain.com* is online. *asterisk* is the associated XMPP account configured in jabber.conf.

Syntax

```
JABBER_STATUS(account,jid)
```

Arguments

- `account` - The local named account to listen on (specified in jabber.conf)
- `jid` - Jabber ID of the buddy to receive message from. It can be a bare JID (username@domain) or a full JID (username@domain/resource).

See Also

- Asterisk 11 Function_JABBER_RECEIVE_res_jabber
- Asterisk 11 Application_JabberSend_res_jabber

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_JABBER_STATUS_res_xmpp

JABBER_STATUS() - [res_xmpp]

Synopsis

Retrieves a buddy's status.

Description

Retrieves the numeric status associated with the buddy identified by *jid*. If the buddy does not exist in the buddylist, returns 7.

Status will be 1-7.

1=Online, 2=Chatty, 3=Away, 4=XAway, 5=DND, 6=Offline

If not in roster variable will be set to 7.

Example: ${JABBER_STATUS(asterisk,bob@domain.com)} returns 1 if *bob@domain.com* is online. *asterisk* is the associated XMPP account configured in xmpp.conf.

Syntax

```
JABBER_STATUS(account,jid)
```

Arguments

- `account` - The local named account to listen on (specified in xmpp.conf)
- `jid` - Jabber ID of the buddy to receive message from. It can be a bare JID (username@domain) or a full JID (username@domain/resource).

See Also

- Asterisk 11 Function_JABBER_RECEIVE_res_xmpp
- Asterisk 11 Application_JabberSend_res_xmpp

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_JITTERBUFFER

JITTERBUFFER()

Synopsis

Add a Jitterbuffer to the Read side of the channel. This dejitters the audio stream before it reaches the Asterisk core. This is a write only function.

Description

max_size: Defaults to 200 ms

Length in milliseconds of buffer.

resync_threshold: Defaults to 1000ms

The length in milliseconds over which a timestamp difference will result in resyncing the jitterbuffer.

target_extra: Defaults to 40ms

This option only affects the adaptive jitterbuffer. It represents the amount time in milliseconds by which the new jitter buffer will pad its size.

Examples:

exten => 1,1,Set(JITTERBUFFER(fixed)=default);Fixed with defaults.

exten => 1,1,Set(JITTERBUFFER(fixed)=200);Fixed with max size 200ms, default resync threshold and target extra.

exten => 1,1,Set(JITTERBUFFER(fixed)=200,1500);Fixed with max size 200ms resync threshold 1500.

exten => 1,1,Set(JITTERBUFFER(adaptive)=default);Adaptive with defaults.

exten => 1,1,Set(JITTERBUFFER(adaptive)=200,,60);Adaptive with max size 200ms, default resync threshold and 40ms target extra.

Syntax

```
JITTERBUFFER(jitterbuffer type)
```

Arguments

- `jitterbuffer type` - Jitterbuffer type can be either `fixed` or `adaptive`.
  Used as follows.
  Set(JITTERBUFFER(type)=max_size[,resync_threshold[,target_extra]])
  Set(JITTERBUFFER(type)=default)

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_KEYPADHASH

KEYPADHASH()

Synopsis

Hash the letters in string into equivalent keypad numbers.

Description

Example: ${KEYPADHASH(Les)} returns "537"

Syntax

```
KEYPADHASH(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_LEN

LEN()

Synopsis

Return the length of the string given.

Description

Example: ${LEN(example)} returns 7

Syntax

```
LEN(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_LISTFILTER

LISTFILTER()

Synopsis

Remove an item from a list, by name.

Description

Remove *value* from the list contained in the *varname* variable, where the list delimiter is specified by the *delim* parameter. This is very useful for removing a single channel name from a list of channels, for example.

Syntax

```
LISTFILTER(varname,delim,value)
```

Arguments

- `varname`
- `delim`
- `value`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_LOCAL

LOCAL()

Synopsis

Manage variables local to the gosub stack frame.

Description

Read and write a variable local to the gosub stack frame, once we Return() it will be lost (or it will go back to whatever value it had before the Gosub()).

Syntax

```
LOCAL(varname)
```

Arguments

- `varname`

See Also

- Asterisk 11 Application_Gosub
- Asterisk 11 Application_GosubIf
- Asterisk 11 Application_Return

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_LOCAL_PEEK

LOCAL_PEEK()

Synopsis

Retrieve variables hidden by the local gosub stack frame.

Description

Read a variable *varname* hidden by *n* levels of gosub stack frames. Note that ${LOCAL_PEEK(0,foo)} is the same as `foo`, since the value of *n* peeks under 0 levels of stack frames; in other words, 0 is the current level. If *n* exceeds the available number of stack frames, then an empty string is returned.

Syntax

```
LOCAL_PEEK(n,varname)
```

Arguments

- `n`
- `varname`

See Also

- Asterisk 11 Application_Gosub
- Asterisk 11 Application_GosubIf
- Asterisk 11 Application_Return

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_LOCK

LOCK()

Synopsis

Attempt to obtain a named mutex.

Description

Attempts to grab a named lock exclusively, and prevents other channels from obtaining the same lock. LOCK will wait for the lock to become available. Returns `1` if the lock was obtained or `0` on error.

> **Note**
> To avoid the possibility of a deadlock, LOCK will only attempt to obtain the lock for 3 seconds if the channel already has another lock.

Syntax

```
LOCK(lockname)
```

Arguments

- `lockname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MAILBOX_EXISTS

MAILBOX_EXISTS()

Synopsis

Tell if a mailbox is configured.

Description

> **Note**
> DEPRECATED. Use VM_INFO(mailbox[@context],exists) instead.

Returns a boolean of whether the corresponding *mailbox* exists. If *context* is not specified, defaults to the `default` context.

Syntax

```
MAILBOX_EXISTS(mailbox@context)
```

Arguments

- `mailbox`
- `context`

See Also

- Asterisk 11 Function_VM_INFO

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_MASTER_CHANNEL

MASTER_CHANNEL()

Synopsis

Gets or sets variables on the master channel

Description

Allows access to the channel which created the current channel, if any. If the channel is already a master channel, then accesses local channel variables.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MATH

MATH()

Synopsis

Performs Mathematical Functions.

Description

Performs mathematical functions based on two parameters and an operator. The returned value type is *type*

Example: Set(i=${MATH(123%16,int)}) - sets var i=11

Syntax

```
MATH(expression,type)
```

Arguments

- `expression` - Is of the form: *number1opnumber2* where the possible values for *op* are:
  +,-,/,*,%,<<,>>,^,AND,OR,XOR,<,>,<=,>=,== (and behave as their C equivalents)
- `type` - Wanted type of result:
  f, float - float(default)
  i, int - integer
  h, hex - hex
  c, char - char

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MD5

MD5()

Synopsis

Computes an MD5 digest.

Description

Computes an MD5 digest.

Syntax

```
MD5(data)
```

Arguments

- `data`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MEETME_INFO

MEETME_INFO()

Synopsis

Query a given conference of various properties.

Description

Syntax

```
MEETME_INFO(keyword,confno)
```

Arguments

- `keyword` - Options:
    - `lock` - Boolean of whether the corresponding conference is locked.
    - `parties` - Number of parties in a given conference
    - `activity` - Duration of conference in seconds.
    - `dynamic` - Boolean of whether the corresponding conference is dynamic.
- `confno` - Conference number to retrieve information from.

See Also

- Asterisk 11 Application_MeetMe
- Asterisk 11 Application_MeetMeCount
- Asterisk 11 Application_MeetMeAdmin
- Asterisk 11 Application_MeetMeChannelAdmin

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MESSAGE

MESSAGE()

Synopsis

Create a message or read fields from a message.

Description

This function will read from or write a value to a text message. It is used both to read the data out of an incoming message, as well as modify or create a message that will be sent outbound.

Syntax

```
MESSAGE(argument)
```

Arguments

- `argument` - Field of the message to get or set.
  - `to` - Read-only. The destination of the message. When processing an incoming message, this will be set to the destination listed as the recipient of the message that was received by Asterisk.
  - `from` - Read-only. The source of the message. When processing an incoming message, this will be set to the source of the message.
  - `custom_data` - Write-only. Mark or unmark all message headers for an outgoing message. The following values can be set:
    - `mark_all_outbound` - Mark all headers for an outgoing message.
    - `clear_all_outbound` - Unmark all headers for an outgoing message.
  - `body` - Read/Write. The message body. When processing an incoming message, this includes the body of the message that Asterisk received. When MessageSend() is executed, the contents of this field are used as the body of the outgoing message. The body will always be UTF-8.

See Also

- Asterisk 11 Application_MessageSend

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MESSAGE_DATA

MESSAGE_DATA()

Synopsis

Read or write custom data attached to a message.

Description

This function will read from or write a value to a text message. It is used both to read the data out of an incoming message, as well as modify a message that will be sent outbound.

> **Note**
> If you want to set an outbound message to carry data in the current message, do Set(MESSAGE_DATA(*key*)=${MESSAGE_DATA(*key*)}).

Syntax

```
MESSAGE_DATA(argument)
```

Arguments

- `argument` - Field of the message to get or set.

See Also

- Asterisk 11 Application_MessageSend

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MINIVMACCOUNT

MINIVMACCOUNT()

Synopsis

Gets MiniVoicemail account information.

Description

Syntax

```
MINIVMACCOUNT(account:item)
```

Arguments

- `account`
- `item` - Valid items are:
    - `path` - Path to account mailbox (if account exists, otherwise temporary mailbox).
    - `hasaccount` - 1 is static Minivm account exists, 0 otherwise.
    - `fullname` - Full name of account owner.
    - `email` - Email address used for account.
    - `etemplate` - Email template for account (default template if none is configured).
    - `ptemplate` - Pager template for account (default template if none is configured).
    - `accountcode` - Account code for the voicemail account.
    - `pincode` - Pin code for voicemail account.
    - `timezone` - Time zone for voicemail account.
    - `language` - Language for voicemail account.
    - `<channel variable name>` - Channel variable value (set in configuration for account).

See Also

- Asterisk 11 Application_MinivmRecord
- Asterisk 11 Application_MinivmGreet
- Asterisk 11 Application_MinivmNotify
- Asterisk 11 Application_MinivmDelete
- Asterisk 11 Application_MinivmAccMess
- Asterisk 11 Application_MinivmMWI
- Asterisk 11 Function_MINIVMCOUNTER

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MINIVMCOUNTER

MINIVMCOUNTER()

Synopsis

Reads or sets counters for MiniVoicemail message.

Description

The operation is atomic and the counter is locked while changing the value. The counters are stored as text files in the minivm account directories. It might be better to use realtime functions if you are using a database to operate your Asterisk.

Syntax

```
MINIVMCOUNTER(account:name:operand)
```

Arguments

- `account` - If account is given and it exists, the counter is specific for the account.
  If account is a domain and the domain directory exists, counters are specific for a domain.
- `name` - The name of the counter is a string, up to 10 characters.
- `operand` - The counters never goes below zero. Valid operands for changing the value of a counter when assigning a value are:
    - `i` - Increment by value.
    - `d` - Decrement by value.
    - `s` - Set to value.

See Also

- Asterisk 11 Application_MinivmRecord
- Asterisk 11 Application_MinivmGreet
- Asterisk 11 Application_MinivmNotify
- Asterisk 11 Application_MinivmDelete
- Asterisk 11 Application_MinivmAccMess
- Asterisk 11 Application_MinivmMWI
- Asterisk 11 Function_MINIVMACCOUNT

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_MUTEAUDIO

MUTEAUDIO()

Synopsis

Muting audio streams in the channel

Description

The MUTEAUDIO function can be used to mute inbound (to the PBX) or outbound audio in a call.

Examples:

MUTEAUDIO(in)=on

MUTEAUDIO(in)=off

Syntax

```
MUTEAUDIO(direction)
```

Arguments

- `direction` - Must be one of
    - `in` - Inbound stream (to the PBX)
    - `out` - Outbound stream (from the PBX)
    - `all` - Both streams

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ODBC

ODBC()

Synopsis

Controls ODBC transaction properties.

Description

The ODBC() function allows setting several properties to influence how a connected database processes transactions.

Syntax

```
ODBC(property[,argument])
```

Arguments

- `property`
    - `transaction` - Gets or sets the active transaction ID. If set, and the transaction ID does not exist and a *database name* is specified as an argument, it will be created.
    - `forcecommit` - Controls whether a transaction will be automatically committed when the channel hangs up. Defaults to false. If a *transaction ID* is specified in the optional argument, the property will be applied to that ID, otherwise to the current active ID.
    - `isolation` - Controls the data isolation on uncommitted transactions. May be one of the following: `read_committed`, `read_uncommitted`, `repeatable_read`, or `serializable`. Defaults to the database setting in `res_odbc.conf` or `read_committed` if not specified. If a *transaction ID* is specified as an optional argument, it will be applied to that ID, otherwise the current active ID.
- `argument`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_ODBC_FETCH

ODBC_FETCH()

Synopsis

Fetch a row from a multirow query.

Description

For queries which are marked as mode=multirow, the original query returns a *result-id* from which results may be fetched. This function implements the actual fetch of the results.

This also sets `ODBC_FETCH_STATUS`.

- `ODBC_FETCH_STATUS`
    - SUCESS - If rows are available.
    - FAILURE - If no rows are available.

Syntax

```
ODBC_FETCH(result-id)
```

Arguments

- `result-id`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_PASSTHRU

PASSTHRU()

Synopsis

Pass the given argument back as a value.

Description

Literally returns the given *string*. The intent is to permit other dialplan functions which take a variable name as an argument to be able to take a literal string, instead.

Note
The functions which take a variable name need to be passed var and not ${var}. Similarly, use PASSTHRU() and not ${PASSTHRU()}.

Example: ${CHANNEL} contains SIP/321-1

${CUT(PASSTHRU(${CUT(CHANNEL,-,1)}),/,2)}) will return 321

Syntax

```
PASSTHRU([string])
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r394345

Asterisk 11 Function_PITCH_SHIFT

PITCH_SHIFT()

Synopsis

Pitch shift both tx and rx audio streams on a channel.

Description

Examples:

exten => 1,1,Set(PITCH_SHIFT(tx)=highest); raises pitch an octave

exten => 1,1,Set(PITCH_SHIFT(rx)=higher) ; raises pitch more

exten => 1,1,Set(PITCH_SHIFT(both)=high) ; raises pitch

exten => 1,1,Set(PITCH_SHIFT(rx)=low) ; lowers pitch

exten => 1,1,Set(PITCH_SHIFT(tx)=lower) ; lowers pitch more

exten => 1,1,Set(PITCH_SHIFT(both)=lowest) ; lowers pitch an octave

exten => 1,1,Set(PITCH_SHIFT(rx)=0.8) ; lowers pitch

exten => 1,1,Set(PITCH_SHIFT(tx)=1.5) ; raises pitch

Syntax

```
PITCH_SHIFT(channel direction)
```

Arguments

- `channel direction` - Direction can be either `rx`, `tx`, or `both`. The direction can either be set to a valid floating point number between 0.1 and 4.0 or one of the enum values listed below. A value of 1.0 has no effect. Greater than 1 raises the pitch. Lower than 1 lowers the pitch.
  The pitch amount can also be set by the following values
  - `highest`
  - `higher`
  - `high`
  - `low`
  - `lower`
  - `lowest`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_POP

POP()

Synopsis

Removes and returns the last item off of a variable containing delimited text

Description

Example:

exten => s,1,Set(array=one,two,three)

exten => s,n,While($["${SET(var=${POP(array)})}" != ""])

exten => s,n,NoOp(var is ${var})

exten => s,n,EndWhile

This would iterate over each value in array, right to left, and would result in NoOp(var is three), NoOp(var is two), and NoOp(var is one) being executed.

Syntax

```
POP(varname[,delimiter])
```

Arguments

- `varname`
- `delimiter`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_PP_EACH_EXTENSION

PP_EACH_EXTENSION()

Synopsis

Execute specified template for each extension.

Description

Output the specified template for each extension associated with the specified MAC address.

Syntax

```
PP_EACH_EXTENSION(mac,template)
```

Arguments

- `mac`
- `template`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_PP_EACH_USER

PP_EACH_USER()

Synopsis

Generate a string for each phoneprov user.

Description

Pass in a string, with phoneprov variables you want substituted in the format of %{VARNAME}, and you will get the string rendered for each user in phoneprov excluding ones with MAC address *exclude_mac*. Probably not useful outside of res_phoneprov.

Example: ${PP_EACH_USER(<item><fn>%{DISPLAY_NAME}</fn></item>|${MAC})

Syntax

```
PP_EACH_USER(string,exclude_mac)
```

Arguments

- `string`
- `exclude_mac`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_PRESENCE_STATE

PRESENCE_STATE()

Synopsis

Get or Set a presence state.

Description

The PRESENCE_STATE function can be used to retrieve the presence from any presence provider. For example:

NoOp(SIP/mypeer has presence ${PRESENCE_STATE(SIP/mypeer,value)})

NoOp(Conference number 1234 has presence message ${PRESENCE_STATE(MeetMe:1234,message)})

The PRESENCE_STATE function can also be used to set custom presence state from the dialplan. The `CustomPresence:` prefix must be used. For example:

Set(PRESENCE_STATE(CustomPresence:lamp1)=away,temporary,Out to lunch)

Set(PRESENCE_STATE(CustomPresence:lamp2)=dnd,,Trying to get work done)

Set(PRESENCE_STATE(CustomPresence:lamp3)=xa,T24gdmFjYXRpb24=,,e)

Set(BASE64_LAMP3_PRESENCE=${PRESENCE_STATE(CustomPresence:lamp3,subtype,e)})

You can subscribe to the status of a custom presence state using a hint in the dialplan:

exten => 1234,hint,CustomPresence:lamp1

The possible values for both uses of this function are:

not_set | unavailable | available | away | xa | chat | dnd

Syntax

```
PRESENCE_STATE(provider,field[,options])
```

Arguments

- `provider` - The provider of the presence, such as `CustomPresence`
- `field` - Which field of the presence state information is wanted.
    - `value` - The current presence, such as `away`
    - `subtype` - Further information about the current presence
    - `message` - A custom message that may indicate further details about the presence
- `options`
    - `e` - On Write - Use this option when the subtype and message provided are Base64 encoded. On Read - Retrieves message/subtype in Base64 encoded form.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r381702

Asterisk 11 Function_PUSH

PUSH()

Synopsis

Appends one or more values to the end of a variable containing delimited text

Description

Example: Set(PUSH(array)=one,two,three) would append one, two, and three to the end of the values stored in the variable "array".

Syntax

```
PUSH(varname[,delimiter])
```

Arguments

- `varname`
- `delimiter`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_EXISTS

QUEUE_EXISTS()

Synopsis

Check if a named queue exists on this server

Description

Returns 1 if the specified queue exists, 0 if it does not

Syntax

```
QUEUE_EXISTS(queuename)
```

Arguments

- `queuename`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_MEMBER

QUEUE_MEMBER()

Synopsis

Count number of members answering a queue.

Description

Allows access to queue counts [R] and member information [R/W].

queuename is required for all operations *interface* is required for all member operations.

Syntax

```
QUEUE_MEMBER(queuename,option[,interface])
```

Arguments

- `queuename`
- `option`
    - `logged` - Returns the number of logged-in members for the specified queue.
    - `free` - Returns the number of logged-in members for the specified queue that either can take calls or are currently wrapping up after a previous call.
    - `ready` - Returns the number of logged-in members for the specified queue that are immediately available to answer a call.
    - `count` - Returns the total number of members for the specified queue.
    - `penalty` - Gets or sets queue member penalty.
    - `paused` - Gets or sets queue member paused status.
    - `ringinuse` - Gets or sets queue member ringinuse.
- `interface`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_MEMBER_COUNT

QUEUE_MEMBER_COUNT()

Synopsis

Count number of members answering a queue.

Description

Returns the number of members currently associated with the specified *queuename*.

> **Warning**
> This function has been deprecated in favor of the `QUEUE_MEMBER()` function

Syntax

```
QUEUE_MEMBER_COUNT(queuename)
```

Arguments

- `queuename`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_MEMBER_LIST

QUEUE_MEMBER_LIST()

Synopsis

Returns a list of interfaces on a queue.

Description

Returns a comma-separated list of members associated with the specified *queuename*.

Syntax

```
QUEUE_MEMBER_LIST(queuename)
```

Arguments

- `queuename`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_MEMBER_PENALTY

QUEUE_MEMBER_PENALTY()

Synopsis

Gets or sets queue members penalty.

Description

Gets or sets queue members penalty.

> **Warning**
> This function has been deprecated in favor of the `QUEUE_MEMBER()` function

Syntax

```
QUEUE_MEMBER_PENALTY(queuename,interface)
```

Arguments

- `queuename`
- `interface`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_VARIABLES

QUEUE_VARIABLES()

Synopsis

Return Queue information in variables.

Description

Makes the following queue variables available.

Returns `0` if queue is found and setqueuevar is defined, `-1` otherwise.

Syntax

```
QUEUE_VARIABLES(queuename)
```

Arguments

- `queuename`
    - `QUEUEMAX` - Maxmimum number of calls allowed.
    - `QUEUESTRATEGY` - The strategy of the queue.
    - `QUEUECALLS` - Number of calls currently in the queue.
    - `QUEUEHOLDTIME` - Current average hold time.
    - `QUEUECOMPLETED` - Number of completed calls for the queue.
    - `QUEUEABANDONED` - Number of abandoned calls.
    - `QUEUESRVLEVEL` - Queue service level.
    - `QUEUESRVLEVELPERF` - Current service level performance.

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUEUE_WAITING_COUNT

QUEUE_WAITING_COUNT()

Synopsis

Count number of calls currently waiting in a queue.

Description

Returns the number of callers currently waiting in the specified *queuename*.

Syntax

```
QUEUE_WAITING_COUNT(queuename)
```

Arguments

- `queuename`

See Also

- Asterisk 11 Application_Queue
- Asterisk 11 Application_QueueLog
- Asterisk 11 Application_AddQueueMember
- Asterisk 11 Application_RemoveQueueMember
- Asterisk 11 Application_PauseQueueMember
- Asterisk 11 Application_UnpauseQueueMember
- Asterisk 11 Function_QUEUE_VARIABLES
- Asterisk 11 Function_QUEUE_MEMBER
- Asterisk 11 Function_QUEUE_MEMBER_COUNT
- Asterisk 11 Function_QUEUE_EXISTS
- Asterisk 11 Function_QUEUE_WAITING_COUNT
- Asterisk 11 Function_QUEUE_MEMBER_LIST
- Asterisk 11 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_QUOTE

QUOTE()

Synopsis

Quotes a given string, escaping embedded quotes as necessary

Description

Example: ${QUOTE(ab"c"de)} will return "abcde"

Syntax

```
QUOTE(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_RAND

RAND()

Synopsis

Choose a random number in a range.

Description

Choose a random number between *min* and *max*. *min* defaults to `0`, if not specified, while *max* defaults to `RAND_MAX` (2147483647 on many systems).

Example: Set(junky=${RAND(1,8)}); Sets junky to a random number between 1 and 8, inclusive.

Syntax

```
RAND(min,max)
```

Arguments

- `min`
- `max`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REALTIME

REALTIME()

Synopsis

RealTime Read/Write Functions.

Description

This function will read or write values from/to a RealTime repository. REALTIME(....) will read names/values from the repository, and REALTIME(....)= will write a new value/field to the repository. On a read, this function returns a delimited text string. The name/value pairs are delimited by *delim1*, and the name and value are delimited between each other with delim2. If there is no match, NULL will be returned by the function. On a write, this function will always return NULL.

Syntax

```
REALTIME(family,fieldmatch,matchvalue,delim1|field,delim2)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`
- `delim1|field` - Use *delim1* with *delim2* on read and *field* without *delim2* on write
  If we are reading and *delim1* is not specified, defaults to `,`
- `delim2` - Parameter only used when reading, if not specified defaults to =

See Also

- Asterisk 11 Function_REALTIME_STORE
- Asterisk 11 Function_REALTIME_DESTROY
- Asterisk 11 Function_REALTIME_FIELD
- Asterisk 11 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REALTIME_DESTROY

REALTIME_DESTROY()

Synopsis

RealTime Destroy Function.

Description

This function acts in the same way as REALTIME(....) does, except that it destroys the matched record in the RT engine.

Syntax

```
REALTIME_DESTROY(family,fieldmatch,matchvalue,delim1,delim2)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`
- `delim1`
- `delim2`

See Also

- Asterisk 11 Function_REALTIME
- Asterisk 11 Function_REALTIME_STORE
- Asterisk 11 Function_REALTIME_FIELD
- Asterisk 11 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REALTIME_FIELD

REALTIME_FIELD()

Synopsis

RealTime query function.

Description

This function retrieves a single item, *fieldname* from the RT engine, where *fieldmatch* contains the value *matchvalue*. When written to, the REALTIME_FIELD() function performs identically to the REALTIME() function.

Syntax

```
REALTIME_FIELD(family,fieldmatch,matchvalue,fieldname)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`
- `fieldname`

See Also

- Asterisk 11 Function_REALTIME
- Asterisk 11 Function_REALTIME_STORE
- Asterisk 11 Function_REALTIME_DESTROY
- Asterisk 11 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REALTIME_HASH

REALTIME_HASH()

Synopsis

RealTime query function.

Description

This function retrieves a single record from the RT engine, where *fieldmatch* contains the value *matchvalue* and formats the output suitably, such that it can be assigned to the HASH() function. The HASH() function then provides a suitable method for retrieving each field value of the record.

Syntax

```
REALTIME_HASH(family,fieldmatch,matchvalue)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`

See Also

- Asterisk 11 Function_REALTIME
- Asterisk 11 Function_REALTIME_STORE
- Asterisk 11 Function_REALTIME_DESTROY
- Asterisk 11 Function_REALTIME_FIELD

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REALTIME_STORE

REALTIME_STORE()

Synopsis

RealTime Store Function.

Description

This function will insert a new set of values into the RealTime repository. If RT engine provides an unique ID of the stored record, REALTIME_STORE(...)=.. creates channel variable named RTSTOREID, which contains value of unique ID. Currently, a maximum of 30 field/value pairs is supported.

Syntax

```
REALTIME_STORE(family,field1,fieldN[,...],field30)
```

Arguments

- `family`
- `field1`
- `fieldN`
- `field30`

See Also

- Asterisk 11 Function_REALTIME
- Asterisk 11 Function_REALTIME_DESTROY
- Asterisk 11 Function_REALTIME_FIELD
- Asterisk 11 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REDIRECTING

REDIRECTING()

Synopsis

Gets or sets Redirecting data on the channel.

Description

Gets or sets Redirecting data on the channel.

The allowable values for the *reason* and *orig-reason* fields are the following:

- `unknown` - Unknown
- `cfb` - Call Forwarding Busy
- `cfnr` - Call Forwarding No Reply
- `unavailable` - Callee is Unavailable
- `time_of_day` - Time of Day
- `dnd` - Do Not Disturb
- `deflection` - Call Deflection
- `follow_me` - Follow Me
- `out_of_order` - Called DTE Out-Of-Order
- `away` - Callee is Away
- `cf_dte` - Call Forwarding By The Called DTE
- `cfu` - Call Forwarding Unconditional
  The allowable values for the *xxx-name-charset* field are the following:

- `unknown` - Unknown
- `iso8859-1` - ISO8859-1
- `withdrawn` - Withdrawn
- `iso8859-2` - ISO8859-2
- `iso8859-3` - ISO8859-3
- `iso8859-4` - ISO8859-4
- `iso8859-5` - ISO8859-5
- `iso8859-7` - ISO8859-7
- `bmp` - ISO10646 Bmp String
- `utf8` - ISO10646 UTF-8 String

Syntax

```
REDIRECTING(datatype,i)
```

Arguments

- `datatype` - The allowable datatypes are:
  - `orig-all`
  - `orig-name`
  - `orig-name-valid`
  - `orig-name-charset`
  - `orig-name-pres`
  - `orig-num`
  - `orig-num-valid`
  - `orig-num-plan`
  - `orig-num-pres`
  - `orig-subaddr`
  - `orig-subaddr-valid`
  - `orig-subaddr-type`
  - `orig-subaddr-odd`
  - `orig-tag`

- orig-reason
- from-all
- from-name
- from-name-valid
- from-name-charset
- from-name-pres
- from-num
- from-num-valid
- from-num-plan
- from-num-pres
- from-subaddr
- from-subaddr-valid
- from-subaddr-type
- from-subaddr-odd
- from-tag
- to-all
- to-name
- to-name-valid
- to-name-charset
- to-name-pres
- to-num
- to-num-valid
- to-num-plan
- to-num-pres
- to-subaddr
- to-subaddr-valid
- to-subaddr-type
- to-subaddr-odd
- to-tag
- priv-orig-all
- priv-orig-name
- priv-orig-name-valid
- priv-orig-name-charset
- priv-orig-name-pres
- priv-orig-num
- priv-orig-num-valid
- priv-orig-num-plan
- priv-orig-num-pres
- priv-orig-subaddr
- priv-orig-subaddr-valid
- priv-orig-subaddr-type
- priv-orig-subaddr-odd
- priv-orig-tag
- priv-from-all
- priv-from-name
- priv-from-name-valid
- priv-from-name-charset
- priv-from-name-pres
- priv-from-num
- priv-from-num-valid
- priv-from-num-plan
- priv-from-num-pres
- priv-from-subaddr
- priv-from-subaddr-valid
- priv-from-subaddr-type
- priv-from-subaddr-odd
- priv-from-tag
- priv-to-all
- priv-to-name
- priv-to-name-valid
- priv-to-name-charset

    - `priv-to-name-pres`
    - `priv-to-num`
    - `priv-to-num-valid`
    - `priv-to-num-plan`
    - `priv-to-num-pres`
    - `priv-to-subaddr`
    - `priv-to-subaddr-valid`
    - `priv-to-subaddr-type`
    - `priv-to-subaddr-odd`
    - `priv-to-tag`
    - `reason`
    - `count`
- `i` - If set, this will prevent the channel from sending out protocol messages because of the value being set

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_REGEX

REGEX()

Synopsis

Check string against a regular expression.

Description

Return `1` on regular expression match or `0` otherwise

Please note that the space following the double quotes separating the regex from the data is optional and if present, is skipped. If a space is desired at the beginning of the data, then put two spaces there; the second will not be skipped.

Syntax

```
REGEX("regular expression" string)
```

Arguments

- `"regular expression"`
- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_REPLACE

REPLACE()

Synopsis

Replace a set of characters in a given string with another character.

Description

Iterates through a string replacing all the *find-chars* with *replace-char*. *replace-char* may be either empty or contain one character. If empty, all *find-chars* will be deleted from the output.

> **Note**
> The replacement only occurs in the output. The original variable is not altered.

Syntax

```
REPLACE(varname,find-chars[,replace-char])
```

Arguments

- `varname`
- `find-chars`
- `replace-char`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SET

SET()

Synopsis

SET assigns a value to a channel variable.

Description

Syntax

```
SET(varname=value)
```

Arguments

- `varname`
- `value`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SHA1

SHA1()

Synopsis

Computes a SHA1 digest.

Description

Generate a SHA1 digest via the SHA1 algorythm.

Example: Set(sha1hash=${SHA1(junky)})

Sets the asterisk variable sha1hash to the string `60fa5675b9303eb62f99a9cd47f9f5837d18f9a0` which is known as his hash

Syntax

```
SHA1(data)
```

Arguments

- `data` - Input string

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SHARED

SHARED()

Synopsis

Gets or sets the shared variable specified.

Description

Implements a shared variable area, in which you may share variables between channels.

The variables used in this space are separate from the general namespace of the channel and thus `SHARED(foo)` and `foo` represent two completely different variables, despite sharing the same name.

Finally, realize that there is an inherent race between channels operating at the same time, fiddling with each others' internal variables, which is why this special variable namespace exists; it is to remind you that variables in the SHARED namespace may change at any time, without warning. You should therefore take special care to ensure that when using the SHARED namespace, you retrieve the variable and store it in a regular channel variable before using it in a set of calculations (or you might be surprised by the result).

Syntax

```
SHARED(varname,channel)
```

Arguments

- `varname` - Variable name
- `channel` - If not specified will default to current channel. It is the complete channel name: `SIP/12-abcd1234` or the prefix only `SIP/12` .

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SHELL

SHELL()

Synopsis

Executes a command using the system shell and captures its output.

Description

Collects the output generated by a command executed by the system shell

Example: `Set(foo=${SHELL(echo \bar)})`

> **Note**
> The command supplied to this function will be executed by the system's shell, typically specified in the SHELL environment variable. There are many different system shells available with somewhat different behaviors, so the output generated by this function may vary between platforms.

Syntax

```
SHELL(command)
```

Arguments

- `command` - The command that the shell should execute.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_SHIFT

SHIFT()

Synopsis

Removes and returns the first item off of a variable containing delimited text

Description

Example:

exten => s,1,Set(array=one,two,three)

exten => s,n,While($["${SET(var=${SHIFT(array)})}" != ""])

exten => s,n,NoOp(var is ${var})

exten => s,n,EndWhile

This would iterate over each value in array, left to right, and would result in NoOp(var is one), NoOp(var is two), and NoOp(var is three) being executed.

Syntax

```
SHIFT(varname[,delimiter])
```

Arguments

- `varname`
- `delimiter`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_SIP_HEADER

SIP_HEADER()

Synopsis

Gets the specified SIP header from an incoming INVITE message.

Description

Since there are several headers (such as Via) which can occur multiple times, SIP_HEADER takes an optional second argument to specify which header with that name to retrieve. Headers start at offset `1`.

Please observe that contents of the SDP (an attachment to the SIP request) can't be accessed with this function.

Syntax

```
SIP_HEADER(name,number)
```

Arguments

- `name`
- `number` - If not specified, defaults to `1`.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SIPCHANINFO

SIPCHANINFO()

Synopsis

Gets the specified SIP parameter from the current channel.

Description

Syntax

```
SIPCHANINFO(item)
```

Arguments

- `item`
  - `peerip` - The IP address of the peer.
  - `recvip` - The source IP address of the peer.
  - `from` - The SIP URI from the `From:` header.
  - `uri` - The SIP URI from the `Contact:` header.
  - `useragent` - The Useragent header used by the peer.
  - `peername` - The name of the peer.
  - `t38passthrough` - 1 if T38 is offered or enabled in this channel, otherwise 0.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SIPPEER

SIPPEER()

Synopsis

Gets SIP peer information.

Description

Syntax

```
SIPPEER(peername,item)
```

Arguments

- peername
- item
  - ip - (default) The IP address.
  - port - The port number.
  - mailbox - The configured mailbox.
  - context - The configured context.
  - expire - The epoch time of the next expire.
  - dynamic - Is it dynamic? (yes/no).
  - callerid_name - The configured Caller ID name.
  - callerid_num - The configured Caller ID number.
  - callgroup - The configured Callgroup.
  - pickupgroup - The configured Pickupgroup.
  - namedcallgroup - The configured Named Callgroup.
  - namedpickupgroup - The configured Named Pickupgroup.
  - codecs - The configured codecs.
  - status - Status (if qualify=yes).
  - regexten - Extension activated at registration.
  - limit - Call limit (call-limit).
  - busylevel - Configured call level for signalling busy.
  - curcalls - Current amount of calls. Only available if call-limit is set.
  - language - Default language for peer.
  - accountcode - Account code for this peer.
  - useragent - Current user agent header used by peer.
  - maxforwards - The value used for SIP loop prevention in outbound requests
  - chanvarname - A channel variable configured with setvar for this peer.
  - codecx - Preferred codec index number *x* (beginning with zero).

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SMDI_MSG

SMDI_MSG()

Synopsis

Retrieve details about an SMDI message.

Description

This function is used to access details of an SMDI message that was pulled from the incoming SMDI message queue using the SMDI_MSG_RETRIEVE() function.

Syntax

```
SMDI_MSG(message_id,component)
```

Arguments

- `message_id`
- `component` - Valid message components are:
    - `number` - The message desk number
    - `terminal` - The message desk terminal
    - `station` - The forwarding station
    - `callerid` - The callerID of the calling party that was forwarded
    - `type` - The call type. The value here is the exact character that came in on the SMDI link. Typically, example values are:
      Options:
        - `D` - Direct Calls
        - `A` - Forward All Calls
        - `B` - Forward Busy Calls
        - `N` - Forward No Answer Calls

See Also

- Asterisk 11 Function_SMDI_MSG_RETRIEVE

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SMDI_MSG_RETRIEVE

SMDI_MSG_RETRIEVE()

Synopsis

Retrieve an SMDI message.

Description

This function is used to retrieve an incoming SMDI message. It returns an ID which can be used with the SMDI_MSG() function to access details of the message. Note that this is a destructive function in the sense that once an SMDI message is retrieved using this function, it is no longer in the global SMDI message queue, and can not be accessed by any other Asterisk channels. The timeout for this function is optional, and the default is 3 seconds. When providing a timeout, it should be in milliseconds.

The default search is done on the forwarding station ID. However, if you set one of the search key options in the options field, you can change this behavior.

Syntax

```
SMDI_MSG_RETRIEVE(smdi port,search key,timeout,options)
```

Arguments

- `smdi port`
- `search key`
- `timeout`
- `options`
    - `t` - Instead of searching on the forwarding station, search on the message desk terminal.
    - `n` - Instead of searching on the forwarding station, search on the message desk number.

See Also

- Asterisk 11 Function_SMDI_MSG

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SORT

SORT()

Synopsis

Sorts a list of key/vals into a list of keys, based upon the vals.

Description

Takes a comma-separated list of keys and values, each separated by a colon, and returns a comma-separated list of the keys, sorted by their values. Values will be evaluated as floating-point numbers.

Syntax

```
SORT(keyval,keyvaln[,...])
```

Arguments

- `keyval`
    - `key1`
    - `val1`
- `keyvaln`
    - `key2`
    - `val2`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPEECH

SPEECH()

Synopsis

Gets information about speech recognition results.

Description

Gets information about speech recognition results.

Syntax

```
SPEECH(argument)
```

Arguments

- `argument`
    - `status` - Returns `1` upon speech object existing, or `0` if not
    - `spoke` - Returns `1` if spoker spoke, or `0` if not
    - `results` - Returns number of results that were recognized.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPEECH_ENGINE

SPEECH_ENGINE()

Synopsis

Change a speech engine specific attribute.

Description

Changes a speech engine specific attribute.

Syntax

```
SPEECH_ENGINE(name)
```

Arguments

- `name`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPEECH_GRAMMAR

SPEECH_GRAMMAR()

Synopsis

Gets the matched grammar of a result if available.

Description

Gets the matched grammar of a result if available.

Syntax

```
SPEECH_GRAMMAR(nbest_number/result_number)
```

Arguments

- `nbest_number`
- `result_number`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPEECH_RESULTS_TYPE

SPEECH_RESULTS_TYPE()

Synopsis

Sets the type of results that will be returned.

Description

Sets the type of results that will be returned. Valid options are normal or nbest.

Syntax

```
SPEECH_RESULTS_TYPE()
```

Arguments

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPEECH_SCORE

SPEECH_SCORE()

Synopsis

Gets the confidence score of a result.

Description

Gets the confidence score of a result.

Syntax

```
SPEECH_SCORE(nbest_number/result_number)
```

Arguments

- `nbest_number`
- `result_number`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPEECH_TEXT

SPEECH_TEXT()

Synopsis

Gets the recognized text of a result.

Description

Gets the recognized text of a result.

Syntax

```
SPEECH_TEXT(nbest_number/result_number)
```

Arguments

- `nbest_number`
- `result_number`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SPRINTF

SPRINTF()

Synopsis

Format a variable according to a format string.

Description

Parses the format string specified and returns a string matching that format. Supports most options found in **sprintf(3)**. Returns a shortened string if a format specifier is not recognized.

Syntax

```
SPRINTF(format,arg1,arg2[,...],argN)
```

Arguments

- `format`
- `arg1`
- `arg2`
- `argN`

See Also

- `sprintf(3)`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SQL_ESC

SQL_ESC()

Synopsis

Escapes single ticks for use in SQL statements.

Description

Used in SQL templates to escape data which may contain single ticks `'` which are otherwise used to delimit data.

Example: SELECT foo FROM bar WHERE baz='${SQL_ESC(${ARG1})}'

Syntax

```
SQL_ESC(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SRVQUERY

SRVQUERY()

Synopsis

Initiate an SRV query.

Description

This will do an SRV lookup of the given service.

Syntax

```
SRVQUERY(service)
```

Arguments

- `service` - The service for which to look up SRV records. An example would be something like `_sip._udp.example.com`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SRVRESULT

SRVRESULT()

Synopsis

Retrieve results from an SRVQUERY.

Description

This function will retrieve results from a previous use of the SRVQUERY function.

Syntax

```
SRVRESULT(id,resultnum)
```

Arguments

- `id` - The identifier returned by the SRVQUERY function.
- `resultnum` - The number of the result that you want to retrieve.
  Results start at `1`. If this argument is specified as `getnum`, then it will return the total number of results that are available.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_STACK_PEEK

STACK_PEEK()

Synopsis

View info about the location which called Gosub

Description

Read the calling {{c}}ontext, {{e}}xtension, {{p}}riority, or {{l}}abel, as specified by *which*, by going up *n* frames in the Gosub stack. If *suppress* is true, then if the number of available stack frames is exceeded, then no error message will be printed.

Syntax

```
STACK_PEEK(n,which[,suppress])
```

Arguments

- `n`
- `which`
- `suppress`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_STAT

STAT()

Synopsis

Does a check on the specified file.

Description

Syntax

```
STAT(flag,filename)
```

Arguments

- `flag` - Flag may be one of the following:
  d - Checks if the file is a directory.
  e - Checks if the file exists.
  f - Checks if the file is a regular file.
  m - Returns the file mode (in octal)
  s - Returns the size (in bytes) of the file
  A - Returns the epoch at which the file was last accessed.
  C - Returns the epoch at which the inode was last changed.
  M - Returns the epoch at which the file was last modified.
- `filename`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_STRFTIME

STRFTIME()

Synopsis

Returns the current date/time in the specified format.

Description

STRFTIME supports all of the same formats as the underlying C function **strftime(3)**. It also supports the following format: `%[n]q` - fractions of a second, with leading zeros.

Example: `%3q` will give milliseconds and `%1q` will give tenths of a second. The default is set at milliseconds (n=3). The common case is to use it in combination with %S, as in `%S.%3q`.

Syntax

```
STRFTIME(epoch,timezone,format)
```

Arguments

- `epoch`
- `timezone`
- `format`

See Also

- `strftime(3)`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_STRPTIME

STRPTIME()

Synopsis

Returns the epoch of the arbitrary date/time string structured as described by the format.

Description

This is useful for converting a date into `EPOCH` time, possibly to pass to an application like SayUnixTime or to calculate the difference between the two date strings

Example: ${STRPTIME(2006-03-01 07:30:35,America/Chicago,%Y-%m-%d %H:%M:%S)} returns 1141219835

Syntax

```
STRPTIME(datetime,timezone,format)
```

Arguments

- `datetime`
- `timezone`
- `format`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_STRREPLACE

STRREPLACE()

Synopsis

Replace instances of a substring within a string with another string.

Description

Searches for all instances of the *find-string* in provided variable and replaces them with *replace-string*. If *replace-string* is an empty string, this will effecively delete that substring. If *max-replacements* is specified, this function will stop after performing replacements *max-replacements* times.

> **Note**
> The replacement only occurs in the output. The original variable is not altered.

Syntax

```
STRREPLACE(varname,find-string[,replace-string[,max-replacements]])
```

Arguments

- `varname`
- `find-string`
- `replace-string`
- `max-replacements`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_SYSINFO

SYSINFO()

Synopsis

Returns system information specified by parameter.

Description

Returns information from a given parameter.

Syntax

```
SYSINFO(parameter)
```

Arguments

- `parameter`
    - `loadavg` - System load average from past minute.
    - `numcalls` - Number of active calls currently in progress.
    - `uptime` - System uptime in hours.

        > **Note**
        > This parameter is dependant upon operating system.

    - `totalram` - Total usable main memory size in KiB.

        > **Note**
        > This parameter is dependant upon operating system.

    - `freeram` - Available memory size in KiB.

        > **Note**
        > This parameter is dependant upon operating system.

    - `bufferram` - Memory used by buffers in KiB.

        > **Note**
        > This parameter is dependant upon operating system.

    - `totalswap` - Total swap space still available in KiB.

        > **Note**
        > This parameter is dependant upon operating system.

    - `freeswap` - Free swap space still available in KiB.

        > **Note**
        > This parameter is dependant upon operating system.

    - `numprocs` - Number of current processes.

        > **Note**
        > This parameter is dependant upon operating system.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_TESTTIME

TESTTIME()

Synopsis

Sets a time to be used with the channel to test logical conditions.

Description

To test dialplan timing conditions at times other than the current time, use this function to set an alternate date and time. For example, you may wish to evaluate whether a location will correctly identify to callers that the area is closed on Christmas Day, when Christmas would otherwise fall on a day when the office is normally open.

Syntax

```
TESTTIME(date,time[,zone])
```

Arguments

- `date` - Date in ISO 8601 format
- `time` - Time in HH:MM:SS format (24-hour time)
- `zone` - Timezone name

See Also

- Asterisk 11 Application_GotoIfTime

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_TIMEOUT

TIMEOUT()

Synopsis

Gets or sets timeouts on the channel. Timeout values are in seconds.

Description

The timeouts that can be manipulated are:

absolute: The absolute maximum amount of time permitted for a call. Setting of 0 disables the timeout.

digit: The maximum amount of time permitted between digits when the user is typing in an extension. When this timeout expires, after the user has started to type in an extension, the extension will be considered complete, and will be interpreted. Note that if an extension typed in is valid, it will not have to timeout to be tested, so typically at the expiry of this timeout, the extension will be considered invalid (and thus control would be passed to the i extension, or if it doesn't exist the call would be terminated). The default timeout is 5 seconds.

response: The maximum amount of time permitted after falling through a series of priorities for a channel in which the user may begin typing an extension. If the user does not type an extension in this amount of time, control will pass to the t extension if it exists, and if not the call would be terminated. The default timeout is 10 seconds.

Syntax

```
TIMEOUT(timeouttype)
```

Arguments

- `timeouttype` - The timeout that will be manipulated. The possible timeout types are: `absolute`, `digit` or `response`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_TOLOWER

TOLOWER()

Synopsis

Convert string to all lowercase letters.

Description

Example: ${TOLOWER(Example)} returns "example"

Syntax

```
TOLOWER(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_TOUPPER

TOUPPER()

Synopsis

Convert string to all uppercase letters.

Description

Example: ${TOUPPER(Example)} returns "EXAMPLE"

Syntax

```
TOUPPER(string)
```

Arguments

- `string`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_TRYLOCK

TRYLOCK()

Synopsis

Attempt to obtain a named mutex.

Description

Attempts to grab a named lock exclusively, and prevents other channels from obtaining the same lock. Returns `1` if the lock was available or `0` otherwise.

Syntax

```
TRYLOCK(lockname)
```

Arguments

- `lockname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_TXTCIDNAME

TXTCIDNAME()

Synopsis

TXTCIDNAME looks up a caller name via DNS.

Description

This function looks up the given phone number in DNS to retrieve the caller id name. The result will either be blank or be the value found in the TXT record in DNS.

Syntax

```
TXTCIDNAME(number,zone-suffix)
```

Arguments

- `number`
- `zone-suffix` - If no *zone-suffix* is given, the default will be `e164.arpa`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_UNLOCK

UNLOCK()

Synopsis

Unlocks a named mutex.

Description

Unlocks a previously locked mutex. Returns `1` if the channel had a lock or `0` otherwise.

> **Note**
> It is generally unnecessary to unlock in a hangup routine, as any locks held are automatically freed when the channel is destroyed.

Syntax

```
UNLOCK(lockname)
```

Arguments

- `lockname`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_UNSHIFT

UNSHIFT()

Synopsis

Inserts one or more values to the beginning of a variable containing delimited text

Description

Example: Set(UNSHIFT(array)=one,two,three) would insert one, two, and three before the values stored in the variable "array".

Syntax

```
UNSHIFT(varname[,delimiter])
```

Arguments

- `varname`
- `delimiter`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r386286

Asterisk 11 Function_URIDECODE

URIDECODE()

Synopsis

Decodes a URI-encoded string according to RFC 2396.

Description

Returns the decoded URI-encoded *data* string.

Syntax

```
URIDECODE(data)
```

Arguments

- `data` - Input string to be decoded.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_URIENCODE

URIENCODE()

Synopsis

Encodes a string to URI-safe encoding according to RFC 2396.

Description

Returns the encoded string defined in *data*.

Syntax

```
URIENCODE(data)
```

Arguments

- `data` - Input string to be encoded.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_VALID_EXTEN

VALID_EXTEN()

Synopsis

Determine whether an extension exists or not.

Description

Returns a true value if the indicated *context*, *extension*, and *priority* exist.

> **Warning**
> This function has been deprecated in favor of the `DIALPLAN_EXISTS()` function

Syntax

```
VALID_EXTEN(context,extension,priority)
```

Arguments

- `context` - Defaults to the current context
- `extension`
- `priority` - Priority defaults to `1`.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_VERSION

VERSION()

Synopsis

Return the Version info for this Asterisk.

Description

If there are no arguments, return the version of Asterisk in this format: SVN-branch-1.4-r44830M

Example: Set(junky=${VERSION()};

Sets junky to the string `SVN-branch-1.6-r74830M`, or possibly, `SVN-trunk-r45126M`.

Syntax

```
VERSION(info)
```

Arguments

- `info` - The possible values are:
  - `ASTERISK_VERSION_NUM` - A string of digits is returned, e.g. 10602 for 1.6.2 or 100300 for 10.3.0, or 999999 when using an SVN build.
  - `BUILD_USER` - The string representing the user's name whose account was used to configure Asterisk, is returned.
  - `BUILD_HOSTNAME` - The string representing the name of the host on which Asterisk was configured, is returned.
  - `BUILD_MACHINE` - The string representing the type of machine on which Asterisk was configured, is returned.
  - `BUILD_OS` - The string representing the OS of the machine on which Asterisk was configured, is returned.
  - `BUILD_DATE` - The string representing the date on which Asterisk was configured, is returned.
  - `BUILD_KERNEL` - The string representing the kernel version of the machine on which Asterisk was configured, is returned.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_VM_INFO

VM_INFO()

Synopsis

Returns the selected attribute from a mailbox.

Description

Returns the selected attribute from the specified *mailbox*. If *context* is not specified, defaults to the `default` context. Where the *folder* can be specified, common folders include `INBOX`, `Old`, `Work`, `Family` and `Friends`.

Syntax

```
VM_INFO(mailbox,attribute[,folder])
```

Arguments

- `mailbox`
  - `mailbox`
  - `context`
- `attribute`
  - `count` - Count of messages in specified *folder*. If *folder* is not specified, defaults to `INBOX`.
  - `email` - E-mail address associated with the mailbox.
  - `exists` - Returns a boolean of whether the corresponding *mailbox* exists.
  - `fullname` - Full name associated with the mailbox.
  - `language` - Mailbox language if overridden, otherwise the language of the channel.
  - `locale` - Mailbox locale if overridden, otherwise global locale.
  - `pager` - Pager e-mail address associated with the mailbox.
  - `password` - Mailbox access password.
  - `tz` - Mailbox timezone if overridden, otherwise global timezone
- `folder` - If not specified, `INBOX` is assumed.

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_VMCOUNT

VMCOUNT()

Synopsis

Count the voicemails in a specified mailbox.

Description

Count the number of voicemails in a specified mailbox, you could also specify the *context* and the mailbox *folder*.

Example: `exten => s,1,Set(foo=${VMCOUNT(125)})`

Syntax

```
VMCOUNT(vmbox[,folder])
```

Arguments

- `vmbox`
    - `vmbox`
    - `context` - If not specified, defaults to `default`.
- `folder` - If not specified, defaults to `INBOX`

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

Asterisk 11 Function_VOLUME

VOLUME()

Synopsis

Set the TX or RX volume of a channel.

Description

The VOLUME function can be used to increase or decrease the `tx` or `rx` gain of any channel.

For example:

Set(VOLUME(TX)=3)

Set(VOLUME(RX)=2)

Set(VOLUME(TX,p)=3)

Set(VOLUME(RX,p)=3)

Syntax

```
VOLUME(direction,options)
```

Arguments

- `direction` - Must be TX or RX.
- `options`
    - p - Enable DTMF volume control

Import Version

This documentation was imported from Asterisk Version SVN-branch-11-r379343

