Asterisk 13 Reference

Asterisk Development Team <asteriskteam@digium.com>

New in 13

Overview

Asterisk 13 is the next Long Term Support (LTS) release of Asterisk. As such, the focus of development for this release of Asterisk was on improving the usability and features developed in the previous Standard release, Asterisk 12. Beyond a general refinement of end user features, development focussed heavily on the Asterisk APIs - the Asterisk Manager Interface (AMI) and the Asterisk REST Interface (ARI) - and the PJSIP stack in Asterisk. Some highlights of the new features include:

- Asterisk security events are now provided via AMI, allowing end users to monitor their Asterisk system in real time for security related issues.
- External control of Message Waiting Indicators (MWI) through both AMI and ARI.
- Reception/transmission of out of call text messages using any supported channel driver/protocol stack through ARI.
- Resource List Server support in the PJSIP stack, providing subscriptions to lists of resources and batched delivery of `NOTIFY` requests.
- Inter-Asterisk distributed device state and mailbox state using the PJSIP stack.

And much more!

It is important to note that Asterisk 13 is built on the architecture developed during the previous Standard release, Asterisk 12. Users upgrading to Asterisk 13 should read about the new features documented in New in 12, as well as the notes on upgrading to Asterisk 12. In particular, users upgrading to Asterisk 13 from a release prior to Asterisk 12 should read the specifications on AMI, CDRs, and CEL, as these also apply to Asterisk 13:

- AMI v2 Specification
- Asterisk 12 CEL Specification
- Asterisk 12 CDR Specification

Finally, all users upgrading to Asterisk 13 should read the notes on upgrading to Asterisk 13.

Asterisk 12 was different
Some of the new features listed below were released in point releases of Asterisk 12. Per the Software Configuration Management Policies laid out for Asterisk 12, new features were periodically merged and released in that branch of Asterisk. This was done to help users of Asterisk migrating to the new platform develop features in preparation for Asterisk 13.

While some of the features listed below were released under an Asterisk 12 release, they are all listed here as "new in 13", for two reasons:

1. If you are upgrading from a previous LTS release (such as Asterisk 11), all of these features are new.
2. If you are upgrading from some version of Asterisk 12, some of the previously released features may be new (as they may not have been in your version of Asterisk 12).

Applications

AgentRequest

- The application will now return a new `AGENT_STATUS` value of `NOT_CONNECTED` if the agent fails to connect with an incoming caller after being alerted to the presence of the incoming caller. The most likely reason this would happen is the agent did not acknowledge the call in time.

ChanSpy

- ChanSpy now accepts a channel uniqueid or a fully specified channel name as the `chanprefix` parameter if the `'u'` option is specified.

ConfBridge

- The ConfBridge dialplan application now sets a channel variable, `CONFBRIGE_RESULT`, upon exiting. This variable can be used to determine how a channel exited the conference. Valid values upon exiting are:

Value	Reason
FAILED	The channel encountered an error and could not enter the conference.
HANGUP	The channel exited the conference by hanging up.
KICKED	The channel was kicked from the conference.
ENDMARKED	The channel left the conference as a result of the last marked user leaving.
DTMF	The channel pressed a DTMF sequence to exit the conference.

- Added conference user option `'announce_join_leave_review'`. This option implies `'announce_join_leave'` with the added effect that the user will be asked if they want to confirm or re-record the recording of their name when entering the conference.

DAHDIBarge

- The module `app_dahdibarge` was deprecated and has been removed. Users of DAHDIBarge should use ChanSpy instead.

Directory

- At exit, the Directory application now sets a channel variable `DIRECTORY_RESULT` to one of the following based on the reason for exiting:

Value	Reason
OPERATOR	user requested operator by pressing '0' for operator
ASSISTANT	user requested assistant by pressing '*' for assistant
TIMEOUT	user pressed nothing and Directory stopped waiting
HANGUP	user's channel hung up
SELECTED	user selected a user from the directory and is routed
USEREXIT	user pressed '#' from the selection prompt to exit
FAILED	directory failed in a way that wasn't accounted for. Dang.

On this Page

MusicOnHold

- MusicOnHold streams (all modes other than "files") now support wide band audio.

MixMonitor

- A new option, `B()`, has been added that will turn on a periodic beep while the call is being recorded.
- New options to play a beep when starting a recording and stopping a recording have been added. The option `'p'` will play a beep to the channel that starts the recording. The option `'P'` will play a beep to the channel that stops the recording.

Monitor

- A new option, `B()`, has been added that will turn on a periodic beep while the call is being recorded.

Page

- Added options `'b'` and `'B'` to apply pre-dial handlers for outgoing calls and for the channel executing Page respectively.

PickupChan

- PickupChan now accepts channel uniqueids of channels to pickup.

ReadFile

- The module `app_readfile` was deprecated and has been removed. Users of ReadFile should use `func_env`'s FILE function instead.

Record

- The Record application now has an option `'o'` which allows `0` to act as an exit key. This will set the the `RECORD_STATUS` variable to `'OPERATOR'` instead of `'DTMF'`.

Say

- If the channel variable `SAY_DTMF_INTERRUPT` is present on a channel and set to `'true'` (case insensitive), then any Say application (SayNumber, SayDigits, SayAlpha, SayAlphaCase, SayUnixTime, and SayCounted) will anticipate DTMF. If DTMF is received, these applications will behave like the background application and jump to the received extension once a match is established or after a short period of inactivity.
- The Say family of dialplan applications now support the Japanese language. The `language` parameter in *`say.conf`* now recognizes a setting of `ja`, which will enable Japanese language specific mechanisms for playing back numbers, dates, and other items.

SayCountPL

- The module `app_saycountpl` was deprecated and has been removed. Users of `app_saycountpl` should use the Say family of applications.

SetMusicOnHold

- The SetMusicOnHold dialplan application was deprecated and has been removed. Users of the application should use the `CHANNEL` function's `musicclass` setting instead.

VoiceMail

- VoiceMail and VoiceMailMain now support the Japanese language. The `language` parameter in *`voicemail.conf`* now recognizes a setting of `ja`, which will enable prompts to be played back using a Japanese grammatical structure. Additional prompts are necessary for this functionality, including:
    - **jb-arimasu**: there is

- **jb-arimasen**: there is not
- **jb-oshitekudasai**: please press
- **jb-ni**: article ni
- **jb-ga**: article ga
- **jb-wa**: article wa
- **jb-wo**: article wo
- VoiceMail mailboxes configured in *voicemail.conf* can now have multiple e-mail address specified for a single mailbox. Each e-mail address is separated by the `|` character.

WaitMusicOnHold

- The WaitMusicOnHold dialplan application was deprecated and has been removed. Users of the application should use MusicOnHold with a duration parameter instead.

Build System

- The location of the sample configuration files delivered with Asterisk have been moved from `configs` to `configs/samples`. This allows for other sample configuration sets to be defined in the future. The action of `make samples` is exactly the same as previous versions of Asterisk.
- The `menuselect` tool has been pulled into the Asterisk repository. Generally, this change is transparent to those using tarballs of Asterisk; to those working directly with the Asterisk repository, there is no accessing of the `menuselect` or `mxml` external repositories.
- The `menuselect` tool no longer uses a bundled `mxml` library. Instead, it now uses `libxml2`. As a result, the `libxml2` development library is now a required dependency for Asterisk.

Core

Account Codes

- Support for `peeraccount` was vastly improved in this version of Asterisk. Except for Queue, an `accountcode` is now consistently propagated to outgoing channels before dialing. A channel's `accountcode` can change from its original non-empty value on channel creation for the following specific reasons:
  1. The dialplan sets it using `CHANNEL(accountcode)`.
  2. An originate method specifies an `accountcode` value.
  3. The calling channel propagates its `peeraccount` or `accountcode` to the outgoing channel's `accountcode` before dialing.

  This change has two visible effects. One, Local channels now cross `accountcode` and `peeraccount` codes across the special bridge between the `;1` and `;2` channels just like channels between normal bridges. Two, the `CHANNEL(peeraccount)` value can now be set before Dial and FollowMe to set the `accountcode` on the outgoing channel(s).
- For Queue, an outgoing channel's non-empty `accountcode` will not change unless explicitly set by `CHANNEL(accountcode)`. The change has three visible effects:
  1. As previously mentioned, Local channels now cross `accountcode` and `peeraccount` across the special bridge between the `;1` and `;2` channels just like channels between normal bridges.
  2. The queue member will get an `accountcode` if it doesn't have one and one is available from the calling channel's `peeraccount`.
  3. `accountcode` propagation includes Local channel members where the `accountcodes` are propagated early enough to be available on the `;2` channel.

AMI

- Added a new module that provides AMI control over MWI within Asterisk, `res_mwi_external_ami`. Note that this module depends on `res_mwi_external`; for more information on enabling this module, see `res_mwi_external`. This module provides the MWIGet/MWIUpdate/MWIDelete actions, as well as the MWIGet/MWIGetComplete events.

Actions

- Added DialplanExtensionAdd and DialplanExtensionRemove AMI actions. These actions are analogous to the `dialplan add extension` and `dialplan remove extension` CLI commands, respectively.
- Added AMI action LoggerRotate, which reloads and rotates logger in the same manner as the CLI command `logger rotate`.
- Added AMI actions FAXSessions, FAXSession, and FAXStats, which replicate the functionality of the CLI commands `fax show sessions`, `fax show session`, and `fax show stats` respectively.
- Added AMI actions PRIDebugSet, PRIDebugFileSet, and PRIDebugFileUnset, which enable manager control over PRI debugging levels and file output.

- The AMI action PJSIPNotify may now send to a URI instead of only to a PJSIP endpoint as long as a default outbound endpoint is set. This also applies to the equivalent CLI command (`pjsip send notify`).
- The AMI action PJSIPShowEndpoint now includes *ContactStatusDetail* sections that give information on Asterisk's attempts to qualify the endpoint.
- The MixMonitor action now has a *Command* header that can be used to indicate a post-process command to run once recording finishes.
- Added AMI actions DeviceStateList, PresenceStateList, and ExtensionStateList. Each of these can be used to list the current device states, presence states, and extension states respectively. The DeviceStateList and PresenceStateList actions are provided by the `res_manager_device_state.so` and `res_manager_presence_state.so` modules, respectively.
- Originate now takes optional parameters: *ChannelId* and *OtherChannelId*, which can be used to set the channel uniqueid on creation. The other id (specified by *OtherChannelId*) is only used when originating a Local channel, and is assigned to the second channel half of a Local channel. If a Local channel is originated and *OtherChannelId* is not specified, Asterisk will default to appending a `;2` to the identifier provided by *ChannelId*.

Events

- New DeviceStateChanged and PresenceStateChanged AMI events have been added. These events are emitted whenever a device state or presence state change occurs. The events are controlled by `res_manager_device_state.so` and `res_manager_presence_state.so`. If the high frequency of these events is problematic for you, do not load these modules.
- New events have been added for the `TALK_DETECT` function. When the function is used on a channel, ChannelTalkingStart/ChannelTalkingStop events will be emitted to connected AMI clients indicating the start/stop of talking on the channel.
- The *DialStatus* field in the DialEnd event can now contain additional statuses that convey how the dial operation terminated. This includes `ABORT`, `CONTINUE`, and `GOTO`.

- AMI will now emit security events. A new class authorization has been added in *manager.conf* for the security events, `security`. The new events are:

Event	Description
FailedACL	Raised when a request violates an ACL check.
InvalidAccountID	Raised when a request fails an authentication check due to an invalid account ID.
SessionLimit	Raised when a request fails due to exceeding the number of allowed concurrent sessions for a service.
MemoryLimit	Raised when a request fails due to an internal memory allocation failure. This event is a bit optimistic. While you may receive this event when Asterisk runs out of memory, it is highly likely that Asterisk is... out of memory. Making events is sometimes out of the question at that point.
LoadAverageLimit	Raised when a request fails because a configured load average limit has been reached.
RequestNotAllowed	Raised when a request is not allowed by the service..
AuthMethodNotAllowed	Raised when a request used an authentication method not allowed by the service.
RequestBadFormat	Raised when a request is received with bad formatting.
SuccessfulAuth	Raised when a request successfully authenticates.
UnexpectedAddress	Raised when a request has a different source address then what is expected for a session already in progress with a service.
ChallengeResponseFailed	Raised when a request's attempt to authenticate has been challenged, and the request failed the authentication challenge.
InvalidPassword	Raised when a request provides an invalid password during an authentication attempt.
ChallengeSent	Raised when an Asterisk service send an authentication challenge to a request.
InvalidTransport	Raised when a request attempts to use a transport not allowed by the Asterisk service.

- Bridge related events now have two additional fields: *BridgeName* and *BridgeCreator*. *BridgeName* is a descriptive name for the bridge; *B* *ridgeCreator* is the name of the entity that created the bridge. This affects the following events: ConfbridgeStart, ConfbridgeEnd, ConfbridgeJoin, ConfbridgeLeave, ConfbridgeRecord, ConfbridgeStopRecord, ConfbridgeMute, ConfbridgeUnmute, ConfbridgeTalking, BlindTransfer, AttendedTransfer, BridgeCreate, BridgeDestroy, BridgeEnter, and BridgeLeave.

ARI

- Operations that create a resource can now provide the unique identifier as a parameter to the creation request. This includes:
    - Channels:
        - A *channelId* can now be provided when creating a channel, either in the request URI (`POST channels/my-channel-id`) or as a query parameter. A Local channel will suffix the second channel id with `;2` unless the *otherChannelId* is provided as a query parameter.
        - A snoop channel can be started with a *snoopId*, in the request URI (`POST channels/my-channel-id/snoop/my-snoop-id`) or as a query parameter.
    - Bridges: A *bridgeId* can now be provided when creating a bridge, either in the request URI (`POST bridges/my-bridge-id`) or as a query parameter.
    - Playbacks: A *playbackId* can be provided when starting a playback, either in the request URI (`POST channels/my-channel-id/play/my-playback-id` or `POST bridges/my-bridge-id/play/my-playback-id`) or as a query parameter.
- Bridges: the bridge type used when creating a bridge is now a comma separated list of bridge properties. Valid options are: `mixing`, `holding`, `dtmf_events`, and `proxy_media`.
- The LiveRecording object in recording events now contains a *target_uri* field which contains the URI of what is being recorded.
- Stored recordings now support a new operation, copy. This will take an existing stored recording and copy it to a new location in the *recordings* directory.
- LiveRecording objects now have three additional fields that can be reported in a RecordingFinished ARI event:
    - *total_duration*: the duration of the recording.
    - *talking_duration*: optional. The duration of talking detected in the recording. This is only available if *max_silence_seconds* was specified when the recording was started.
    - *silence_duration*: optional. The duration of silence detected in the recording. This is only available if *max_silence_seconds* was specified when the recording was started.

  Note that all duration values are reported in seconds.
- Users of ARI can now send and receive out of call text messages. Messages can be sent using a `sendMessage` operation either directly to a particular endpoint or to the endpoints resource directly. In the latter case, the destination is derived from the URI scheme. Text messages are passed to ARI clients as TextMessageReceived events. ARI clients can choose to receive text messages by subscribing to the particular endpoint technology or endpoints that they are interested in.
- The `applications` resource now supports subscriptions to all endpoints of a particular channel technology. For example, subscribing to an `eventSource` of `endpoint:PJSIP` will subscribe to all `PJSIP` endpoints.
- New event models have been added for the `TALK_DETECT` function. When the function is used on a channel, ChannelTalkingStarted/ChannelTalkingFinished events will be emitted to connected WebSockets subscribed to the channel, indicating the start/stop of talking on the channel.
- A new Playback URI `tone` has been added. Tones are specified either as an indication name, e.g., `tone:busy` , from *indications.conf* or as a tone pattern, e.g., `tone:240/250,0/250`. Tones differ from normal playback URIs in that they must be stopped manually and will continue to occupy a channel's ARI control queue until they are stopped. They also can not be rewound or fast-forwarded.
- User events can now be generated from ARI. Events can be signalled with arbitrary JSON variables, and include one or more of `channel`, `bridge`, or `endpoint` snapshots. An application must be specified which will receive the event message (other applications can subscribe to it). If a `channel` is specified, the message will also be delivered to connected AMI clients. Dialplan generated user event messages are still transmitted via the `channel`, and will only be received by a Stasis application they are attached to or if something is subscribed to the `channel`.
- The Bridge data model now contains the additional fields *name* and *creator*. The *name* field conveys a descriptive name for the bridge; the *creator* field conveys the name of the entity that created the bridge. This affects all responses to HTTP requests that return a Bridge data model as well as all event derived data models that contain a Bridge data model. The `POST /bridges` operation may now optionally specify a *name* to give to the bridge being created.
- Added a new ARI resource mailboxes which allows the creation and modification of mailboxes managed by external MWI. Modules `res_mwi_external` and `res_stasis_mailbox` must be enabled to use this resource. For more information on external MWI control, see `res_mwi_external`.
- Added new events for externally initiated transfers. The event BridgeBlindTransfer is now raised when a channel initiates a blind transfer of a bridge in the ARI controlled application to the dialplan; the BridgeAttendedTransfer event is raised when a channel initiates an attended transfer of a bridge in the ARI controlled application to the dialplan.
- Channel variables may now be specified as a body parameter to the `POST /channels` operation. The `variables` key in the JSON is interpreted as a sequence of key/value pairs that will be added to the created channel as channel variables. Other parameters in the JSON body are treated as query parameters of the same name.

CEL

- The *bridge_technology* extra field key has been added to `BRIDGE_ENTER` and `BRIDGE_EXIT` events.

CLI

- `core show locks` output now includes Thread/LWP ID, if the platform supports this feature.
- New `logger add channel` and `logger remove channel` CLI commands have been added to allow creation and deletion of dynamic logger channels without configuration changes. These dynamic logger channels will only exist until the next restart of asterisk.

Features

- Channel variables are now substituted in arguments passed to applications run by using dynamic features.

HTTP

- Asterisk's HTTP server now supports chunked Transfer-Encoding. This will be automatically handled by the HTTP server if a request is received with a Transfer-Encoding type of `chunked`.

RealTime

- A new set of Alembic scripts has been added for CDR tables. This will create a `cdr` table with the default schema that Asterisk expects.
- Numerous updates have been made to the database schemas for several tables. See the Upgrading to Asterisk 13 notes for more information.

TLS

- The TLS core in Asterisk now supports Perfect Forward Secrecy (PFS). Enabling PFS is attempted by default, and is dependent on the configuration of the module using TLS.
  - Ephemeral ECDH (ECDHE) is enabled by default. To disable it, do not specify a ECDHE cipher suite in *sip.conf*, for example:

    ```
    tlscipher=AES128-SHA:DES-CBC3-SHA
    ```

  - Ephemeral DH (DHE) is disabled by default. To enable it, add DH parameters into the private key file, e.g., *sip.conf* `tlsprivatekey`. For example, the default `dh2048.pem` - see http://www.opensource.apple.com/source/OpenSSL098/OpenSSL098-35.1/src/apps/dh2048.pem?txt
  - Because clients expect the server to prefer PFS, and because OpenSSL sorts its cipher suites by bit strength, see `openssl ciphers -v DEFAULT`. Consider re-ordering your cipher suites in the respective configuration file. For example:

    ```
    tlscipher=AES128+kEECDH:AES128+kEDH:3DES+kEDH:AES128-SHA:DES-CBC3-SHA:-ADH:-AECDH
    ```

    will use PFS when offered by the client. Clients which do not offer PFS fall-back to AES-128 (or even 3DES, as recommended by RFC 3261).

CDR Backends

cdr_sqlite

- This module was deprecated and has been removed. Users of `cdr_sqlite` should use `cdr_sqlite3_custom`.

cdr_pgsql

- Added the ability to support PostgreSQL `application_name` on connections. This allows PostgreSQL to display the configured name in the `pg_stat_activity` view and CSV log entries. This setting is configurable for `cdr_pgsql` via the `appname` configuration setting in *cdr_pgsql.conf*.

CEL Backends

cel_pgsql

- Added the ability to support PostgreSQL `application_name` on connections. This allows PostgreSQL to display the configured name in the `pg_stat_activity` view and CSV log entries. This setting is configurable for `cel_pgsql` via the `appname` configuration setting in *cel_pgsql.conf*.

Channel Drivers

chan_dahdi

- SS7 support now requires `libss7` v2.0 or later.
- Added SS7 support for connected line and redirecting.
- Most SS7 CLI commands have been reworked as well; additionally, new SS7 commands added. See the online CLI help for more information.
- Several SS7 config option parameters have been added; see the description in `chan_dahdi.conf.sample`.

chan_gtalk

- This module was deprecated and has been removed. Users of `chan_gtalk` should use `chan_motif`.

chan_h323

- This module was deprecated and has been removed. Users of `chan_h323` should use `chan_ooh323`.

chan_jingle

- This module was deprecated and has been removed. Users of `chan_jingle` should use `chan_motif`.

chan_sip

- The `SIPPEER` dialplan function no longer supports using a colon as a delimiter for parameters. The parameters for the function should be delimited using a comma.
- The `SIPCHANINFO` dialplan function was deprecated and has been removed. Users of the function should use the `CHANNEL` function instead.
- SIP peers can now specify `trust_id_outbound` which affects RPID/PAI fields for prohibited `callingpres` information. Values are `legacy`, `no`, and `yes`. By default, `legacy` is used.
    - `trust_id_outbound=legacy` - behaviour remains the same as in previous versions of Asterisk. When dealing with prohibited `callingpres` and `sendrpid=pai/rpid`, RPID/PAI headers are appended to outbound SIP messages just as they are with allowed `callingpres` values, but data about the remote party's identity is anonymized. When `sendrpid=rpid`, only the remote party's domain is anonymized.
    - `trust_id_outbound=no` - when dealing with prohibited `callingpres`, RPID/PAI headers are not sent.
    - `trust_id_outbound=yes` - RPID/PAI headers are applied with the full remote party information intact even for prohibited `callingpres` information. In the case of PAI, a `Privacy: id` header will be appended for prohibited calling information to communicate that the private information should not be relayed to untrusted parties.
- TEL URI support for inbound INVITE requests has been added. `chan_sip` will now handle TEL schemes in the Request and From URIs. The phone-context in the Request URI will be stored in the `SIPURIPHONECONTEXT` channel variable on the inbound channel.

Functions

AST_SORCERY

- The `AST_SORCERY` function exposes sorcery-based configuration files like *pjsip.conf* to the dialplan.

AUDIOHOOK_INHERIT

- The `AUDIOHOOK_INHERIT` function has been deprecated. Audiohooks are now unconditionally inherited through masquerades. As a side benefit, more than one audiohook of a given type may persist through a masquerade now.

CONFBRIDGE

- The CONFBRIDGE dialplan function is now capable of creating/modifying dynamic conference user menus.
- The CONFBRIDGE dialplan function is now capable of removing dynamic conference menus, bridge settings, and user settings that have been applied by the CONFBRIDGE dialplan function.

JACK_HOOK

- The `JACK_HOOK` function now supports audio with a sample rate higher than 8kHz.

MIXMONITOR

- A new function, `MIXMONITOR`, has been added to allow access to individual instances of MixMonitor on a channel.

PERIODIC_HOOK

- A new function, PERIODIC_HOOK, has been added which allows for running a periodic dialplan hook on a channel. Any audio generated by this hook will be injected into the call.

TALK_DETECT

- A new function, `TALK_DETECT`, has been added. When set on a channel, this function causes events indicating the starting/stopping of talking on said channel to be emitted to both AMI and ARI clients.

Resources

res_config_pgsql

- Added the ability to support PostgreSQL `application_name` on connections. This allows PostgreSQL to display the configured name in the `pg_stat_activity` view and CSV log entries. This setting is configurable for `res_config_pgsql` via the `dbappname` configuration setting in *res_pgsql.conf*.

res_hep

- A new module, `res_hep`, has been added that acts as a generic packet capture agent for the Homer Encapsulation Protocol (HEP) version 3. It can be configured via *`hep.conf`*. Other modules use `res_hep` to send message traffic to a HEP capture server.

res_hep_pjsip

- A new module, `res_hep_pjsip`, has been added that will forward PJSIP message traffic to a HEP capture server. See `res_hep` for more information.

res_hep_rtcp

- A new module, `res_hep_rtcp`, has been added that will forward RTCP call statistics to a HEP capture server. See `res_hep` for more information.

res_mwi_external

- A new module, `res_mwi_external`, has been added to Asterisk. This module acts as a base framework that other modules can build on top of to allow an external system to control MWI within Asterisk. For implementations that make use of `res_mwi_external`, see the `res_mwi_external_ami` notes under the AMI changes and `res_ari_mailboxes` notes under the ARI changes. Note that `res_mwi_external` conflicts with other modules that may produce MWI themselves, such as `app_voicemail`. `res_mwi_external` and other modules that depend on it cannot be built or loaded with `app_voicemail` present.

res_parking

- Manager action Park now takes an additional argument *AnnounceChannel* which can be used to announce the parked call's location to an arbitrary channel in a bridge. If *Channel* and *TimeoutChannel* are the two parties in a two-party bridge, *TimeoutChannel* is treated as having parked *Channel* (in the same manner as the Park Call DTMF feature) and will receive announcements prior to being hung up.

res_pjsip

- The `endpoint` configuration object now supports `accountcode`. Any channel created for an endpoint with this setting will have its `accountcode` set to the specified value.
- `transport` and `endpoint` ToS options (`tos`, `tos_audio`, and `tos_video`) may now be set as the named set of ToS values (`cs0` - `cs7`, `af11` - `af43`, `ef`).
- Added the following new CLI commands:
    - `pjsip show contacts` - list all current PJSIP contacts.
    - `pjsip show contact` - show specific information about a current PJSIP contact.
    - `pjsip show channel` - show detailed information about a PJSIP channel.
- Path support has been added with the `support_path` option in `registration` and `aor` sections. This functionality is provided by a new module, `res_pjsip_path.so`.
- A `debug` option has been added to the `globals` section that will allow sip messages to be logged.
- A `set_var` option has been added to endpoints that will automatically set the desired variable(s) on a channel created for that endpoint.
- DNS functionality will now automatically be enabled if the system configured nameservers can be retrieved. If the system configured nameservers can not be retrieved the functionality will resort to using basic system resolution. Functionality such as SRV records and fail-over will not be available if the basic system resolution is in use.
- Several new tables and columns have been added to the realtime schema for the `res_pjsip` related modules. See the UPGRADE notes for updating the database schema.

res_pjsip_multihomed

- A new module, `res_pjsip_multihomed` handles situations where the system Asterisk is running out has multiple interfaces. `res_pjsip_multihomed` determines which interface should be used during message sending.

res_pjsip_outbound_publish

- A new module, `res_pjsip_outbound_publish` provides the mechanisms for sending `PUBLISH` requests for specific event packages to another SIP User Agent. See Exchanging Device and Mailbox State Using PJSIP for examples on configuring this feature.

res_pjsip_outbound_registration

- A new CLI command has been added: `pjsip show registrations`, which lists all configured PJSIP registrations.

res_pjsip_pidf_digium_body_supplement

- A new module, `res_pjsip_pidf_digium_body_supplement` provides NOTIFY request body formatting for presence support in Digium phones.

res_pjsip_pubsub

- Subscriptions can now be persisted via the `subscription_persistence` object in *`pjsip.conf`*. Note that it is up to the configuration in *`sorcery.conf`* to determine how the subscription is persisted.
- The publish/subscribe core module has been updated to support RFC 4662 Resource Lists, allowing Asterisk to act as a Resource List Server (RLS). Resource lists are configured in `pjsip.conf` under a new object type, `resource_list`. Resource lists can contain either `message-summary` or `presence` events, can be composed of specific resources that provide the event, or other resource lists.
- Inbound publication support is provided by a new object, `inbound-publication`. This configures `res_pjsip_pubsub` to accept `PUBLISH` requests from a particular resource. Which events are accepted is constructed dynamically; see `res_pjsip_publish_asterisk` f

or more information and Exchanging Device and Mailbox State Using PJSIP for examples on configuring this feature.

res_pjsip_publish_asterisk

- A new module, `res_pjsip_publish_asterisk` adds support for `PUBLISH` requests of Asterisk information to other Asterisk servers. This module is intended only for Asterisk to Asterisk exchanges of information. Currently, this includes both mailbox state and device state information. See Exchanging Device and Mailbox State Using PJSIP for examples on configuring this feature.

res_pjsip_send_to_voicemail

- A new module, `res_pjsip_send_to_voicemail` allows for REFER requests with particular headers to transfer a PJSIP channel directly to a particular extension that has VoiceMail. This is intended to be used with Digium phones that support this feature.

Upgrading to Asterisk 13

Overview

As Asterisk 13 is built on the architecture introduced in Asterisk 12, users upgrading to Asterisk 13 from an older version of Asterisk should be aware of the architectural changes that were made in the previous Standard release. It is recommended that you review:

- The upgrade notes on this page
- The New in 13 information, which lists the major new features in Asterisk 13
- The notes on Upgrading to Asterisk 12 **if** you are upgrading from a version of Asterisk prior to Asterisk 12
  The notes on what is New in 12 **if** if you are upgrading from a version of Asterisk prior to Asterisk 12.

General Asterisk Updates

- The asterisk command line `-I` option and the *`asterisk.conf`* `internal_timing` option have been removed. Internal timing is always enabled if any timing module is loaded.
- The per console verbose level feature as previously implemented in Asterisk 11 caused a large performance penalty. The fix required some minor incompatibilities if the new `rasterisk` is used to connect to an earlier version. If the new `rasterisk` connects to an older Asterisk version then the root console verbose level is always affected by the `core set verbose` command of the remote console even though it may appear to only affect the current console. If an older version of `rasterisk` connects to the new version of Asterisk then the `core set verbose` command will have no effect.
- The asterisk compatibility options in *`asterisk.conf`* have been removed. These options enabled certain backwards compatibility features for `pbx_realtime`, `res_agi`, and `app_set` that made their behaviour similar to Asterisk 1.4. Users who used these backwards compatibility settings should update their dialplans to use `','` instead of `'|'` as a delimiter, and should use the Set dialplan application instead of the MSet dia lplan application.

Applications

ConfBridge

- The `sound_place_into_conference` sound used in ConfBridge is now deprecated and is no longer functional. It has technically been broken since its inception and - to meet its documented use case - a different method is used to achieve the same goal. The new method is to use `sound_begin` to play a sound to the conference when `waitmarked` users are moved into the conference.

SetMusicOnHold

- The SetMusicOnHold dialplan application was deprecated and has been removed. Users of the application should use the `CHANNEL` function's `musicclass` setting instead.

WaitMusicOnHold

- The WaitMusicOnHold dialplan application was deprecated and has been removed. Users of the application should use MusicOnHold with a `duration` parameter instead.

On this Page

Build System

- Sample config files have been moved from *`configs/`* to a sub-folder of that directory, *`samples`*.
- The `menuselect` utility has been pulled into the Asterisk repository. As a result, the `libxml2` development library is now a required dependency for Asterisk.
- A new Compiler Flag, `REF_DEBUG`, has been added. When enabled, reference counted objects will emit additional debug information to the *`refs`* log file located in the standard Asterisk log file directory. This log file is useful in tracking down object leaks and other reference counting issues. Prior to this version, this option was only available by modifying the source code directly. This change also includes a new script, *`refcounter.py`*, in the *`contrib`* folder that will process the *`refs`* log file. Note that this replaces the *`refcounter`* utility that could be built from the *`utils`* directory.

CDR Backends

cdr_sqlite

- The `cdr_sqlite` module was deprecated and has been removed. Users of this module should use the `cdr_sqlite3_custom` module instead.

Channel Drivers

chan_dahdi

- SS7 support now requires `libss7` v2.0 or later.
- Added the `inband_on_setup_ack` compatibility option to *`chan_dahdi.conf`* to deal with switches that don't send an inband progress indication in the `SETUP ACKNOWLEDGE` message. Default is now `no`.

chan_gtalk

- This module was deprecated and has been removed. Users of `chan_gtalk` should use `chan_motif`.

chan_h323

- This module was deprecated and has been removed. Users of `chan_h323` should use `chan_ooh323`.

chan_jingle

- This module was deprecated and has been removed. Users of `chan_jingle` should use `chan_motif`.

chan_pjsip

- Added a `force_avp` option to `chan_pjsip` which will force the usage of `RTP/AVP`, `RTP/AVPF`, `RTP/SAVP`, or `RTP/SAVPF` as the media transport type in SDP offers depending on settings, even when DTLS is used for media encryption. This option can be set to improve interoperability with WebRTC clients that don't use the RFC defined transport for DTLS.
- Added a `media_use_received_transport` option to `chan_pjsip` which will cause the SDP answer to use the media transport as received in the SDP offer.

chan_sip

- Made set `SIPREFERREDBYHDR` as inheritable for better `chan_pjsip` interoperability.
- The `SIPPEER` dialplan function no longer supports using a colon as a delimiter for parameters. The parameters for the function should be delimited using a comma.
- The `SIPCHANINFO` dialplan function was deprecated and has been removed. Users of the function should use the `CHANNEL` function instead.
- Added a `force_avp` option for `chan_sip`. When enabled this option will cause the media transport in the offer or answer SDP to be `RTP/AVP`, `RTP/AVPF`, `RTP/SAVP`, or `RTP/SAVPF` even if a DTLS stream has been configured. This option can be set to improve interoperability with WebRTC clients that don't use the RFC defined transport for DTLS.
- The `dtlsverify` option in `chan_sip` now has additional values besides `yes` and `no`. If `yes` is specified both the certificate and fingerprint will be verified. If `no` is specified then neither the certificate or fingerprint is verified. If `certificate` is specified then only the certificate is verified. If `fingerprint` is specified then only the fingerprint is verified.
- A `dtlsfingerprint` option has been added to `chan_sip` which allows the hash to be specified for the DTLS fingerprint placed in SDP. Supported values are `sha-1` and `sha-256` with `sha-256` being the default.
- The `progressinband=never` option is now more zealous in the persecution of progress messages coming from Asterisk. Channels bridged with a SIP channel that has `progressinband=never` set will not be able to forward their progress indications through to the SIP device. `chan_sip` will now turn such progress indications into a 180 Ringing (if a 180 has not yet been transmitted) if `progressinband=never`.
- The codec preference order in an SDP during an offer is slightly different than previous releases. Prior to Asterisk 13, the preference order of codecs used to be:

  a. Our preferred codec
  b. Our configured codecs
  c. Any non-audio joint codecs

> **Internal Implementation Details Ahead**
> One of the ways the new media format architecture in Asterisk 13 improves performance is by reference counting formats, such that they can be reused in many places without additional allocation. To not require a large amount of locking, an instance of a format is immutable by convention. This works well except for formats with attributes. Since a media format with an attribute is a different object than the same format without an attribute, we have to carry over the formats with attributes from an inbound offer so that the correct attributes are offered in an outgoing INVITE request. This requires some subtle tweaks to the preference order to ensure that the media format with attributes is offered to a remote peer, as opposed to the same media format (but without attributes) that may be stored in the peer object.

  Now, in Asterisk 13, the preference order of codecs is:

  a. Our preferred codec
  b. Any joint codecs offered by the inbound offer
  c. All other codecs that are not the preferred codec and not a joint codec offered by the inbound offer

- chan_sip is now an extended support module.

chan_unistim

- The *unistim.conf* `dateformat` has changed the meaning of options values to conform to the values used inside Unistim protocol.
- Added `dtmf_duration` option with changing default operation to disable received DTMF playback on a Unistim phone.

Core

- The behaviour of `accountcode` has changed somewhat to support `peeraccount`. The main change is that Local channels now cross a `ccountcode` and `peeraccount` settings across the special bridge between the `;1` and `;2` channels just like channels between normal bridges. See New in 13 for more information.

ARI

- The ARI version has been changed to 1.5.0. This is to reflect the backwards compatible changes listed in New in 13.
- A bug fix in bridge creation has caused a behavioural change in how subscriptions are created for bridges. A bridge created through ARI, does not, by itself, have a subscription created for any particular Stasis application. When a channel in a Stasis application joins a bridge, an implicit event subscription is created for that bridge as well. Previously, when a channel left such a bridge, the subscription was leaked; this allowed for later bridge events to continue to be pushed to the subscribed applications. That leak has been fixed; as a result, bridge events that were delivered after a channel left the bridge are no longer delivered. An application must subscribe to a bridge through the applications resource if it wishes to receive all events related to a bridge.

AMI

- The AMI version has been changed to 2.5.0. This is to reflect the backwards compatible changes listed in New in 13.
- MixMonitor AMI actions now require users to have authorization classes:
    - MixMonitor - `system`
    - MixMonitorMute - `call` or `system`
    - StopMixMonitor - `call` or `system`
- The undocumented *manager.conf* setting `block-sockets` has been removed. It interferes with TCP/TLS inactivity timeouts.
- The response to the PresenceState AMI action has historically contained two *Message* keys. The first of these is used as an informative message regarding the success/failure of the action; the second contains a Presence state specific message. Having two keys with the same unique name in an AMI message is cumbersome for some client; hence, the Presence specific *Message* has been deprecated. The message will now contain a *PresenceMessage* key for the presence specific information; the *Message* key containing presence information will be removed in the next major version of AMI.
- The *manager.conf* setting `eventfilter` now takes an "extended" regular expression instead of a "basic" one.

CDR

- The `endbeforehexten` setting now defaults to `yes`, instead of `no`. When set to `no`, this setting will cause a new CDR to be generated when a channel enters into hangup logic (either the `'h'` extension or a hangup handler subroutine). In general, this is not the preferred default: this causes extra CDRs to be generated for a channel in many common dialplans.

CLI

- `core show settings` now lists the current console verbosity in addition to the root console verbosity.
- `core set verbose` has not been able to support the by module verbose logging levels since verbose logging levels were made per console. That syntax is now removed and a silence option added in its place.

HTTP

- Added *http.conf* `session_inactivity` timer option to close HTTP connections that aren't doing anything.
- Added support for persistent HTTP connections. To enable persistent HTTP connections configure the keep alive time between HTTP requests. The keep alive time between HTTP requests is configured in *http.conf* with the `session_keep_alive` parameter.

Logging

- The `verbose` setting in *logger.conf* still takes an optional argument, specifying the verbosity level for each logging destination. However, the default is now to once again follow the current root console level. As a result, using the AMI Command action with `core set verbose` could again set the root console verbose level and affect the verbose level logged.

RealTime

> **Whoops**
> The database migration script that adds the `extensions` table had to be modified due to an error when installing for MySQL. The `extensions` table's `id` column was changed to be a primary key. This could potentially cause a migration problem. If so, it may be necessary to manually alter the affected table/column to bring it back in line with the migration scripts.

- A number of Alembic scripts have been updated between Asterisk 12 and Asterisk 13. These include the following:
  - For the `config` RealTime schemas:
    - `1758e8bbf6b_increase_useragent_column_size.py` - increase the size of the `useragent` column in `sippeers` from `20` characters to 255 characters.
    - `1d50859ed02e_create_accountcode.py` - add the `accountcode` column to the `ps_endpoints` table.
    - `21e526ad3040_add_pjsip_debug_option.py` - add the `debug` column to the `ps_globals` table.
    - `28887f25a46f_create_queue_tables.py` - creates the various Queue related tables.
    - `2fc7930b41b3_add_pjsip_endpoint_options_for_12_1.py` - adds the `ps_systems`, `ps_globals`, `ps_transports`, and `ps_registrations` tables. Adds several new columns for `ps_endpoints`, `ps_contacts`, and `ps_aors`.
    - `3855ee4e5f85_add_missing_pjsip_options.py` - adds the `message_context` column for the `ps_endpoints` table and the `user_agent` column for the `ps_contacts` table.
    - `4c573e7135bd_fix_tos_field_types.py` - changes the type of the `ps_endpoints.tos_audio`, `ps_endpoints.tos_video`, and `ps_transports.tos` columns.
    - `5139253c0423_make_q_member_uniqueid_autoinc.py` - modifies the `uniqueid` column on the `queue_members` table to be a unique auto-incrementing index, if the database supports it.
    - `51f8cb66540e_add_further_dtls_options.py` - adds the `force_avp` and `media_use_received_transport` columns to the `ps_endpoints` table.
    - `c6d929b23a8_create_pjsip_subscription_persistence_.py` - adds the `ps_subscription_persistence` table.
    - `e96a0b8071c_increase_pjsip_column_size.py` - increases the size of the columns `ps_globals.user_agent`, `ps_contacts.id`, `ps_contacts.uri`, `ps_contacts.user_agent`, `ps_registrations.client_uri`, and `ps_registrations.server_uri`.
  - For the `voicemail` ODBC backend schemas:
    - 39428242f7f5_increase_recording_column_size.py - changed the type of the `voicemail_messages.recording` column to `LargeBinary`, with a max size of `4294967295`.
  - Added a new family of schemas for CDR backends, `cdr`.

Resources

res_http_websocket

- Added a compatibility option to *`ari.conf`*, *`sip.conf`*, and *`pjsip.conf`* - `websocket_write_timeout`. When a websocket connection exists where Asterisk writes a substantial amount of data to the connected client, and the connected client is slow to process the received data, the socket may be disconnected. In such cases, it may be necessary to adjust this value. Default is 100 ms.

res_odbc

- The compatibility setting, `allow_empty_string_in_nontext`, has been removed. Empty column values will be stored as empty strings during RealTime updates.

res_jabber

- This module was deprecated and has been removed. Users of this module should use `res_xmpp` instead.

Scripts

safe_asterisk

- The *`safe_asterisk`* script was previously not installed on top of an existing version. This caused bug-fixes in that script not to be deployed. If your *`safe_asterisk`* script is customized, be sure to keep your changes. Custom values for variables should be created in `*.sh` file(s) inside *`ASTETCDIR/startup.d/`*. For more information, see the original bug report that necessitated this change, ASTERISK-21965.
- Changed a log message in *`safe_asterisk`* and the `$NOTIFY` mail subject. If you use tools to parse either of them, update your parse functions accordingly. The changed strings are:

- `"Exited on signal $EXITSIGNAL"` => `"Asterisk exited on signal $EXITSIGNAL."`
- `"Asterisk Died"` => `"Asterisk on $MACHINE died (sig $EXITSIGNAL)"`

Utilities

refcounter

- The *`refcounter`* program has been removed in favour of the *`refcounter.py`* script in *`contrib/scripts`*.

Asterisk 13 Command Reference

This page is the top level page for the XML/JSON derived documentation in Asterisk 13:

- Dialplan applications and functions
- Manager actions and events
- AGI commands
- ARI HTTP requests and events
- Asterisk module configurations

Note that all documentation contained in this section is auto-generated. Requested changes to the documentation in this section should be made as patches to the Asterisk source through the Asterisk issue tracker.

Asterisk 13 AGI Commands

Asterisk 13 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 13 AGICommand_hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 AGICommand_hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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. If offsetms is provided then the audio will seek to offsetms before play starts. 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. Returns the position where playback was terminated as endpos.

It sets the following channel variables upon completion:

- `CPLAYBACKSTATUS` - Contains the status of the attempt as a text string
  - SUCCESS
  - USERSTOPPED
  - REMOTESTOPPED
  - 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

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

Arguments

- `filename` - The file extension must not be included in the filename.
- `escape_digits`
- `skipms`
- `ffchar` - Defaults to `*`
- `rewchr` - Defaults to `#`
- `pausechr`
- `offsetms` - Offset, in milliseconds, to start the audio playback

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 AGICommand_stream file

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

- Asterisk 13 Application_GoSub

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 AGICommand_noop

NOOP

Synopsis

Does nothing.

Description

Does nothing.

Syntax

```
NOOP
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 *timeout*. *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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 AGICommand_speech create

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

It sets the following channel variables upon completion:

- `PLAYBACKSTATUS` - The status of the playback attempt as a text string.
  - SUCCESS
  - FAILED

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 13 AGICommand_control stream file

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 AMI Actions

Asterisk 13 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).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_Agents

Agents

Synopsis

Lists agents and their status.

Description

Will list info about all defined agents.

Syntax

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

Arguments

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

See Also

- Asterisk 13 ManagerEvent_Agents
- Asterisk 13 ManagerEvent_AgentsComplete

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_Atxfer

Atxfer

Synopsis

Attended transfer.

Description

Attended transfer.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BlindTransfer

BlindTransfer

Synopsis

Blind transfer channel(s) to the given destination

Description

Redirect all channels currently bridged to the specified channel to the specified destination.

Syntax

```
Action: BlindTransfer
Channel: <value>
Context: <value>
Exten: <value>
```

Arguments

- `Channel`
- `Context`
- `Exten`

See Also

- Asterisk 13 ManagerAction_Redirect

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.
    - `no`
    - `Channel1`
    - `Channel2`
    - `Both`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeDestroy

BridgeDestroy

Synopsis

Destroy a bridge.

Description

Deletes the bridge, causing channels to continue or hang up.

Syntax

```
Action: BridgeDestroy
ActionID: <value>
BridgeUniqueid: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `BridgeUniqueid` - The unique ID of the bridge to destroy.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeInfo

BridgeInfo

Synopsis

Get information about a bridge.

Description

Returns detailed information about a bridge and the channels in it.

Syntax

```
Action: BridgeInfo
ActionID: <value>
BridgeUniqueid: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `BridgeUniqueid` - The unique ID of the bridge about which to retreive information.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeKick

BridgeKick

Synopsis

Kick a channel from a bridge.

Description

The channel is removed from the bridge.

Syntax

```
Action: BridgeKick
ActionID: <value>
[BridgeUniqueid:] <value>
Channel: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `BridgeUniqueid` - The unique ID of the bridge containing the channel to destroy. This parameter can be omitted, or supplied to insure that the channel is not removed from the wrong bridge.
- `Channel` - The channel to kick out of a bridge.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeList

BridgeList

Synopsis

Get a list of bridges in the system.

Description

Returns a list of bridges, optionally filtering on a bridge type.

Syntax

```
Action: BridgeList
ActionID: <value>
BridgeType: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `BridgeType` - Optional type for filtering the resulting list of bridges.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeTechnologyList

BridgeTechnologyList

Synopsis

List available bridging technologies and their statuses.

Description

Returns detailed information about the available bridging technologies.

Syntax

```
Action: BridgeTechnologyList
ActionID: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeTechnologySuspend

BridgeTechnologySuspend

Synopsis

Suspend a bridging technology.

Description

Marks a bridging technology as suspended, which prevents subsequently created bridges from using it.

Syntax

```
Action: BridgeTechnologySuspend
ActionID: <value>
BridgeTechnology: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `BridgeTechnology` - The name of the bridging technology to suspend.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_BridgeTechnologyUnsuspend

BridgeTechnologyUnsuspend

Synopsis

Unsuspend a bridging technology.

Description

Clears a previously suspended bridging technology, which allows subsequently created bridges to use it.

Syntax

```
Action: BridgeTechnologyUnsuspend
ActionID: <value>
BridgeTechnology: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `BridgeTechnology` - The name of the bridging technology to unsuspend.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - If this parameter is not a complete channel name, the first channel with this prefix will be used.
  If this parameter is "all", all channels will be kicked from the conference.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - If this parameter is not a complete channel name, the first channel with this prefix will be used.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - If this parameter is not a complete channel name, the first channel with this prefix will be used.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - If this parameter is not a complete channel name, the first channel with this prefix will be used.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_ControlPlayback

ControlPlayback

Synopsis

Control the playback of a file being played to a channel.

Description

Control the operation of a media file being played back to a channel. Note that this AMI action does not initiate playback of media to channel, but rather controls the operation of a media operation that was already initiated on the channel.

> **Note**
> The `pause` and `restart` *Control* options will stop a playback operation if that operation was not initiated from the *ControlPlayback* application or the *control stream file* AGI command.

Syntax

```
Action: ControlPlayback
ActionID: <value>
Channel: <value>
Control: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - The name of the channel that currently has a file being played back to it.
- `Control`
    - `stop` - Stop the playback operation.
    - `forward` - Move the current position in the media forward. The amount of time that the stream moves forward is determined by the *skipms* value passed to the application that initiated the playback.

      > **Note**
      > The default skipms value is `3000` ms.

    - `reverse` - Move the current position in the media backward. The amount of time that the stream moves backward is determined by the *skipms* value passed to the application that initiated the playback.

      > **Note**
      > The default skipms value is `3000` ms.

    - `pause` - Pause/unpause the playback operation, if supported. If not supported, stop the playback.
    - `restart` - Restart the playback operation, if supported. If not supported, stop the playback.

See Also

- Asterisk 13 Application_Playback
- Asterisk 13 Application_ControlPlayback
- Asterisk 13 AGICommand_stream file
- Asterisk 13 AGICommand_control stream file

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_DeviceStateList

DeviceStateList

Synopsis

List the current known device states.

Description

This will list out all known device states in a sequence of *DeviceStateChange* events. When finished, a *DeviceStateListComplete* event will be emitted.

Syntax

```
Action: DeviceStateList
ActionID: <value>
```

Arguments

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

See Also

- Asterisk 13 ManagerEvent_DeviceStateChange
- Asterisk 13 Function_DEVICE_STATE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_DialplanExtensionAdd

DialplanExtensionAdd

Synopsis

Add an extension to the dialplan

Description

Syntax

```
Action: DialplanExtensionAdd
ActionID: <value>
Context: <value>
Extension: <value>
Priority: <value>
Application: <value>
[ApplicationData:] <value>
[Replace:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Context` - Context where the extension will be created. The context will be created if it does not already exist.
- `Extension` - Name of the extension that will be created (may include callerid match by separating with '/')
- `Priority` - Priority being added to this extension. Must be either `hint` or a numerical value.
- `Application` - The application to use for this extension at the requested priority
- `ApplicationData` - Arguments to the application.
- `Replace` - If set to 'yes', '1', 'true' or any of the other values we evaluate as true, then if an extension already exists at the requested context, extension, and priority it will be overwritten. Otherwise, the existing extension will remain and the action will fail.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_DialplanExtensionRemove

DialplanExtensionRemove

Synopsis

Remove an extension from the dialplan

Description

Syntax

```
Action: DialplanExtensionRemove
ActionID: <value>
Context: <value>
Extension: <value>
[Priority:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Context` - Context of the extension being removed
- `Extension` - Name of the extension being removed (may include callerid match by separating with '/')
- `Priority` - If provided, only remove this priority from the extension instead of all priorities in the extension.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_ExtensionStateList

ExtensionStateList

Synopsis

List the current known extension states.

Description

This will list out all known extension states in a sequence of *ExtensionStatus* events. When finished, a *ExtensionStateListComplete* event will be emitted.

Syntax

```
Action: ExtensionStateList
ActionID: <value>
```

Arguments

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

See Also

- Asterisk 13 ManagerAction_ExtensionState
- Asterisk 13 Function_HINT
- Asterisk 13 Function_EXTENSION_STATE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_FAXSession

FAXSession

Synopsis

Responds with a detailed description of a single FAX session

Description

Provides details about a specific FAX session. The response will include a common subset of the output from the CLI command 'fax show session <session_number>' for each technology. If the FAX technolgy used by this session does not include a handler for FAXSession, then this action will fail.

Syntax

```
Action: FAXSession
ActionID: <value>
SessionNumber: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `SessionNumber` - The session ID of the fax the user is interested in.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_FAXSessions

FAXSessions

Synopsis

Lists active FAX sessions

Description

Will generate a series of FAXSession events with information about each FAXSession. Closes with a FAXSessionsComplete event which includes a count of the included FAX sessions. This action works in the same manner as the CLI command 'fax show sessions'

Syntax

```
Action: FAXSessions
ActionID: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_FAXStats

FAXStats

Synopsis

Responds with fax statistics

Description

Provides FAX statistics including the number of active sessions, reserved sessions, completed sessions, failed sessions, and the number of receive/transmit attempts. This command provides all of the non-technology specific information provided by the CLI command 'fax show stats'

Syntax

```
Action: FAXStats
ActionID: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

Syntax

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_Getvar

Getvar

Synopsis

Gets a channel variable or function value.

Description

Get the value of a channel variable or function return.

> **Note**
> If a channel name is not provided then the variable is considered 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, function or expression.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_IAXnetstats

IAXnetstats

Synopsis

Show IAX Netstats.

Description

Show IAX channels network statistics.

Syntax

```
Action: IAXnetstats
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_IAXpeers

IAXpeers

Synopsis

List IAX peers.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_LoggerRotate

LoggerRotate

Synopsis

Reload and rotate the Asterisk logger.

Description

Reload and rotate the logger. Analogous to the CLI command 'logger rotate'.

Syntax

```
Action: LoggerRotate
ActionID: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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*.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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*.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_MessageSend

MessageSend

Synopsis

Send an out of call message to an endpoint.

Description

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: PJSIP**
      Specifying a prefix of `pjsip:` will send the message as a SIP MESSAGE request.
    - **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: PJSIP**
      The `from` parameter can be a configured endpoint or in the form of "display-name" <URI>.
    - **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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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>
Command: <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.
- `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

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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
ActionID: <value>
Module: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Module` - Asterisk module name (not including extension).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_MWIDelete

MWIDelete

Synopsis

Delete selected mailboxes.

Description

Delete the specified mailboxes.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Mailbox` - Mailbox ID in the form of / *regex*/ for all mailboxes matching the regular expression. Otherwise it is for a specific mailbox.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_MWIGet

MWIGet

Synopsis

Get selected mailboxes with message counts.

Description

Get a list of mailboxes with their message counts.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Mailbox` - Mailbox ID in the form of / *regex*/ for all mailboxes matching the regular expression. Otherwise it is for a specific mailbox.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_MWIUpdate

MWIUpdate

Synopsis

Update the mailbox message counts.

Description

Update the mailbox message counts.

Syntax

```
Action: MWIUpdate
ActionID: <value>
Mailbox: <value>
OldMessages: <value>
NewMessages: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Mailbox` - Specific mailbox ID.
- `OldMessages` - The number of old messages in the mailbox. Defaults to zero if missing.
- `NewMessages` - The number of new messages in the mailbox. Defaults to zero if missing.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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>
ChannelId: <value>
OtherChannelId: <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.
- `ChannelId` - Channel UniqueId to be set on the channel.
- `OtherChannelId` - Channel UniqueId to be set on the second local channel.

See Also

- Asterisk 13 ManagerEvent_OriginateResponse

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_Park

Park

Synopsis

Park a channel.

Description

Park an arbitrary channel with optional arguments for specifying the parking lot used, how long the channel should remain parked, and what dial string to use as the parker if the call times out.

Syntax

```
Action: Park
ActionID: <value>
Channel: <value>
[TimeoutChannel:] <value>
[AnnounceChannel:] <value>
[Timeout:] <value>
[Parkinglot:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to park.
- `TimeoutChannel` - Channel name to use when constructing the dial string that will be dialed if the parked channel times out. If `TimeoutChannel` is in a two party bridge with `Channel`, then `TimeoutChannel` will receive an announcement and be treated as having parked `Channel` in the same manner as the Park Call DTMF feature.
- `AnnounceChannel` - If specified, then this channel will receive an announcement when `Channel` is parked if `AnnounceChannel` is in a state where it can receive announcements (AnnounceChannel must be bridged). `AnnounceChannel` has no bearing on the actual state of the parked call.
- `Timeout` - Overrides the timeout of the parking lot for this park action. Specified in milliseconds, but will be converted to seconds. Use a value of 0 to disable the timeout.
- `Parkinglot` - The parking lot to use when parking the channel

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_ParkedCalls

ParkedCalls

Synopsis

List parked calls.

Description

List parked calls.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `ParkingLot` - If specified, only show parked calls from the parking lot with this name.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPNotify

PJSIPNotify

Synopsis

Send a NOTIFY to either an endpoint or an arbitrary URI.

Description

Sends a NOTIFY to an endpoint or an arbitrary URI.

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

> **Note**
> One (and only one) of `Endpoint` or `URI` must be specified. If `URI` is used, thedefault outbound endpoint will be used to send the message. If the default outbound endpoint isn't configured, this command can not send to an arbitrary URI.

Syntax

```
Action: PJSIPNotify
ActionID: <value>
[Endpoint:] <value>
[URI:] <value>
Variable: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Endpoint` - The endpoint to which to send the NOTIFY.
- `URI` - Abritrary URI to which to send the NOTIFY.
- `Variable` - Appends variables as headers/content to the NOTIFY. If the variable is named `Content`, then the value will compose the body of the message if another variable sets `Content-Type`. *name=value*

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPQualify

PJSIPQualify

Synopsis

Qualify a chan_pjsip endpoint.

Description

Qualify a chan_pjsip endpoint.

Syntax

```
Action: PJSIPQualify
ActionID: <value>
Endpoint: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Endpoint` - The endpoint you want to qualify.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowEndpoint

PJSIPShowEndpoint

Synopsis

Detail listing of an endpoint and its objects.

Description

Provides a detailed listing of options for a given endpoint. Events are issued showing the configuration and status of the endpoint and associated objects. These events include `EndpointDetail`, `AorDetail`, `AuthDetail`, `TransportDetail`, and `IdentifyDetail`. Some events may be listed multiple times if multiple objects are associated (for instance AoRs). Once all detail events have been raised a final `EndpointDetailComplete` event is issued.

Syntax

```
Action: PJSIPShowEndpoint
ActionID: <value>
Endpoint: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Endpoint` - The endpoint to list.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowEndpoints

PJSIPShowEndpoints

Synopsis

Lists PJSIP endpoints.

Description

Provides a listing of all endpoints. For each endpoint an `EndpointList` event is raised that contains relevant attributes and status information. Once all endpoints have been listed an `EndpointListComplete` event is issued.

Syntax

```
Action: PJSIPShowEndpoints
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowRegistrationsInbound

PJSIPShowRegistrationsInbound

Synopsis

Lists PJSIP inbound registrations.

Description

In response `InboundRegistrationDetail` events showing configuration and status information are raised for each inbound registration object. As well as `AuthDetail` events for each associated auth object. Once all events are completed an `InboundRegistrationDetailComplete` is issued.

Syntax

```
Action: PJSIPShowRegistrationsInbound
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowRegistrationsOutbound

PJSIPShowRegistrationsOutbound

Synopsis

Lists PJSIP outbound registrations.

Description

In response `OutboundRegistrationDetail` events showing configuration and status information are raised for each outbound registration object. `AuthDetail` events are raised for each associated auth object as well. Once all events are completed an `OutboundRegistrationDetailComplete` is issued.

Syntax

```
Action: PJSIPShowRegistrationsOutbound
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowResourceLists

PJSIPShowResourceLists

Synopsis

Displays settings for configured resource lists.

Description

Provides a listing of all resource lists. An event `ResourceListDetail` is issued for each resource list object. Once all detail events are completed a `ResourceListDetailComplete` event is issued.

Syntax

```
Action: PJSIPShowResourceLists
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowSubscriptionsInbound

PJSIPShowSubscriptionsInbound

Synopsis

Lists subscriptions.

Description

Provides a listing of all inbound subscriptions. An event `InboundSubscriptionDetail` is issued for each subscription object. Once all detail events are completed an `InboundSubscriptionDetailComplete` event is issued.

Syntax

```
Action: PJSIPShowSubscriptionsInbound
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPShowSubscriptionsOutbound

PJSIPShowSubscriptionsOutbound

Synopsis

Lists subscriptions.

Description

Provides a listing of all outbound subscriptions. An event `OutboundSubscriptionDetail` is issued for each subscription object. Once all detail events are completed an `OutboundSubscriptionDetailComplete` event is issued.

Syntax

```
Action: PJSIPShowSubscriptionsOutbound
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PJSIPUnregister

PJSIPUnregister

Synopsis

Unregister an outbound registration.

Description

Syntax

```
Action: PJSIPUnregister
ActionID: <value>
Registration: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Registration` - The outbound registration to unregister.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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>
[Duration:] <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Channel` - Channel name to send digit to.
- `Digit` - The DTMF digit to play.
- `Duration` - The duration, in milliseconds, of the digit to be played.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PresenceStateList

PresenceStateList

Synopsis

List the current known presence states.

Description

This will list out all known presence states in a sequence of *PresenceStateChange* events. When finished, a *PresenceStateListComplete* event will be emitted.

Syntax

```
Action: PresenceStateList
ActionID: <value>
```

Arguments

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

See Also

- Asterisk 13 ManagerAction_PresenceState
- Asterisk 13 ManagerEvent_PresenceStatus
- Asterisk 13 Function_PRESENCE_STATE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PRIDebugFileSet

PRIDebugFileSet

Synopsis

Set the file used for PRI debug message output

Description

Equivalent to the CLI command "pri set debug file <output-file>"

Syntax

```
Action: PRIDebugFileSet
ActionID: <value>
File: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `File` - Path of file to write debug output.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PRIDebugFileUnset

PRIDebugFileUnset

Synopsis

Disables file output for PRI debug messages

Description

Syntax

```
Action: PRIDebugFileUnset
ActionID: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_PRIDebugSet

PRIDebugSet

Synopsis

Set PRI debug levels for a span

Description

Equivalent to the CLI command "pri set debug <level> span <span>".

Syntax

```
Action: PRIDebugSet
ActionID: <value>
Span: <value>
Level: <value>
```

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Span` - Which span to affect.
- `Level` - What debug level to set. May be a numerical value or a text value from the list below
    - `off`
    - `on`
    - `hex`
    - `intense`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - Queue's name.
- `Interface` - The name of the interface (tech/name) to add to the queue.
- `Penalty` - A penalty (number) to apply to this member. Asterisk will distribute calls to members with higher penalties only after attempting to distribute calls to those with lower penalty.
- `Paused` - To pause or not the member initially (true/false or 1/0).
- `MemberName` - Text alias for the interface.
- `StateInterface`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_QueuePause

QueuePause

Synopsis

Makes a queue member temporarily unavailable.

Description

Pause or unpause a member in a queue.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Interface` - The name of the interface (tech/name) to pause or unpause.
- `Paused` - Pause or unpause the interface. Set to 'true' to pause the member or 'false' to unpause.
- `Queue` - The name of the queue in which to pause or unpause this member. If not specified, the member will be paused or unpaused in all the queues it is a member of.
- `Reason` - Text description, returned in the event QueueMemberPaused.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_QueuePenalty

QueuePenalty

Synopsis

Set the penalty for a queue member.

Description

Change the penalty of a queue member

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Interface` - The interface (tech/name) of the member whose penalty to change.
- `Penalty` - The new penalty (number) for the member. Must be nonnegative.
- `Queue` - If specified, only set the penalty for the member of this queue. Otherwise, set the penalty for the member in all queues to which the member belongs.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - The name of the queue to take action on. If no queue name is specified, then all queues are affected.
- `Members` - Whether to reload the queue's members.
    - `yes`
    - `no`
- `Rules` - Whether to reload queuerules.conf
    - `yes`
    - `no`
- `Parameters` - Whether to reload the other queue options.
    - `yes`
    - `no`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - The name of the queue to take action on.
- `Interface` - The interface (tech/name) to remove from queue.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_QueueReset

QueueReset

Synopsis

Reset queue statistics.

Description

Reset the statistics for a queue.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue` - The name of the queue on which to reset statistics.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_QueueRule

QueueRule

Synopsis

Queue Rules.

Description

List queue rules defined in queuerules.conf

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Rule` - The name of the rule in queuerules.conf whose contents to list.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_Queues

Queues

Synopsis

Queues.

Description

Show queues information.

Syntax

```
Action: Queues
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_QueueStatus

QueueStatus

Synopsis

Show queue status.

Description

Check the status of one or more queues.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue` - Limit the response to the status of the specified queue.
- `Member` - Limit the response to the status of the specified member.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_QueueSummary

QueueSummary

Synopsis

Show queue summary.

Description

Request the manager to send a QueueSummary event.

Syntax

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

Arguments

- `ActionID` - ActionID for this transaction. Will be returned.
- `Queue` - Queue for which the summary is requested.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_Setvar

Setvar

Synopsis

Sets a channel variable or function value.

Description

This command can be used to set the value of channel variables or dialplan functions.

> **Note**
> If a channel name is not provided then the variable is considered 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, function or expression.
- `Value` - Variable or function value.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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*

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

- Asterisk 13 ManagerEvent_SIPQualifyPeerDone

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_VoicemailRefresh

VoicemailRefresh

Synopsis

Tell Asterisk to poll mailboxes for a change

Description

Normally, MWI indicators are only sent when Asterisk itself changes a mailbox. With external programs that modify the content of a mailbox from outside the application, an option exists called `pollmailboxes` that will cause voicemail to continually scan all mailboxes on a system for changes. This can cause a large amount of load on a system. This command allows external applications to signal when a particular mailbox has changed, thus permitting external applications to modify mailboxes and MWI to work without introducing considerable CPU load.

If *Context* is not specified, all mailboxes on the system will be polled for changes. If *Context* is specified, but *Mailbox* is omitted, then all mailboxes within *C ontext* will be polled. Otherwise, only a single mailbox will be polled for changes.

Syntax

```
Action: VoicemailRefresh
ActionID: <value>
Context: <value>
Mailbox: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerAction_VoicemailUsersList

VoicemailUsersList

Synopsis

List All Voicemail User Information.

Description

Syntax

```
Action: VoicemailUsersList
ActionID: <value>
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 AMI Events

Asterisk 13 ManagerEvent_AgentCalled

AgentCalled

Synopsis

Raised when an queue member is notified of a caller in the queue.

Description

Syntax

```
Event: AgentCalled
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
Queue: <value>
MemberName: <value>
Interface: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`

    * `Ringing`
    * `Up`
    * `Busy`
    * `Dialing Offhook`
    * `Pre-ring`
    * `Unknown`
* `DestCallerIDNum`
* `DestCallerIDName`
* `DestConnectedLineNum`
* `DestConnectedLineName`
* `DestAccountCode`
* `DestContext`
* `DestExten`
* `DestPriority`
* `DestUniqueid`
* `Queue` - The name of the queue.
* `MemberName` - The name of the queue member.
* `Interface` - The queue member's channel technology or location.

Class

AGENT

See Also

* Asterisk 13 ManagerEvent_AgentRingNoAnswer
* Asterisk 13 ManagerEvent_AgentComplete
* Asterisk 13 ManagerEvent_AgentConnect

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentComplete

AgentComplete

Synopsis

Raised when a queue member has finished servicing a caller in the queue.

Description

Syntax

```
Event: AgentComplete
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
Queue: <value>
MemberName: <value>
Interface: <value>
HoldTime: <value>
TalkTime: <value>
Reason: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
    - `Down`
    - `Rsrvd`

        - OffHook
        - Dialing
        - Ring
        - Ringing
        - Up
        - Busy
        - Dialing Offhook
        - Pre-ring
        - Unknown
- DestCallerIDNum
- DestCallerIDName
- DestConnectedLineNum
- DestConnectedLineName
- DestAccountCode
- DestContext
- DestExten
- DestPriority
- DestUniqueid
- Queue - The name of the queue.
- MemberName - The name of the queue member.
- Interface - The queue member's channel technology or location.
- HoldTime - The time the channel was in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- TalkTime - The time the queue member talked with the caller in the queue, expressed in seconds since 00:00, Jan 1, 1970 UTC.
- Reason
    - caller
    - agent
    - transfer

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_AgentCalled
- Asterisk 13 ManagerEvent_AgentConnect

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentConnect

AgentConnect

Synopsis

Raised when a queue member answers and is bridged to a caller in the queue.

Description

Syntax

```
Event: AgentConnect
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
Queue: <value>
MemberName: <value>
Interface: <value>
RingTime: <value>
HoldTime: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`

    * `Dialing`
    * `Ring`
    * `Ringing`
    * `Up`
    * `Busy`
    * `Dialing Offhook`
    * `Pre-ring`
    * `Unknown`
* `DestCallerIDNum`
* `DestCallerIDName`
* `DestConnectedLineNum`
* `DestConnectedLineName`
* `DestAccountCode`
* `DestContext`
* `DestExten`
* `DestPriority`
* `DestUniqueid`
* `Queue` - The name of the queue.
* `MemberName` - The name of the queue member.
* `Interface` - The queue member's channel technology or location.
* `RingTime` - The time the queue member 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.

Class

AGENT

See Also

* Asterisk 13 ManagerEvent_AgentCalled
* Asterisk 13 ManagerEvent_AgentComplete
* Asterisk 13 ManagerEvent_AgentDump

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentDump

AgentDump

Synopsis

Raised when a queue member hangs up on a caller in the queue.

Description

Syntax

```
Event: AgentDump
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
Queue: <value>
MemberName: <value>
Interface: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`

    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `DestCallerIDNum`
- `DestCallerIDName`
- `DestConnectedLineNum`
- `DestConnectedLineName`
- `DestAccountCode`
- `DestContext`
- `DestExten`
- `DestPriority`
- `DestUniqueid`
- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_AgentCalled
- Asterisk 13 ManagerEvent_AgentConnect

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentLogin

AgentLogin

Synopsis

Raised when an Agent has logged in.

Description

Syntax

```
Event: AgentLogin
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Agent: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Agent` - Agent ID of the agent.

Class

AGENT

See Also

- Asterisk 13 Application_AgentLogin
- Asterisk 13 ManagerEvent_AgentLogoff

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentLogoff

AgentLogoff

Synopsis

Raised when an Agent has logged off.

Description

Syntax

```
Event: AgentLogoff
Agent: <value>
Logintime: <value>
```

Arguments

- `Agent` - Agent ID of the agent.
- `Logintime` - The number of seconds the agent was logged in.

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_AgentLogin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentRingNoAnswer

AgentRingNoAnswer

Synopsis

Raised when a queue member is notified of a caller in the queue and fails to answer.

Description

Syntax

```
Event: AgentRingNoAnswer
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
Queue: <value>
MemberName: <value>
Interface: <value>
RingTime: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
  - `Down`
  - `Rsrvd`
  - `OffHook`
  - `Dialing`

        - `Ring`
        - `Ringing`
        - `Up`
        - `Busy`
        - `Dialing Offhook`
        - `Pre-ring`
        - `Unknown`
    - `DestCallerIDNum`
    - `DestCallerIDName`
    - `DestConnectedLineNum`
    - `DestConnectedLineName`
    - `DestAccountCode`
    - `DestContext`
    - `DestExten`
    - `DestPriority`
    - `DestUniqueid`
    - `Queue` - The name of the queue.
    - `MemberName` - The name of the queue member.
    - `Interface` - The queue member's channel technology or location.
    - `RingTime` - The time the queue member was rung, expressed in seconds since 00:00, Jan 1, 1970 UTC.

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_AgentCalled

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Agents

Agents

Synopsis

Response event in a series to the Agents AMI action containing information about a defined agent.

Description

The channel snapshot is present if the Status value is `AGENT_IDLE` or `AGENT_ONCALL`.

Syntax

```
Event: Agents
Agent: <value>
Name: <value>
Status: <value>
TalkingToChan: <value>
CallStarted: <value>
LoggedInTime: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
ActionID: <value>
```

Arguments

- `Agent` - Agent ID of the agent.
- `Name` - User friendly name of the agent.
- `Status` - Current status of the agent.
  The valid values are:
    - `AGENT_LOGGEDOFF`
    - `AGENT_IDLE`
    - `AGENT_ONCALL`
- `TalkingToChan` - BRIDGEPEER value on agent channel.
  Present if Status value is `AGENT_ONCALL`.
- `CallStarted` - Epoche time when the agent started talking with the caller.
  Present if Status value is `AGENT_ONCALL`.
- `LoggedInTime` - Epoche time when the agent logged in.
  Present if Status value is `AGENT_IDLE` or `AGENT_ONCALL`.
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`

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

Class

AGENT

See Also

* Asterisk 13 ManagerAction_Agents

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AgentsComplete

AgentsComplete

Synopsis

Final response event in a series of events to the Agents AMI action.

Description

Syntax

```
Event: AgentsComplete
ActionID: <value>
```

Arguments

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

Class

AGENT

See Also

- Asterisk 13 ManagerAction_Agents

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AGIExecEnd

AGIExecEnd

Synopsis

Raised when a received AGI command completes processing.

Description

Syntax

```
Event: AGIExecEnd
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Command: <value>
CommandId: <value>
ResultCode: <value>
Result: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Command` - The AGI command as received from the external source.
- `CommandId` - Random identification number assigned to the execution of this command.
- `ResultCode` - The numeric result code from AGI
- `Result` - The text result reason from AGI

Class

AGI

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AGIExecStart

AGIExecStart

Synopsis

Raised when a received AGI command starts processing.

Description

Syntax

```
Event: AGIExecStart
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Command: <value>
CommandId: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Command` - The AGI command as received from the external source.
- `CommandId` - Random identification number assigned to the execution of this command.

Class

AGI

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Alarm

Alarm

Synopsis

Raised when an alarm is set on a DAHDI channel.

Description

Syntax

```
Event: Alarm
DAHDIChannel: <value>
Alarm: <value>
```

Arguments

- `DAHDIChannel` - The channel on which the alarm occurred.

> **Note**
> This is not an Asterisk channel identifier.

- `Alarm` - A textual description of the alarm that occurred.

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AlarmClear

AlarmClear

Synopsis

Raised when an alarm is cleared on a DAHDI channel.

Description

Syntax

```
Event: AlarmClear
DAHDIChannel: <value>
```

Arguments

- `DAHDIChannel` - The DAHDI channel on which the alarm was cleared.

> **Note**
> This is not an Asterisk channel identifier.

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AOC-D

AOC-D

Synopsis

Raised when an Advice of Charge message is sent during a call.

Description

Syntax

```
Event: AOC-D
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Charge: <value>
Type: <value>
BillingID: <value>
TotalType: <value>
Currency: <value>
Name: <value>
Cost: <value>
Multiplier: <value>
Units: <value>
NumberOf: <value>
TypeOf: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Charge`
- `Type`
    - `NotAvailable`
    - `Free`
    - `Currency`
    - `Units`
- `BillingID`
    - `Normal`
    - `Reverse`
    - `CreditCard`
    - `CallForwardingUnconditional`

    - CallForwardingBusy
    - CallForwardingNoReply
    - CallDeflection
    - CallTransfer
    - NotAvailable
- TotalType
    - SubTotal
    - Total
- Currency
- Name
- Cost
- Multiplier
    - 1/1000
    - 1/100
    - 1/10
    - 1
    - 10
    - 100
    - 1000
- Units
- NumberOf
- TypeOf

Class

AOC

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AOC-E

AOC-E

Synopsis

Raised when an Advice of Charge message is sent at the end of a call.

Description

Syntax

```
Event: AOC-E
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
ChargingAssociation: <value>
Number: <value>
Plan: <value>
ID: <value>
Charge: <value>
Type: <value>
BillingID: <value>
TotalType: <value>
Currency: <value>
Name: <value>
Cost: <value>
Multiplier: <value>
Units: <value>
NumberOf: <value>
TypeOf: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `ChargingAssociation`
- `Number`
- `Plan`
- `ID`
- `Charge`
- `Type`
    - `NotAvailable`
    - `Free`

    - Currency
    - Units
- BillingID
    - Normal
    - Reverse
    - CreditCard
    - CallForwardingUnconditional
    - CallForwardingBusy
    - CallForwardingNoReply
    - CallDeflection
    - CallTransfer
    - NotAvailable
- TotalType
    - SubTotal
    - Total
- Currency
- Name
- Cost
- Multiplier
    - 1/1000
    - 1/100
    - 1/10
    - 1
    - 10
    - 100
    - 1000
- Units
- NumberOf
- TypeOf

Class

AOC

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AOC-S

AOC-S

Synopsis

Raised when an Advice of Charge message is sent at the beginning of a call.

Description

Syntax

```
Event: AOC-S
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Chargeable: <value>
RateType: <value>
Currency: <value>
Name: <value>
Cost: <value>
Multiplier: <value>
ChargingType: <value>
StepFunction: <value>
Granularity: <value>
Length: <value>
Scale: <value>
Unit: <value>
SpecialCode: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Chargeable`
- `RateType`
    - `NotAvailable`
    - `Free`
    - `FreeFromBeginning`
    - `Duration`
    - `Flag`
    - `Volume`
    - `SpecialCode`

- `Currency`
- `Name`
- `Cost`
- `Multiplier`
    - `1/1000`
    - `1/100`
    - `1/10`
    - `1`
    - `10`
    - `100`
    - `1000`
- `ChargingType`
- `StepFunction`
- `Granularity`
- `Length`
- `Scale`
- `Unit`
    - `Octect`
    - `Segment`
    - `Message`
- `SpecialCode`

Class

AOC

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AorDetail

AorDetail

Synopsis

Provide details about an Address of Record (AoR) section.

Description

Syntax

```
Event: AorDetail
ObjectType: <value>
ObjectName: <value>
MinimumExpiration: <value>
MaximumExpiration: <value>
DefaultExpiration: <value>
QualifyFrequency: <value>
AuthenticateQualify: <value>
MaxContacts: <value>
RemoveExisting: <value>
Mailboxes: <value>
OutboundProxy: <value>
SupportPath: <value>
TotalContacts: <value>
ContactsRegistered: <value>
EndpointName: <value>
```

Arguments

- `ObjectType` - The object's type. This will always be 'aor'.
- `ObjectName` - The name of this object.
- `MinimumExpiration` - Minimum keep alive time for an AoR
- `MaximumExpiration` - Maximum time to keep an AoR
- `DefaultExpiration` - Default expiration time in seconds for contacts that are dynamically bound to an AoR.
- `QualifyFrequency` - Interval at which to qualify an AoR
- `AuthenticateQualify` - Authenticates a qualify request if needed
- `MaxContacts` - Maximum number of contacts that can bind to an AoR
- `RemoveExisting` - Determines whether new contacts replace existing ones.
- `Mailboxes` - Allow subscriptions for the specified mailbox(es)
- `OutboundProxy` - Outbound proxy used when sending OPTIONS request
- `SupportPath` - Enables Path support for REGISTER requests and Route support for other requests.
- `TotalContacts` - The total number of contacts associated with this AoR.
- `ContactsRegistered` - The number of non-permanent contacts associated with this AoR.
- `EndpointName` - The name of the endpoint associated with this information.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AsyncAGIEnd

AsyncAGIEnd

Synopsis

Raised when a channel stops AsyncAGI command processing.

Description

Syntax

```
Event: AsyncAGIEnd
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

AGI

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AsyncAGIExec

AsyncAGIExec

Synopsis

Raised when AsyncAGI completes an AGI command.

Description

Syntax

```
Event: AsyncAGIExec
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
[CommandID:] <value>
Result: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `CommandID` - Optional command ID sent by the AsyncAGI server to identify the command.
- `Result` - URL encoded result string from the executed AGI command.

Class

AGI

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AsyncAGIStart

AsyncAGIStart

Synopsis

Raised when a channel starts AsyncAGI command processing.

Description

Syntax

```
Event: AsyncAGIStart
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Env: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Env` - URL encoded string read from the AsyncAGI server.

Class

AGI

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AttendedTransfer

AttendedTransfer

Synopsis

Raised when an attended transfer is complete.

Description

The headers in this event attempt to describe all the major details of the attended transfer. The two transferer channels and the two bridges are determined based on their chronological establishment. So consider that Alice calls Bob, and then Alice transfers the call to Voicemail. The transferer and bridge headers would be arranged as follows:

OrigTransfererChannel: Alice's channel in the bridge with Bob.

OrigBridgeUniqueid: The bridge between Alice and Bob.

SecondTransfererChannel: Alice's channel that called Voicemail.

SecondBridgeUniqueid: Not present, since a call to Voicemail has no bridge.

Now consider if the order were reversed; instead of having Alice call Bob and transfer him to Voicemail, Alice instead calls her Voicemail and transfers that to Bob. The transferer and bridge headers would be arranged as follows:

OrigTransfererChannel: Alice's channel that called Voicemail.

OrigBridgeUniqueid: Not present, since a call to Voicemail has no bridge.

SecondTransfererChannel: Alice's channel in the bridge with Bob.

SecondBridgeUniqueid: The bridge between Alice and Bob.

Syntax


```
Event: AttendedTransfer
Result: <value>
OrigTransfererChannel: <value>
OrigTransfererChannelState: <value>
OrigTransfererChannelStateDesc: <value>
OrigTransfererCallerIDNum: <value>
OrigTransfererCallerIDName: <value>
OrigTransfererConnectedLineNum: <value>
OrigTransfererConnectedLineName: <value>
OrigTransfererAccountCode: <value>
OrigTransfererContext: <value>
OrigTransfererExten: <value>
OrigTransfererPriority: <value>
OrigTransfererUniqueid: <value>
OrigBridgeUniqueid: <value>
OrigBridgeType: <value>
OrigBridgeTechnology: <value>
OrigBridgeCreator: <value>
OrigBridgeName: <value>
OrigBridgeNumChannels: <value>
SecondTransfererChannel: <value>
SecondTransfererChannelState: <value>
SecondTransfererChannelStateDesc: <value>
SecondTransfererCallerIDNum: <value>
SecondTransfererCallerIDName: <value>
SecondTransfererConnectedLineNum: <value>
SecondTransfererConnectedLineName: <value>
SecondTransfererAccountCode: <value>
SecondTransfererContext: <value>
SecondTransfererExten: <value>
SecondTransfererPriority: <value>
SecondTransfererUniqueid: <value>
SecondBridgeUniqueid: <value>
SecondBridgeType: <value>
SecondBridgeTechnology: <value>
SecondBridgeCreator: <value>
SecondBridgeName: <value>
SecondBridgeNumChannels: <value>
DestType: <value>
DestBridgeUniqueid: <value>
DestApp: <value>
LocalOneChannel: <value>
LocalOneChannelState: <value>
LocalOneChannelStateDesc: <value>
LocalOneCallerIDNum: <value>
LocalOneCallerIDName: <value>
LocalOneConnectedLineNum: <value>
LocalOneConnectedLineName: <value>
LocalOneAccountCode: <value>
LocalOneContext: <value>
LocalOneExten: <value>
LocalOnePriority: <value>
LocalOneUniqueid: <value>
LocalTwoChannel: <value>
LocalTwoChannelState: <value>
LocalTwoChannelStateDesc: <value>
LocalTwoCallerIDNum: <value>
LocalTwoCallerIDName: <value>
LocalTwoConnectedLineNum: <value>
LocalTwoConnectedLineName: <value>
LocalTwoAccountCode: <value>
LocalTwoContext: <value>
LocalTwoExten: <value>
LocalTwoPriority: <value>
LocalTwoUniqueid: <value>
DestTransfererChannel: <value>
TransfereeChannel: <value>
TransfereeChannelState: <value>
TransfereeChannelStateDesc: <value>
TransfereeCallerIDNum: <value>
TransfereeCallerIDName: <value>
TransfereeConnectedLineNum: <value>
TransfereeConnectedLineName: <value>
TransfereeAccountCode: <value>
TransfereeContext: <value>
TransfereeExten: <value>
TransfereePriority: <value>
TransfereeUniqueid: <value>
```

Arguments

- `Result` - Indicates if the transfer was successful or if it failed.
    - `Fail` - An internal error occurred.
    - `Invalid` - Invalid configuration for transfer (e.g. Not bridged)
    - `Not Permitted` - Bridge does not permit transfers

        - `Success` - Transfer completed successfully

> **Note**
> A result of `Success` does not necessarily mean that a target was succesfully contacted. It means that a party was succesfully placed into the dialplan at the expected location.

- `OrigTransfererChannel`
- `OrigTransfererChannelState` - A numeric code for the channel's current state, related to OrigTransfererChannelStateDesc
- `OrigTransfererChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `OrigTransfererCallerIDNum`
- `OrigTransfererCallerIDName`
- `OrigTransfererConnectedLineNum`
- `OrigTransfererConnectedLineName`
- `OrigTransfererAccountCode`
- `OrigTransfererContext`
- `OrigTransfererExten`
- `OrigTransfererPriority`
- `OrigTransfererUniqueid`
- `OrigBridgeUniqueid`
- `OrigBridgeType` - The type of bridge
- `OrigBridgeTechnology` - Technology in use by the bridge
- `OrigBridgeCreator` - Entity that created the bridge if applicable
- `OrigBridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `OrigBridgeNumChannels` - Number of channels in the bridge
- `SecondTransfererChannel`
- `SecondTransfererChannelState` - A numeric code for the channel's current state, related to SecondTransfererChannelStateDesc
- `SecondTransfererChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `SecondTransfererCallerIDNum`
- `SecondTransfererCallerIDName`
- `SecondTransfererConnectedLineNum`
- `SecondTransfererConnectedLineName`
- `SecondTransfererAccountCode`
- `SecondTransfererContext`
- `SecondTransfererExten`
- `SecondTransfererPriority`
- `SecondTransfererUniqueid`
- `SecondBridgeUniqueid`
- `SecondBridgeType` - The type of bridge
- `SecondBridgeTechnology` - Technology in use by the bridge
- `SecondBridgeCreator` - Entity that created the bridge if applicable
- `SecondBridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `SecondBridgeNumChannels` - Number of channels in the bridge
- `DestType` - Indicates the method by which the attended transfer completed.
    - `Bridge` - The transfer was accomplished by merging two bridges into one.
    - `App` - The transfer was accomplished by having a channel or bridge run a dialplan application.
    - `Link` - The transfer was accomplished by linking two bridges together using a local channel pair.
    - `Threeway` - The transfer was accomplished by placing all parties into a threeway call.
    - `Fail` - The transfer failed.

- `DestBridgeUniqueid` - Indicates the surviving bridge when bridges were merged to complete the transfer

> **Note**
> This header is only present when *DestType* is `Bridge` or `Threeway`

- `DestApp` - Indicates the application that is running when the transfer completes

> **Note**
> This header is only present when *DestType* is App

- `LocalOneChannel`
- `LocalOneChannelState` - A numeric code for the channel's current state, related to LocalOneChannelStateDesc
- `LocalOneChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalOneCallerIDNum`
- `LocalOneCallerIDName`
- `LocalOneConnectedLineNum`
- `LocalOneConnectedLineName`
- `LocalOneAccountCode`
- `LocalOneContext`
- `LocalOneExten`
- `LocalOnePriority`
- `LocalOneUniqueid`
- `LocalTwoChannel`
- `LocalTwoChannelState` - A numeric code for the channel's current state, related to LocalTwoChannelStateDesc
- `LocalTwoChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalTwoCallerIDNum`
- `LocalTwoCallerIDName`
- `LocalTwoConnectedLineNum`
- `LocalTwoConnectedLineName`
- `LocalTwoAccountCode`
- `LocalTwoContext`
- `LocalTwoExten`
- `LocalTwoPriority`
- `LocalTwoUniqueid`
- `DestTransfererChannel` - The name of the surviving transferer channel when a transfer results in a threeway call

> **Note**
> This header is only present when *DestType* is `Threeway`

- `TransfereeChannel`
- `TransfereeChannelState` - A numeric code for the channel's current state, related to TransfereeChannelStateDesc
- `TransfereeChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`

        - `Up`
        - `Busy`
        - `Dialing Offhook`
        - `Pre-ring`
        - `Unknown`
- `TransfereeCallerIDNum`
- `TransfereeCallerIDName`
- `TransfereeConnectedLineNum`
- `TransfereeConnectedLineName`
- `TransfereeAccountCode`
- `TransfereeContext`
- `TransfereeExten`
- `TransfereePriority`
- `TransfereeUniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AuthDetail

AuthDetail

Synopsis

Provide details about an authentication section.

Description

Syntax

```
Event: AuthDetail
ObjectType: <value>
ObjectName: <value>
Username: <value>
Password: <value>
Md5Cred: <value>
Realm: <value>
NonceLifetime: <value>
AuthType: <value>
EndpointName: <value>
```

Arguments

- `ObjectType` - The object's type. This will always be 'auth'.
- `ObjectName` - The name of this object.
- `Username` - Username to use for account
- `Password` - Username to use for account
- `Md5Cred` - MD5 Hash used for authentication.
- `Realm` - SIP realm for endpoint
- `NonceLifetime` - Lifetime of a nonce associated with this authentication config.
- `AuthType` - Authentication type
- `EndpointName` - The name of the endpoint associated with this information.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_AuthMethodNotAllowed

AuthMethodNotAllowed

Synopsis

Raised when a request used an authentication method not allowed by the service.

Description

Syntax

```
Event: AuthMethodNotAllowed
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
AuthMethod: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
  - `Informational`
  - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `AuthMethod` - The authentication method attempted.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BlindTransfer

BlindTransfer

Synopsis

Raised when a blind transfer is complete.

Description

Syntax

```
Event: BlindTransfer
Result: <value>
TransfererChannel: <value>
TransfererChannelState: <value>
TransfererChannelStateDesc: <value>
TransfererCallerIDNum: <value>
TransfererCallerIDName: <value>
TransfererConnectedLineNum: <value>
TransfererConnectedLineName: <value>
TransfererAccountCode: <value>
TransfererContext: <value>
TransfererExten: <value>
TransfererPriority: <value>
TransfererUniqueid: <value>
TransfereeChannel: <value>
TransfereeChannelState: <value>
TransfereeChannelStateDesc: <value>
TransfereeCallerIDNum: <value>
TransfereeCallerIDName: <value>
TransfereeConnectedLineNum: <value>
TransfereeConnectedLineName: <value>
TransfereeAccountCode: <value>
TransfereeContext: <value>
TransfereeExten: <value>
TransfereePriority: <value>
TransfereeUniqueid: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
IsExternal: <value>
Context: <value>
Extension: <value>
```

Arguments

- `Result` - Indicates if the transfer was successful or if it failed.
    - `Fail` - An internal error occurred.
    - `Invalid` - Invalid configuration for transfer (e.g. Not bridged)
    - `Not Permitted` - Bridge does not permit transfers
    - `Success` - Transfer completed successfully

    > **Note**
    > A result of `Success` does not necessarily mean that a target was succesfully contacted. It means that a party was succesfully placed into the dialplan at the expected location.

- `TransfererChannel`
- `TransfererChannelState` - A numeric code for the channel's current state, related to TransfererChannelStateDesc
- `TransfererChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`

- `TransfererCallerIDNum`
- `TransfererCallerIDName`
- `TransfererConnectedLineNum`
- `TransfererConnectedLineName`
- `TransfererAccountCode`
- `TransfererContext`
- `TransfererExten`
- `TransfererPriority`
- `TransfererUniqueid`
- `TransfereeChannel`
- `TransfereeChannelState` - A numeric code for the channel's current state, related to TransfereeChannelStateDesc
- `TransfereeChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `TransfereeCallerIDNum`
- `TransfereeCallerIDName`
- `TransfereeConnectedLineNum`
- `TransfereeConnectedLineName`
- `TransfereeAccountCode`
- `TransfereeContext`
- `TransfereeExten`
- `TransfereePriority`
- `TransfereeUniqueid`
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `IsExternal` - Indicates if the transfer was performed outside of Asterisk. For instance, a channel protocol native transfer is external. A DTMF transfer is internal.
    - `Yes`
    - `No`
- `Context` - Destination context for the blind transfer.
- `Extension` - Destination extension for the blind transfer.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BridgeCreate

BridgeCreate

Synopsis

Raised when a bridge is created.

Description

Syntax

```
Event: BridgeCreate
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BridgeDestroy

BridgeDestroy

Synopsis

Raised when a bridge is destroyed.

Description

Syntax

```
Event: BridgeDestroy
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BridgeEnter

BridgeEnter

Synopsis

Raised when a channel enters a bridge.

Description

Syntax

```
Event: BridgeEnter
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
SwapUniqueid: <value>
```

Arguments

- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `SwapUniqueid` - The uniqueid of the channel being swapped out of the bridge

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BridgeInfoChannel

BridgeInfoChannel

Synopsis

Information about a channel in a bridge.

Description

Syntax

```
Event: BridgeInfoChannel
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BridgeInfoComplete

BridgeInfoComplete

Synopsis

Information about a bridge.

Description

Syntax

```
Event: BridgeInfoComplete
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_BridgeLeave

BridgeLeave

Synopsis

Raised when a channel leaves a bridge.

Description

Syntax

```
Event: BridgeLeave
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

Import Version

Asterisk 13 ManagerEvent_BridgeMerge

BridgeMerge

Synopsis

Raised when two bridges are merged.

Description

Syntax

```
Event: BridgeMerge
ToBridgeUniqueid: <value>
ToBridgeType: <value>
ToBridgeTechnology: <value>
ToBridgeCreator: <value>
ToBridgeName: <value>
ToBridgeNumChannels: <value>
FromBridgeUniqueid: <value>
FromBridgeType: <value>
FromBridgeTechnology: <value>
FromBridgeCreator: <value>
FromBridgeName: <value>
FromBridgeNumChannels: <value>
```

Arguments

- `ToBridgeUniqueid`
- `ToBridgeType` - The type of bridge
- `ToBridgeTechnology` - Technology in use by the bridge
- `ToBridgeCreator` - Entity that created the bridge if applicable
- `ToBridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `ToBridgeNumChannels` - Number of channels in the bridge
- `FromBridgeUniqueid`
- `FromBridgeType` - The type of bridge
- `FromBridgeTechnology` - Technology in use by the bridge
- `FromBridgeCreator` - Entity that created the bridge if applicable
- `FromBridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `FromBridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ChallengeResponseFailed

ChallengeResponseFailed

Synopsis

Raised when a request's attempt to authenticate has been challenged, and the request failed the authentication challenge.

Description

Syntax

```
Event: ChallengeResponseFailed
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
Challenge: <value>
Response: <value>
ExpectedResponse: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Challenge` - The challenge that was sent.
- `Response` - The response that was received.
- `ExpectedResponse` - The expected response to the challenge.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ChallengeSent

ChallengeSent

Synopsis

Raised when an Asterisk service sends an authentication challenge to a request.

Description

Syntax

```
Event: ChallengeSent
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
Challenge: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Challenge` - The challenge that was sent.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ChannelTalkingStart

ChannelTalkingStart

Synopsis

Raised when talking is detected on a channel.

Description

Syntax

```
Event: ChannelTalkingStart
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CLASS

See Also

- Asterisk 13 Function_TALK_DETECT
- Asterisk 13 ManagerEvent_ChannelTalkingStop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ChannelTalkingStop

ChannelTalkingStop

Synopsis

Raised when talking is no longer detected on a channel.

Description

Syntax

```
Event: ChannelTalkingStop
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Duration: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Duration` - The length in time, in milliseconds, that talking was detected on the channel.

Class

CLASS

See Also

- Asterisk 13 Function_TALK_DETECT
- Asterisk 13 ManagerEvent_ChannelTalkingStart

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ChanSpyStart

ChanSpyStart

Synopsis

Raised when one channel begins spying on another channel.

Description

Syntax

```
Event: ChanSpyStart
SpyerChannel: <value>
SpyerChannelState: <value>
SpyerChannelStateDesc: <value>
SpyerCallerIDNum: <value>
SpyerCallerIDName: <value>
SpyerConnectedLineNum: <value>
SpyerConnectedLineName: <value>
SpyerAccountCode: <value>
SpyerContext: <value>
SpyerExten: <value>
SpyerPriority: <value>
SpyerUniqueid: <value>
SpyeeChannel: <value>
SpyeeChannelState: <value>
SpyeeChannelStateDesc: <value>
SpyeeCallerIDNum: <value>
SpyeeCallerIDName: <value>
SpyeeConnectedLineNum: <value>
SpyeeConnectedLineName: <value>
SpyeeAccountCode: <value>
SpyeeContext: <value>
SpyeeExten: <value>
SpyeePriority: <value>
SpyeeUniqueid: <value>
```

Arguments

- `SpyerChannel`
- `SpyerChannelState` - A numeric code for the channel's current state, related to SpyerChannelStateDesc
- `SpyerChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `SpyerCallerIDNum`
- `SpyerCallerIDName`
- `SpyerConnectedLineNum`
- `SpyerConnectedLineName`
- `SpyerAccountCode`
- `SpyerContext`
- `SpyerExten`
- `SpyerPriority`
- `SpyerUniqueid`
- `SpyeeChannel`
- `SpyeeChannelState` - A numeric code for the channel's current state, related to SpyeeChannelStateDesc
- `SpyeeChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`

    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `SpyeeCallerIDNum`
- `SpyeeCallerIDName`
- `SpyeeConnectedLineNum`
- `SpyeeConnectedLineName`
- `SpyeeAccountCode`
- `SpyeeContext`
- `SpyeeExten`
- `SpyeePriority`
- `SpyeeUniqueid`

Class

CALL

See Also

- Asterisk 13 Application_ChanSpyStop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ChanSpyStop

ChanSpyStop

Synopsis

Raised when a channel has stopped spying.

Description

Syntax

```
Event: ChanSpyStop
SpyerChannel: <value>
SpyerChannelState: <value>
SpyerChannelStateDesc: <value>
SpyerCallerIDNum: <value>
SpyerCallerIDName: <value>
SpyerConnectedLineNum: <value>
SpyerConnectedLineName: <value>
SpyerAccountCode: <value>
SpyerContext: <value>
SpyerExten: <value>
SpyerPriority: <value>
SpyerUniqueid: <value>
SpyeeChannel: <value>
SpyeeChannelState: <value>
SpyeeChannelStateDesc: <value>
SpyeeCallerIDNum: <value>
SpyeeCallerIDName: <value>
SpyeeConnectedLineNum: <value>
SpyeeConnectedLineName: <value>
SpyeeAccountCode: <value>
SpyeeContext: <value>
SpyeeExten: <value>
SpyeePriority: <value>
SpyeeUniqueid: <value>
```

Arguments

- `SpyerChannel`
- `SpyerChannelState` - A numeric code for the channel's current state, related to SpyerChannelStateDesc
- `SpyerChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `SpyerCallerIDNum`
- `SpyerCallerIDName`
- `SpyerConnectedLineNum`
- `SpyerConnectedLineName`
- `SpyerAccountCode`
- `SpyerContext`
- `SpyerExten`
- `SpyerPriority`
- `SpyerUniqueid`
- `SpyeeChannel`
- `SpyeeChannelState` - A numeric code for the channel's current state, related to SpyeeChannelStateDesc
- `SpyeeChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`

        - `Busy`
        - `Dialing Offhook`
        - `Pre-ring`
        - `Unknown`
    - `SpyeeCallerIDNum`
    - `SpyeeCallerIDName`
    - `SpyeeConnectedLineNum`
    - `SpyeeConnectedLineName`
    - `SpyeeAccountCode`
    - `SpyeeContext`
    - `SpyeeExten`
    - `SpyeePriority`
    - `SpyeeUniqueid`

Class

CALL

See Also

- Asterisk 13 Application_ChanSpyStart

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeEnd

ConfbridgeEnd

Synopsis

Raised when a conference ends.

Description

Syntax

```
Event: ConfbridgeEnd
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

- Asterisk 13 ManagerEvent_ConfbridgeStart
- Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeJoin

ConfbridgeJoin

Synopsis

Raised when a channel joins a Confbridge conference.

Description

Syntax

```
Event: ConfbridgeJoin
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

* Asterisk 13 ManagerEvent_ConfbridgeLeave
* Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeLeave

ConfbridgeLeave

Synopsis

Raised when a channel leaves a Confbridge conference.

Description

Syntax

```
Event: ConfbridgeLeave
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

* Asterisk 13 ManagerEvent_ConfbridgeJoin
* Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeMute

ConfbridgeMute

Synopsis

Raised when a Confbridge participant mutes.

Description

Syntax

```
Event: ConfbridgeMute
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

- Asterisk 13 ManagerEvent_ConfbridgeUnmute
- Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeRecord

ConfbridgeRecord

Synopsis

Raised when a conference starts recording.

Description

Syntax

```
Event: ConfbridgeRecord
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

- Asterisk 13 ManagerEvent_ConfbridgeStopRecord
- Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeStart

ConfbridgeStart

Synopsis

Raised when a conference starts.

Description

Syntax

```
Event: ConfbridgeStart
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

- Asterisk 13 ManagerEvent_ConfbridgeEnd
- Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeStopRecord

ConfbridgeStopRecord

Synopsis

Raised when a conference that was recording stops recording.

Description

Syntax

```
Event: ConfbridgeStopRecord
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge

Class

CALL

See Also

- Asterisk 13 ManagerEvent_ConfbridgeRecord
- Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeTalking

ConfbridgeTalking

Synopsis

Raised when a confbridge participant unmutes.

Description

Syntax

```
Event: ConfbridgeTalking
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
TalkingStatus: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `TalkingStatus`
    - `on`
    - `off`

Class

CALL

See Also

- Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ConfbridgeUnmute

ConfbridgeUnmute

Synopsis

Raised when a confbridge participant unmutes.

Description

Syntax

```
Event: ConfbridgeUnmute
Conference: <value>
BridgeUniqueid: <value>
BridgeType: <value>
BridgeTechnology: <value>
BridgeCreator: <value>
BridgeName: <value>
BridgeNumChannels: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Conference` - The name of the Confbridge conference.
- `BridgeUniqueid`
- `BridgeType` - The type of bridge
- `BridgeTechnology` - Technology in use by the bridge
- `BridgeCreator` - Entity that created the bridge if applicable
- `BridgeName` - Name used to refer to the bridge by its BridgeCreator if applicable
- `BridgeNumChannels` - Number of channels in the bridge
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

* Asterisk 13 ManagerEvent_ConfbridgeMute
* Asterisk 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ContactStatusDetail

ContactStatusDetail

Synopsis

Provide details about a contact's status.

Description

Syntax

```
Event: ContactStatusDetail
AOR: <value>
URI: <value>
Status: <value>
RoundtripUsec: <value>
EndpointName: <value>
```

Arguments

- `AOR` - The AoR that owns this contact.
- `URI` - This contact's URI.
- `Status` - This contact's status.
    - `Reachable`
    - `Unreachable`
- `RoundtripUsec` - The round trip time in microseconds.
- `EndpointName` - The name of the endpoint associated with this information.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DAHDIChannel

DAHDIChannel

Synopsis

Raised when a DAHDI channel is created or an underlying technology is associated with a DAHDI channel.

Description

Syntax

```
Event: DAHDIChannel
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DAHDISpan: <value>
DAHDIChannel: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DAHDISpan` - The DAHDI span associated with this channel.
- `DAHDIChannel` - The DAHDI channel associated with this channel.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DeviceStateChange

DeviceStateChange

Synopsis

Raised when a device state changes

Description

This differs from the `ExtensionStatus` event because this event is raised for all device state changes, not only for changes that affect dialplan hints.

Syntax

```
Event: DeviceStateChange
Device: <value>
State: <value>
```

Arguments

- `Device` - The device whose state has changed
- `State` - The new state of the device

Class

CALL

See Also

- Asterisk 13 ManagerEvent_ExtensionStatus

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 ManagerEvent_DeviceStateListComplete

DeviceStateListComplete

Synopsis

Indicates the end of the list the current known extension states.

Description

Syntax

```
Event: DeviceStateListComplete
EventList: <value>
ListItems: <value>
```

Arguments

- `EventList` - Conveys the status of the event list.
- `ListItems` - Conveys the number of statuses reported.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DialBegin

DialBegin

Synopsis

Raised when a dial action has started.

Description

Syntax

```
Event: DialBegin
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
DialString: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`

    * `Up`
    * `Busy`
    * `Dialing Offhook`
    * `Pre-ring`
    * `Unknown`
* `DestCallerIDNum`
* `DestCallerIDName`
* `DestConnectedLineNum`
* `DestConnectedLineName`
* `DestAccountCode`
* `DestContext`
* `DestExten`
* `DestPriority`
* `DestUniqueid`
* `DialString` - The non-technology specific device being dialed.

Class

CALL

See Also

* Asterisk 13 Application_Dial

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DialEnd

DialEnd

Synopsis

Raised when a dial action has completed.

Description

Syntax

```
Event: DialEnd
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
DestChannel: <value>
DestChannelState: <value>
DestChannelStateDesc: <value>
DestCallerIDNum: <value>
DestCallerIDName: <value>
DestConnectedLineNum: <value>
DestConnectedLineName: <value>
DestAccountCode: <value>
DestContext: <value>
DestExten: <value>
DestPriority: <value>
DestUniqueid: <value>
DialStatus: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `DestChannel`
- `DestChannelState` - A numeric code for the channel's current state, related to DestChannelStateDesc
- `DestChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`

    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `DestCallerIDNum`
- `DestCallerIDName`
- `DestConnectedLineNum`
- `DestConnectedLineName`
- `DestAccountCode`
- `DestContext`
- `DestExten`
- `DestPriority`
- `DestUniqueid`
- `DialStatus` - The result of the dial operation.
    - `ABORT` - The call was aborted.
    - `ANSWER` - The caller answered.
    - `BUSY` - The caller was busy.
    - `CANCEL` - The caller cancelled the call.
    - `CHANUNAVAIL` - The requested channel is unavailable.
    - `CONGESTION` - The called party is congested.
    - `CONTINUE` - The dial completed, but the caller elected to continue in the dialplan.
    - `GOTO` - The dial completed, but the caller jumped to a dialplan location.
      If known, the location the caller is jumping to will be appended to the result following a ":".
    - `NOANSWER` - The called party failed to answer.

Class

CALL

See Also

- Asterisk 13 Application_Dial

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DNDState

DNDState

Synopsis

Raised when the Do Not Disturb state is changed on a DAHDI channel.

Description

Syntax

```
Event: DNDState
DAHDIChannel: <value>
Status: <value>
```

Arguments

- `DAHDIChannel` - The DAHDI channel on which DND status changed.

  > **Note**
  > This is not an Asterisk channel identifier.

- `Status`
  - `enabled`
  - `disabled`

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DTMFBegin

DTMFBegin

Synopsis

Raised when a DTMF digit has started on a channel.

Description

Syntax

```
Event: DTMFBegin
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Digit: <value>
Direction: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Digit` - DTMF digit received or transmitted (0-9, A-E, # or *
- `Direction`
    - `Received`
    - `Sent`

Class

DTMF

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_DTMFEnd

DTMFEnd

Synopsis

Raised when a DTMF digit has ended on a channel.

Description

Syntax

```
Event: DTMFEnd
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Digit: <value>
DurationMs: <value>
Direction: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Digit` - DTMF digit received or transmitted (0-9, A-E, # or *
- `DurationMs` - Duration (in milliseconds) DTMF was sent/received
- `Direction`
    - `Received`
    - `Sent`

Class

DTMF

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_EndpointDetail

EndpointDetail

Synopsis

Provide details about an endpoint section.

Description

Syntax

```
Event: EndpointDetail
ObjectType: <value>
ObjectName: <value>
Context: <value>
Disallow: <value>
Allow: <value>
DtmfMode: <value>
RtpIpv6: <value>
RtpSymmetric: <value>
IceSupport: <value>
UsePtime: <value>
ForceRport: <value>
RewriteContact: <value>
Transport: <value>
OutboundProxy: <value>
MohSuggest: <value>
100rel: <value>
Timers: <value>
TimersMinSe: <value>
TimersSessExpires: <value>
Auth: <value>
OutboundAuth: <value>
Aors: <value>
MediaAddress: <value>
IdentifyBy: <value>
DirectMedia: <value>
DirectMediaMethod: <value>
ConnectedLineMethod: <value>
DirectMediaGlareMitigation: <value>
DisableDirectMediaOnNat: <value>
Callerid: <value>
CalleridPrivacy: <value>
CalleridTag: <value>
TrustIdInbound: <value>
TrustIdOutbound: <value>
SendPai: <value>
SendRpid: <value>
SendDiversion: <value>
Mailboxes: <value>
AggregateMwi: <value>
MediaEncryption: <value>
UseAvpf: <value>
ForceAvp: <value>
MediaUseReceivedTransport: <value>
OneTouchRecording: <value>
InbandProgress: <value>
CallGroup: <value>
PickupGroup: <value>
NamedCallGroup: <value>
NamedPickupGroup: <value>
DeviceStateBusyAt: <value>
T38Udptl: <value>
T38UdptlEc: <value>
T38UdptlMaxdatagram: <value>
FaxDetect: <value>
T38UdptlNat: <value>
T38UdptlIpv6: <value>
ToneZone: <value>
Language: <value>
RecordOnFeature: <value>
RecordOffFeature: <value>
AllowTransfer: <value>
SdpOwner: <value>
SdpSession: <value>
TosAudio: <value>
TosVideo: <value>
CosAudio: <value>
CosVideo: <value>
AllowSubscribe: <value>
SubMinExpiry: <value>
FromUser: <value>
```

```
FromDomain: <value>
MwiFromUser: <value>
RtpEngine: <value>
DtlsVerify: <value>
DtlsRekey: <value>
DtlsCertFile: <value>
DtlsPrivateKey: <value>
DtlsCipher: <value>
DtlsCaFile: <value>
DtlsCaPath: <value>
DtlsSetup: <value>
SrtpTag32: <value>
RedirectMethod: <value>
SetVar: <value>
MessageContext: <value>
Accountcode: <value>
```



```
DeviceState: <value>
ActiveChannels: <value>
```

Arguments

- `ObjectType` - The object's type. This will always be 'endpoint'.
- `ObjectName` - The name of this object.
- `Context` - Dialplan context for inbound sessions
- `Disallow` - Media Codec(s) to disallow
- `Allow` - Media Codec(s) to allow
- `DtmfMode` - DTMF mode
- `RtpIpv6` - Allow use of IPv6 for RTP traffic
- `RtpSymmetric` - Enforce that RTP must be symmetric
- `IceSupport` - Enable the ICE mechanism to help traverse NAT
- `UsePtime` - Use Endpoint's requested packetisation interval
- `ForceRport` - Force use of return port
- `RewriteContact` - Allow Contact header to be rewritten with the source IP address-port
- `Transport` - Desired transport configuration
- `OutboundProxy` - Proxy through which to send requests, a full SIP URI must be provided
- `MohSuggest` - Default Music On Hold class
- `100rel` - Allow support for RFC3262 provisional ACK tags
- `Timers` - Session timers for SIP packets
- `TimersMinSe` - Minimum session timers expiration period
- `TimersSessExpires` - Maximum session timer expiration period
- `Auth` - Authentication Object(s) associated with the endpoint
- `OutboundAuth` - Authentication object used for outbound requests
- `Aors` - AoR(s) to be used with the endpoint
- `MediaAddress` - IP address used in SDP for media handling
- `IdentifyBy` - Way(s) for Endpoint to be identified
- `DirectMedia` - Determines whether media may flow directly between endpoints.
- `DirectMediaMethod` - Direct Media method type
- `ConnectedLineMethod` - Connected line method type
- `DirectMediaGlareMitigation` - Mitigation of direct media (re)INVITE glare
- `DisableDirectMediaOnNat` - Disable direct media session refreshes when NAT obstructs the media session
- `Callerid` - CallerID information for the endpoint
- `CalleridPrivacy` - Default privacy level
- `CalleridTag` - Internal id_tag for the endpoint
- `TrustIdInbound` - Accept identification information received from this endpoint
- `TrustIdOutbound` - Send private identification details to the endpoint.
- `SendPai` - Send the P-Asserted-Identity header
- `SendRpid` - Send the Remote-Party-ID header
- `SendDiversion` - Send the Diversion header, conveying the diversion information to the called user agent
- `Mailboxes` - NOTIFY the endpoint when state changes for any of the specified mailboxes
- `AggregateMwi` - Condense MWI notifications into a single NOTIFY.
- `MediaEncryption` - Determines whether res_pjsip will use and enforce usage of media encryption for this endpoint.
- `UseAvpf` - Determines whether res_pjsip will use and enforce usage of AVPF for this endpoint.
- `ForceAvp` - Determines whether res_pjsip will use and enforce usage of AVP, regardless of the RTP profile in use for this endpoint.
- `MediaUseReceivedTransport` - Determines whether res_pjsip will use the media transport received in the offer SDP in the corresponding answer SDP.
- `OneTouchRecording` - Determines whether one-touch recording is allowed for this endpoint.
- `InbandProgress` - Determines whether chan_pjsip will indicate ringing using inband progress.
- `CallGroup` - The numeric pickup groups for a channel.
- `PickupGroup` - The numeric pickup groups that a channel can pickup.
- `NamedCallGroup` - The named pickup groups for a channel.
- `NamedPickupGroup` - The named pickup groups that a channel can pickup.
- `DeviceStateBusyAt` - The number of in-use channels which will cause busy to be returned as device state
- `T38Udptl` - Whether T.38 UDPTL support is enabled or not
- `T38UdptlEc` - T.38 UDPTL error correction method
- `T38UdptlMaxdatagram` - T.38 UDPTL maximum datagram size
- `FaxDetect` - Whether CNG tone detection is enabled
- `T38UdptlNat` - Whether NAT support is enabled on UDPTL sessions
- `T38UdptlIpv6` - Whether IPv6 is used for UDPTL Sessions
- `ToneZone` - Set which country's indications to use for channels created for this endpoint.
- `Language` - Set the default language to use for channels created for this endpoint.
- `RecordOnFeature` - The feature to enact when one-touch recording is turned on.
- `RecordOffFeature` - The feature to enact when one-touch recording is turned off.
- `AllowTransfer` - Determines whether SIP REFER transfers are allowed for this endpoint

- `SdpOwner` - String placed as the username portion of an SDP origin (o=) line.
- `SdpSession` - String used for the SDP session (s=) line.
- `TosAudio` - DSCP TOS bits for audio streams
- `TosVideo` - DSCP TOS bits for video streams
- `CosAudio` - Priority for audio streams
- `CosVideo` - Priority for video streams
- `AllowSubscribe` - Determines if endpoint is allowed to initiate subscriptions with Asterisk.
- `SubMinExpiry` - The minimum allowed expiry time for subscriptions initiated by the endpoint.
- `FromUser` - Username to use in From header for requests to this endpoint.
- `FromDomain` - Domain to user in From header for requests to this endpoint.
- `MwiFromUser` - Username to use in From header for unsolicited MWI NOTIFYs to this endpoint.
- `RtpEngine` - Name of the RTP engine to use for channels created for this endpoint
- `DtlsVerify` - Verify that the provided peer certificate is valid
- `DtlsRekey` - Interval at which to renegotiate the TLS session and rekey the SRTP session
- `DtlsCertFile` - Path to certificate file to present to peer
- `DtlsPrivateKey` - Path to private key for certificate file
- `DtlsCipher` - Cipher to use for DTLS negotiation
- `DtlsCaFile` - Path to certificate authority certificate
- `DtlsCaPath` - Path to a directory containing certificate authority certificates
- `DtlsSetup` - Whether we are willing to accept connections, connect to the other party, or both.
- `SrtpTag32` - Determines whether 32 byte tags should be used instead of 80 byte tags.
- `RedirectMethod` - How redirects received from an endpoint are handled
- `SetVar` - Variable set on a channel involving the endpoint.
- `MessageContext` - Context to route incoming MESSAGE requests to.
- `Accountcode` - An accountcode to set automatically on any channels created for this endpoint.
- `DeviceState` - The aggregate device state for this endpoint.
- `ActiveChannels` - The number of active channels associated with this endpoint.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 ManagerEvent_EndpointDetailComplete

EndpointDetailComplete

Synopsis

Provide final information about endpoint details.

Description

Syntax

```
Event: EndpointDetailComplete
EventList: <value>
ListItems: <value>
```

Arguments

- `EventList`
- `ListItems`

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_EndpointList

EndpointList

Synopsis

Provide details about a contact's status.

Description

Syntax

```
Event: EndpointList
ObjectType: <value>
ObjectName: <value>
Transport: <value>
Aor: <value>
Auths: <value>
OutboundAuths: <value>
DeviceState: <value>
ActiveChannels: <value>
```

Arguments

- `ObjectType` - The object's type. This will always be 'endpoint'.
- `ObjectName` - The name of this object.
- `Transport` - The transport configurations associated with this endpoint.
- `Aor` - The aor configurations associated with this endpoint.
- `Auths` - The inbound authentication configurations associated with this endpoint.
- `OutboundAuths` - The outbound authentication configurations associated with this endpoint.
- `DeviceState` - The aggregate device state for this endpoint.
- `ActiveChannels` - The number of active channels associated with this endpoint.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_EndpointListComplete

EndpointListComplete

Synopsis

Provide final information about an endpoint list.

Description

Syntax

```
Event: EndpointListComplete
EventList: <value>
ListItems: <value>
```

Arguments

- `EventList`
- `ListItems`

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ExtensionStateListComplete

ExtensionStateListComplete

Synopsis

Indicates the end of the list the current known extension states.

Description

Syntax

```
Event: ExtensionStateListComplete
EventList: <value>
ListItems: <value>
```

Arguments

- `EventList` - Conveys the status of the event list.
- `ListItems` - Conveys the number of statuses reported.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ExtensionStatus

ExtensionStatus

Synopsis

Raised when a hint changes due to a device state change.

Description

Syntax

```
Event: ExtensionStatus
Exten: <value>
Context: <value>
Hint: <value>
Status: <value>
StatusText: <value>
```

Arguments

- `Exten`
- `Context`
- `Hint`
- `Status`
- `StatusText`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FailedACL

FailedACL

Synopsis

Raised when a request violates an ACL check.

Description

Syntax

```
Event: FailedACL
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
[Module:] <value>
[ACLName:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Module` - If available, the name of the module that raised the event.
- `ACLName` - If available, the name of the ACL that failed.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FAXSession

FAXSession

Synopsis

Raised in response to FAXSession manager command

Description

Syntax

```
Event: FAXSession
[ActionID:] <value>
SessionNumber: <value>
Operation: <value>
State: <value>
[ErrorCorrectionMode:] <value>
[DataRate:] <value>
[ImageResolution:] <value>
[PageNumber:] <value>
[FileName:] <value>
[PagesTransmitted:] <value>
[PagesReceived:] <value>
[TotalBadLines:] <value>
```

Arguments

- `ActionID`
- `SessionNumber` - The numerical identifier for this particular session
- `Operation` - FAX session operation type
    - `gateway`
    - `V.21`
    - `send`
    - `receive`
    - `none`
- `State` - Current state of the FAX session
    - `Uninitialized`
    - `Initialized`
    - `Open`
    - `Active`
    - `Complete`
    - `Reserved`
    - `Inactive`
    - `Unknown`
- `ErrorCorrectionMode` - Whether error correcting mode is enabled for the FAX session. This field is not included when operation is 'V.21 Detect' or if operation is 'gateway' and state is 'Uninitialized'
    - `yes`
    - `no`
- `DataRate` - Bit rate of the FAX. This field is not included when operation is 'V.21 Detect' or if operation is 'gateway' and state is 'Uninitialized'.
- `ImageResolution` - Resolution of each page of the FAX. Will be in the format of X_RESxY_RES. This field is not included if the operation is anything other than Receive/Transmit.
- `PageNumber` - Current number of pages transferred during this FAX session. May change as the FAX progresses. This field is not included when operation is 'V.21 Detect' or if operation is 'gateway' and state is 'Uninitialized'.
- `FileName` - Filename of the image being sent/recieved for this FAX session. This field is not included if Operation isn't 'send' or 'receive'.
- `PagesTransmitted` - Total number of pages sent during this session. This field is not included if Operation isn't 'send' or 'receive'. Will always be 0 for 'receive'.
- `PagesReceived` - Total number of pages received during this session. This field is not included if Operation is not 'send' or 'receive'. Will be 0 for 'send'.
- `TotalBadLines` - Total number of bad lines sent/recieved during this session. This field is not included if Operation is not 'send' or 'received'.

Class

REPORTING

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FAXSessionsComplete

FAXSessionsComplete

Synopsis

Raised when all FAXSession events are completed for a FAXSessions command

Description

Syntax

```
Event: FAXSessionsComplete
[ActionID:] <value>
Total: <value>
```

Arguments

- `ActionID`
- `Total` - Count of FAXSession events sent in response to FAXSessions action

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FAXSessionsEntry

FAXSessionsEntry

Synopsis

A single list item for the FAXSessions AMI command

Description

Syntax

```
Event: FAXSessionsEntry
[ActionID:] <value>
Channel: <value>
Technology: <value>
SessionNumber: <value>
SessionType: <value>
Operation: <value>
State: <value>
Files: <value>
```

Arguments

- `ActionID`
- `Channel` - Name of the channel responsible for the FAX session
- `Technology` - The FAX technology that the FAX session is using
- `SessionNumber` - The numerical identifier for this particular session
- `SessionType` - FAX session passthru/relay type
    - `G.711`
    - `T.38`
- `Operation` - FAX session operation type
    - `gateway`
    - `V.21`
    - `send`
    - `receive`
    - `none`
- `State` - Current state of the FAX session
    - `Uninitialized`
    - `Initialized`
    - `Open`
    - `Active`
    - `Complete`
    - `Reserved`
    - `Inactive`
    - `Unknown`
- `Files` - File or list of files associated with this FAX session

Class

REPORTING

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FAXStats

FAXStats

Synopsis

Raised in response to FAXStats manager command

Description

Syntax

```
Event: FAXStats
[ActionID:] <value>
CurrentSessions: <value>
ReservedSessions: <value>
TransmitAttempts: <value>
ReceiveAttempts: <value>
CompletedFAXes: <value>
FailedFAXes: <value>
```

Arguments

- `ActionID`
- `CurrentSessions` - Number of active FAX sessions
- `ReservedSessions` - Number of reserved FAX sessions
- `TransmitAttempts` - Total FAX sessions for which Asterisk is/was the transmitter
- `ReceiveAttempts` - Total FAX sessions for which Asterisk is/was the recipient
- `CompletedFAXes` - Total FAX sessions which have been completed successfully
- `FailedFAXes` - Total FAX sessions which failed to complete successfully

Class

REPORTING

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FAXStatus

FAXStatus

Synopsis

Raised periodically during a fax transmission.

Description

Syntax

```
Event: FAXStatus
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Operation: <value>
Status: <value>
LocalStationID: <value>
FileName: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Operation`
    - `gateway`
    - `receive`
    - `send`
- `Status` - A text message describing the current status of the fax
- `LocalStationID` - The value of the `LOCALSTATIONID` channel variable
- `FileName` - The files being affected by the fax operation

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_FullyBooted

FullyBooted

Synopsis

Raised when all Asterisk initialization procedures have finished.

Description

Syntax

```
Event: FullyBooted
Status: <value>
```

Arguments

- `Status` - Informational message

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Hangup

Hangup

Synopsis

Raised when a channel is hung up.

Description

Syntax

```
Event: Hangup
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Cause: <value>
Cause-txt: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Cause` - A numeric cause code for why the channel was hung up.
- `Cause-txt` - A description of why the channel was hung up.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_HangupHandlerPop

HangupHandlerPop

Synopsis

Raised when a hangup handler is removed from the handler stack by the CHANNEL() function.

Description

Syntax

```
Event: HangupHandlerPop
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Handler: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Handler` - Hangup handler parameter string passed to the Gosub application.

Class

DIALPLAN

See Also

- Asterisk 13 ManagerEvent_HangupHandlerPush
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_HangupHandlerPush

HangupHandlerPush

Synopsis

Raised when a hangup handler is added to the handler stack by the CHANNEL() function.

Description

Syntax

```
Event: HangupHandlerPush
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Handler: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Handler` - Hangup handler parameter string passed to the Gosub application.

Class

DIALPLAN

See Also

- Asterisk 13 ManagerEvent_HangupHandlerPop
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_HangupHandlerRun

HangupHandlerRun

Synopsis

Raised when a hangup handler is about to be called.

Description

Syntax

```
Event: HangupHandlerRun
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Handler: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Handler` - Hangup handler parameter string passed to the Gosub application.

Class

DIALPLAN

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_HangupRequest

HangupRequest

Synopsis

Raised when a hangup is requested.

Description

Syntax

```
Event: HangupRequest
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Cause: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Cause` - A numeric cause code for why the channel was hung up.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Hold

Hold

Synopsis

Raised when a channel goes on hold.

Description

Syntax

```
Event: Hold
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
MusicClass: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `MusicClass` - The suggested MusicClass, if provided.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_IdentifyDetail

IdentifyDetail

Synopsis

Provide details about an identify section.

Description

Syntax

```
Event: IdentifyDetail
ObjectType: <value>
ObjectName: <value>
Endpoint: <value>
Match: <value>
EndpointName: <value>
```

Arguments

- `ObjectType` - The object's type. This will always be 'identify'.
- `ObjectName` - The name of this object.
- `Endpoint` - Name of Endpoint
- `Match` - IP addresses or networks to match against
- `EndpointName` - The name of the endpoint associated with this information.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_InvalidAccountID

InvalidAccountID

Synopsis

Raised when a request fails an authentication check due to an invalid account ID.

Description

Syntax

```
Event: InvalidAccountID
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_InvalidPassword

InvalidPassword

Synopsis

Raised when a request provides an invalid password during an authentication attempt.

Description

Syntax

```
Event: InvalidPassword
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
[Module:] <value>
[SessionTV:] <value>
[Challenge:] <value>
[ReceivedChallenge:] <value>
[RecievedHash:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.
- `Challenge` - The challenge that was sent.
- `ReceivedChallenge` - The challenge that was received.
- `RecievedHash` - The hash that was received.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_InvalidTransport

InvalidTransport

Synopsis

Raised when a request attempts to use a transport not allowed by the Asterisk service.

Description

Syntax

```
Event: InvalidTransport
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
AttemptedTransport: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `AttemptedTransport` - The transport type that the request attempted to use.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_LoadAverageLimit

LoadAverageLimit

Synopsis

Raised when a request fails because a configured load average limit has been reached.

Description

Syntax

```
Event: LoadAverageLimit
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_LocalBridge

LocalBridge

Synopsis

Raised when two halves of a Local Channel form a bridge.

Description

Syntax

```
Event: LocalBridge
LocalOneChannel: <value>
LocalOneChannelState: <value>
LocalOneChannelStateDesc: <value>
LocalOneCallerIDNum: <value>
LocalOneCallerIDName: <value>
LocalOneConnectedLineNum: <value>
LocalOneConnectedLineName: <value>
LocalOneAccountCode: <value>
LocalOneContext: <value>
LocalOneExten: <value>
LocalOnePriority: <value>
LocalOneUniqueid: <value>
LocalTwoChannel: <value>
LocalTwoChannelState: <value>
LocalTwoChannelStateDesc: <value>
LocalTwoCallerIDNum: <value>
LocalTwoCallerIDName: <value>
LocalTwoConnectedLineNum: <value>
LocalTwoConnectedLineName: <value>
LocalTwoAccountCode: <value>
LocalTwoContext: <value>
LocalTwoExten: <value>
LocalTwoPriority: <value>
LocalTwoUniqueid: <value>
Context: <value>
Exten: <value>
LocalOptimization: <value>
```

Arguments

- `LocalOneChannel`
- `LocalOneChannelState` - A numeric code for the channel's current state, related to LocalOneChannelStateDesc
- `LocalOneChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalOneCallerIDNum`
- `LocalOneCallerIDName`
- `LocalOneConnectedLineNum`
- `LocalOneConnectedLineName`
- `LocalOneAccountCode`
- `LocalOneContext`
- `LocalOneExten`
- `LocalOnePriority`
- `LocalOneUniqueid`
- `LocalTwoChannel`
- `LocalTwoChannelState` - A numeric code for the channel's current state, related to LocalTwoChannelStateDesc
- `LocalTwoChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`

    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalTwoCallerIDNum`
- `LocalTwoCallerIDName`
- `LocalTwoConnectedLineNum`
- `LocalTwoConnectedLineName`
- `LocalTwoAccountCode`
- `LocalTwoContext`
- `LocalTwoExten`
- `LocalTwoPriority`
- `LocalTwoUniqueid`
- `Context` - The context in the dialplan that Channel2 starts in.
- `Exten` - The extension in the dialplan that Channel2 starts in.
- `LocalOptimization`
    - `Yes`
    - `No`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_LocalOptimizationBegin

LocalOptimizationBegin

Synopsis

Raised when two halves of a Local Channel begin to optimize themselves out of the media path.

Description

Syntax

```
Event: LocalOptimizationBegin
LocalOneChannel: <value>
LocalOneChannelState: <value>
LocalOneChannelStateDesc: <value>
LocalOneCallerIDNum: <value>
LocalOneCallerIDName: <value>
LocalOneConnectedLineNum: <value>
LocalOneConnectedLineName: <value>
LocalOneAccountCode: <value>
LocalOneContext: <value>
LocalOneExten: <value>
LocalOnePriority: <value>
LocalOneUniqueid: <value>
LocalTwoChannel: <value>
LocalTwoChannelState: <value>
LocalTwoChannelStateDesc: <value>
LocalTwoCallerIDNum: <value>
LocalTwoCallerIDName: <value>
LocalTwoConnectedLineNum: <value>
LocalTwoConnectedLineName: <value>
LocalTwoAccountCode: <value>
LocalTwoContext: <value>
LocalTwoExten: <value>
LocalTwoPriority: <value>
LocalTwoUniqueid: <value>
SourceChannel: <value>
SourceChannelState: <value>
SourceChannelStateDesc: <value>
SourceCallerIDNum: <value>
SourceCallerIDName: <value>
SourceConnectedLineNum: <value>
SourceConnectedLineName: <value>
SourceAccountCode: <value>
SourceContext: <value>
SourceExten: <value>
SourcePriority: <value>
SourceUniqueid: <value>
DestUniqueId: <value>
Id: <value>
```

Arguments

- `LocalOneChannel`
- `LocalOneChannelState` - A numeric code for the channel's current state, related to LocalOneChannelStateDesc
- `LocalOneChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalOneCallerIDNum`
- `LocalOneCallerIDName`
- `LocalOneConnectedLineNum`
- `LocalOneConnectedLineName`
- `LocalOneAccountCode`
- `LocalOneContext`
- `LocalOneExten`
- `LocalOnePriority`

- `LocalOneUniqueid`
- `LocalTwoChannel`
- `LocalTwoChannelState` - A numeric code for the channel's current state, related to LocalTwoChannelStateDesc
- `LocalTwoChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalTwoCallerIDNum`
- `LocalTwoCallerIDName`
- `LocalTwoConnectedLineNum`
- `LocalTwoConnectedLineName`
- `LocalTwoAccountCode`
- `LocalTwoContext`
- `LocalTwoExten`
- `LocalTwoPriority`
- `LocalTwoUniqueid`
- `SourceChannel`
- `SourceChannelState` - A numeric code for the channel's current state, related to SourceChannelStateDesc
- `SourceChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `SourceCallerIDNum`
- `SourceCallerIDName`
- `SourceConnectedLineNum`
- `SourceConnectedLineName`
- `SourceAccountCode`
- `SourceContext`
- `SourceExten`
- `SourcePriority`
- `SourceUniqueid`
- `DestUniqueId` - The unique ID of the bridge into which the local channel is optimizing.
- `Id` - Identification for the optimization operation.

Class

CALL

See Also

- Asterisk 13 ManagerEvent_LocalOptimizationEnd
- Asterisk 13 ManagerAction_LocalOptimizeAway

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_LocalOptimizationEnd

LocalOptimizationEnd

Synopsis

Raised when two halves of a Local Channel have finished optimizing themselves out of the media path.

Description

Syntax

```
Event: LocalOptimizationEnd
LocalOneChannel: <value>
LocalOneChannelState: <value>
LocalOneChannelStateDesc: <value>
LocalOneCallerIDNum: <value>
LocalOneCallerIDName: <value>
LocalOneConnectedLineNum: <value>
LocalOneConnectedLineName: <value>
LocalOneAccountCode: <value>
LocalOneContext: <value>
LocalOneExten: <value>
LocalOnePriority: <value>
LocalOneUniqueid: <value>
LocalTwoChannel: <value>
LocalTwoChannelState: <value>
LocalTwoChannelStateDesc: <value>
LocalTwoCallerIDNum: <value>
LocalTwoCallerIDName: <value>
LocalTwoConnectedLineNum: <value>
LocalTwoConnectedLineName: <value>
LocalTwoAccountCode: <value>
LocalTwoContext: <value>
LocalTwoExten: <value>
LocalTwoPriority: <value>
LocalTwoUniqueid: <value>
Success: <value>
Id: <value>
```

Arguments

- `LocalOneChannel`
- `LocalOneChannelState` - A numeric code for the channel's current state, related to LocalOneChannelStateDesc
- `LocalOneChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `LocalOneCallerIDNum`
- `LocalOneCallerIDName`
- `LocalOneConnectedLineNum`
- `LocalOneConnectedLineName`
- `LocalOneAccountCode`
- `LocalOneContext`
- `LocalOneExten`
- `LocalOnePriority`
- `LocalOneUniqueid`
- `LocalTwoChannel`
- `LocalTwoChannelState` - A numeric code for the channel's current state, related to LocalTwoChannelStateDesc
- `LocalTwoChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`

        - `Ringing`
        - `Up`
        - `Busy`
        - `Dialing Offhook`
        - `Pre-ring`
        - `Unknown`
- `LocalTwoCallerIDNum`
- `LocalTwoCallerIDName`
- `LocalTwoConnectedLineNum`
- `LocalTwoConnectedLineName`
- `LocalTwoAccountCode`
- `LocalTwoContext`
- `LocalTwoExten`
- `LocalTwoPriority`
- `LocalTwoUniqueid`
- `Success` - Indicates whether the local optimization succeeded.
- `Id` - Identification for the optimization operation. Matches the *Id* from a previous `LocalOptimizationBegin`

Class

CALL

See Also

- Asterisk 13 ManagerEvent_LocalOptimizationBegin
- Asterisk 13 ManagerAction_LocalOptimizeAway

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_LogChannel

LogChannel

Synopsis

Raised when a logging channel is re-enabled after a reload operation.

Description

Syntax

```
Event: LogChannel
Channel: <value>
Enabled: <value>
```

Arguments

- `Channel` - The name of the logging channel.
- `Enabled`

Class

SYSTEM

See Also

Synopsis

Raised when a logging channel is disabled.

Description

Syntax

```
Event: LogChannel
Channel: <value>
Enabled: <value>
Reason: <value>
```

Arguments

- `Channel` - The name of the logging channel.
- `Enabled`
- `Reason`

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MCID

MCID

Synopsis

Published when a malicious call ID request arrives.

Description

Syntax

```
Event: MCID
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
MCallerIDNumValid: <value>
MCallerIDNum: <value>
MCallerIDton: <value>
MCallerIDNumPlan: <value>
MCallerIDNumPres: <value>
MCallerIDNameValid: <value>
MCallerIDName: <value>
MCallerIDNameCharSet: <value>
MCallerIDNamePres: <value>
MCallerIDSubaddr: <value>
MCallerIDSubaddrType: <value>
MCallerIDSubaddrOdd: <value>
MCallerIDPres: <value>
MConnectedIDNumValid: <value>
MConnectedIDNum: <value>
MConnectedIDton: <value>
MConnectedIDNumPlan: <value>
MConnectedIDNumPres: <value>
MConnectedIDNameValid: <value>
MConnectedIDName: <value>
MConnectedIDNameCharSet: <value>
MConnectedIDNamePres: <value>
MConnectedIDSubaddr: <value>
MConnectedIDSubaddrType: <value>
MConnectedIDSubaddrOdd: <value>
MConnectedIDPres: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`

* `Uniqueid`
* `MCallerIDNumValid`
* `MCallerIDNum`
* `MCallerIDton`
* `MCallerIDNumPlan`
* `MCallerIDNumPres`
* `MCallerIDNameValid`
* `MCallerIDName`
* `MCallerIDNameCharSet`
* `MCallerIDNamePres`
* `MCallerIDSubaddr`
* `MCallerIDSubaddrType`
* `MCallerIDSubaddrOdd`
* `MCallerIDPres`
* `MConnectedIDNumValid`
* `MConnectedIDNum`
* `MConnectedIDton`
* `MConnectedIDNumPlan`
* `MConnectedIDNumPres`
* `MConnectedIDNameValid`
* `MConnectedIDName`
* `MConnectedIDNameCharSet`
* `MConnectedIDNamePres`
* `MConnectedIDSubaddr`
* `MConnectedIDSubaddrType`
* `MConnectedIDSubaddrOdd`
* `MConnectedIDPres`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MeetmeEnd

MeetmeEnd

Synopsis

Raised when a MeetMe conference ends.

Description

Syntax

```
Event: MeetmeEnd
Meetme: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MeetmeJoin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MeetmeJoin

MeetmeJoin

Synopsis

Raised when a user joins a MeetMe conference.

Description

Syntax

```
Event: MeetmeJoin
Meetme: <value>
Usernum: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MeetmeLeave
- Asterisk 13 Application_MeetMe

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MeetmeLeave

MeetmeLeave

Synopsis

Raised when a user leaves a MeetMe conference.

Description

Syntax

```
Event: MeetmeLeave
Meetme: <value>
Usernum: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Duration: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Duration` - The length of time in seconds that the Meetme user was in the conference.

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MeetmeJoin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MeetmeMute

MeetmeMute

Synopsis

Raised when a MeetMe user is muted or unmuted.

Description

Syntax

```
Event: MeetmeMute
Meetme: <value>
Usernum: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Duration: <value>
Status: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Duration` - The length of time in seconds that the Meetme user has been in the conference at the time of this event.
- `Status`
    - `on`
    - `off`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MeetmeTalking

MeetmeTalking

Synopsis

Raised when a MeetMe user begins or ends talking.

Description

Syntax

```
Event: MeetmeTalking
Meetme: <value>
Usernum: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Duration: <value>
Status: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Duration` - The length of time in seconds that the Meetme user has been in the conference at the time of this event.
- `Status`
    - `on`
    - `off`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MeetmeTalkRequest

MeetmeTalkRequest

Synopsis

Raised when a MeetMe user has started talking.

Description

Syntax

```
Event: MeetmeTalkRequest
Meetme: <value>
Usernum: <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Duration: <value>
Status: <value>
```

Arguments

- `Meetme` - The identifier for the MeetMe conference.
- `Usernum` - The identifier of the MeetMe user who joined.
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Duration` - The length of time in seconds that the Meetme user has been in the conference at the time of this event.
- `Status`
    - `on`
    - `off`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MemoryLimit

MemoryLimit

Synopsis

Raised when a request fails due to an internal memory allocation failure.

Description

Syntax

```
Event: MemoryLimit
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MessageWaiting

MessageWaiting

Synopsis

Raised when the state of messages in a voicemail mailbox has changed or when a channel has finished interacting with a mailbox.

Description

> **Note**
> The Channel related parameters are only present if a channel was involved in the manipulation of a mailbox. If no channel is involved, the parameters are not included with the event.

Syntax

```
Event: MessageWaiting
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Mailbox: <value>
Waiting: <value>
New: <value>
Old: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Mailbox` - The mailbox with the new message, specified as `mailbox@context`
- `Waiting` - Whether or not the mailbox has messages waiting for it.
- `New` - The number of new messages.
- `Old` - The number of old messages.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MiniVoiceMail

MiniVoiceMail

Synopsis

Raised when a notification is sent out by a MiniVoiceMail application

Description

Syntax

```
Event: MiniVoiceMail
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Action: <value>
Mailbox: <value>
Counter: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Action` - What action was taken. Currently, this will always be `SentNotification`
- `Mailbox` - The mailbox that the notification was about, specified as `mailbox@context`
- `Counter` - A message counter derived from the `MVM_COUNTER` channel variable.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MonitorStart

MonitorStart

Synopsis

Raised when monitoring has started on a channel.

Description

Syntax

```
Event: MonitorStart
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MonitorStop
- Asterisk 13 Application_Monitor
- Asterisk 13 ManagerAction_Monitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MonitorStop

MonitorStop

Synopsis

Raised when monitoring has stopped on a channel.

Description

Syntax

```
Event: MonitorStop
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MonitorStart
- Asterisk 13 Application_StopMonitor
- Asterisk 13 ManagerAction_StopMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MusicOnHoldStart

MusicOnHoldStart

Synopsis

Raised when music on hold has started on a channel.

Description

Syntax

```
Event: MusicOnHoldStart
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Class: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Class` - The class of music being played on the channel

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MusicOnHoldStop
- Asterisk 13 Application_MusicOnHold

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MusicOnHoldStop

MusicOnHoldStop

Synopsis

Raised when music on hold has stopped on a channel.

Description

Syntax

```
Event: MusicOnHoldStop
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

- Asterisk 13 ManagerEvent_MusicOnHoldStart
- Asterisk 13 Application_StopMusicOnHold

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MWIGet

MWIGet

Synopsis

Raised in response to a MWIGet command.

Description

Syntax

```
Event: MWIGet
[ActionID:] <value>
Mailbox: <value>
OldMessages: <value>
NewMessages: <value>
```

Arguments

- `ActionID`
- `Mailbox` - Specific mailbox ID.
- `OldMessages` - The number of old messages in the mailbox.
- `NewMessages` - The number of new messages in the mailbox.

Class

REPORTING

See Also

- Asterisk 13 ManagerAction_MWIGet

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_MWIGetComplete

MWIGetComplete

Synopsis

Raised in response to a MWIGet command.

Description

Syntax

```
Event: MWIGetComplete
[ActionID:] <value>
EventList: <value>
ListItems: <value>
```

Arguments

- `ActionID`
- `EventList`
- `ListItems` - The number of mailboxes reported.

Class

REPORTING

See Also

- Asterisk 13 ManagerAction_MWIGet

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_NewAccountCode

NewAccountCode

Synopsis

Raised when a Channel's AccountCode is changed.

Description

Syntax

```
Event: NewAccountCode
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
OldAccountCode: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `OldAccountCode` - The channel's previous account code

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_NewCallerid

NewCallerid

Synopsis

Raised when a channel receives new Caller ID information.

Description

Syntax

```
Event: NewCallerid
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
CID-CallingPres: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `CID-CallingPres` - A description of the Caller ID presentation.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Newchannel

Newchannel

Synopsis

Raised when a new channel is created.

Description

Syntax

```
Event: Newchannel
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_NewExten

NewExten

Synopsis

Raised when a channel enters a new context, extension, priority.

Description

Syntax

```
Event: NewExten
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Extension: <value>
Application: <value>
AppData: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Extension` - Deprecated in 12, but kept for backward compatability. Please use 'Exten' instead.
- `Application` - The application about to be executed.
- `AppData` - The data to be passed to the application.

Class

DIALPLAN

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Newstate

Newstate

Synopsis

Raised when a channel's state changes.

Description

Syntax

```
Event: Newstate
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_OriginateResponse

OriginateResponse

Synopsis

Raised in response to an Originate command.

Description

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`

Class

CALL

See Also

- Asterisk 13 ManagerAction_Originate

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ParkedCall

ParkedCall

Synopsis

Raised when a channel is parked.

Description

Syntax

```
Event: ParkedCall
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
```

Arguments

- `ParkeeChannel`
- `ParkeeChannelState` - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- `ParkeeChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `ParkeeCallerIDNum`
- `ParkeeCallerIDName`
- `ParkeeConnectedLineNum`
- `ParkeeConnectedLineName`
- `ParkeeAccountCode`
- `ParkeeContext`
- `ParkeeExten`
- `ParkeePriority`
- `ParkeeUniqueid`
- `ParkerDialString` - Dial String that can be used to call back the parker on ParkingTimeout.
- `Parkinglot` - Name of the parking lot that the parkee is parked in
- `ParkingSpace` - Parking Space that the parkee is parked in
- `ParkingTimeout` - Time remaining until the parkee is forcefully removed from parking in seconds
- `ParkingDuration` - Time the parkee has been in the parking bridge (in seconds)

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ParkedCallGiveUp

ParkedCallGiveUp

Synopsis

Raised when a channel leaves a parking lot because it hung up without being answered.

Description

Syntax

```
Event: ParkedCallGiveUp
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerChannel: <value>
ParkerChannelState: <value>
ParkerChannelStateDesc: <value>
ParkerCallerIDNum: <value>
ParkerCallerIDName: <value>
ParkerConnectedLineNum: <value>
ParkerConnectedLineName: <value>
ParkerAccountCode: <value>
ParkerContext: <value>
ParkerExten: <value>
ParkerPriority: <value>
ParkerUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
```

Arguments

- `ParkeeChannel`
- `ParkeeChannelState` - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- `ParkeeChannelStateDesc`
  - `Down`
  - `Rsrvd`
  - `OffHook`
  - `Dialing`
  - `Ring`
  - `Ringing`
  - `Up`
  - `Busy`
  - `Dialing Offhook`
  - `Pre-ring`
  - `Unknown`
- `ParkeeCallerIDNum`
- `ParkeeCallerIDName`
- `ParkeeConnectedLineNum`
- `ParkeeConnectedLineName`
- `ParkeeAccountCode`
- `ParkeeContext`
- `ParkeeExten`
- `ParkeePriority`
- `ParkeeUniqueid`
- `ParkerChannel`
- `ParkerChannelState` - A numeric code for the channel's current state, related to ParkerChannelStateDesc
- `ParkerChannelStateDesc`
  - `Down`
  - `Rsrvd`
  - `OffHook`

    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `ParkerCallerIDNum`
- `ParkerCallerIDName`
- `ParkerConnectedLineNum`
- `ParkerConnectedLineName`
- `ParkerAccountCode`
- `ParkerContext`
- `ParkerExten`
- `ParkerPriority`
- `ParkerUniqueid`
- `ParkerDialString` - Dial String that can be used to call back the parker on ParkingTimeout.
- `Parkinglot` - Name of the parking lot that the parkee is parked in
- `ParkingSpace` - Parking Space that the parkee is parked in
- `ParkingTimeout` - Time remaining until the parkee is forcefully removed from parking in seconds
- `ParkingDuration` - Time the parkee has been in the parking bridge (in seconds)

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ParkedCallTimeOut

ParkedCallTimeOut

Synopsis

Raised when a channel leaves a parking lot due to reaching the time limit of being parked.

Description

Syntax

```
Event: ParkedCallTimeOut
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerChannel: <value>
ParkerChannelState: <value>
ParkerChannelStateDesc: <value>
ParkerCallerIDNum: <value>
ParkerCallerIDName: <value>
ParkerConnectedLineNum: <value>
ParkerConnectedLineName: <value>
ParkerAccountCode: <value>
ParkerContext: <value>
ParkerExten: <value>
ParkerPriority: <value>
ParkerUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
```

Arguments

- `ParkeeChannel`
- `ParkeeChannelState` - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- `ParkeeChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`
    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `ParkeeCallerIDNum`
- `ParkeeCallerIDName`
- `ParkeeConnectedLineNum`
- `ParkeeConnectedLineName`
- `ParkeeAccountCode`
- `ParkeeContext`
- `ParkeeExten`
- `ParkeePriority`
- `ParkeeUniqueid`
- `ParkerChannel`
- `ParkerChannelState` - A numeric code for the channel's current state, related to ParkerChannelStateDesc
- `ParkerChannelStateDesc`
    - `Down`
    - `Rsrvd`
    - `OffHook`

    - `Dialing`
    - `Ring`
    - `Ringing`
    - `Up`
    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `ParkerCallerIDNum`
- `ParkerCallerIDName`
- `ParkerConnectedLineNum`
- `ParkerConnectedLineName`
- `ParkerAccountCode`
- `ParkerContext`
- `ParkerExten`
- `ParkerPriority`
- `ParkerUniqueid`
- `ParkerDialString` - Dial String that can be used to call back the parker on ParkingTimeout.
- `Parkinglot` - Name of the parking lot that the parkee is parked in
- `ParkingSpace` - Parking Space that the parkee is parked in
- `ParkingTimeout` - Time remaining until the parkee is forcefully removed from parking in seconds
- `ParkingDuration` - Time the parkee has been in the parking bridge (in seconds)

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_PeerStatus

PeerStatus

Synopsis

Raised when the state of a peer changes.

Description

Syntax

```
Event: PeerStatus
ChannelType: <value>
Peer: <value>
PeerStatus: <value>
Cause: <value>
Address: <value>
Port: <value>
Time: <value>
```

Arguments

- `ChannelType` - The channel technology of the peer.
- `Peer` - The name of the peer (including channel technology).
- `PeerStatus` - New status of the peer.
    - `Unknown`
    - `Registered`
    - `Unregistered`
    - `Rejected`
    - `Lagged`
- `Cause` - The reason the status has changed.
- `Address` - New address of the peer.
- `Port` - New port for the peer.
- `Time` - Time it takes to reach the peer and receive a response.

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Pickup

Pickup

Synopsis

Raised when a call pickup occurs.

Description

Syntax

```
Event: Pickup
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
TargetChannel: <value>
TargetChannelState: <value>
TargetChannelStateDesc: <value>
TargetCallerIDNum: <value>
TargetCallerIDName: <value>
TargetConnectedLineNum: <value>
TargetConnectedLineName: <value>
TargetAccountCode: <value>
TargetContext: <value>
TargetExten: <value>
TargetPriority: <value>
TargetUniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `TargetChannel`
- `TargetChannelState` - A numeric code for the channel's current state, related to TargetChannelStateDesc
- `TargetChannelStateDesc`
  - `Down`
  - `Rsrvd`
  - `OffHook`
  - `Dialing`
  - `Ring`
  - `Ringing`
  - `Up`

    - `Busy`
    - `Dialing Offhook`
    - `Pre-ring`
    - `Unknown`
- `TargetCallerIDNum`
- `TargetCallerIDName`
- `TargetConnectedLineNum`
- `TargetConnectedLineName`
- `TargetAccountCode`
- `TargetContext`
- `TargetExten`
- `TargetPriority`
- `TargetUniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_PresenceStateChange

PresenceStateChange

Synopsis

Raised when a presence state changes

Description

This differs from the `PresenceStatus` event because this event is raised for all presence state changes, not only for changes that affect dialplan hints.

Syntax

```
Event: PresenceStateChange
Presentity: <value>
Status: <value>
Subtype: <value>
Message: <value>
```

Arguments

- `Presentity` - The entity whose presence state has changed
- `Status` - The new status of the presentity
- `Subtype` - The new subtype of the presentity
- `Message` - The new message of the presentity

Class

CALL

See Also

- Asterisk 13 ManagerEvent_PresenceStatus

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 ManagerEvent_PresenceStateListComplete

PresenceStateListComplete

Synopsis

Indicates the end of the list the current known extension states.

Description

Syntax

```
Event: PresenceStateListComplete
EventList: <value>
ListItems: <value>
```

Arguments

- `EventList` - Conveys the status of the event list.
- `ListItems` - Conveys the number of statuses reported.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_PresenceStatus

PresenceStatus

Synopsis

Raised when a hint changes due to a presence state change.

Description

Syntax

```
Event: PresenceStatus
Exten: <value>
Context: <value>
Hint: <value>
Status: <value>
Subtype: <value>
Message: <value>
```

Arguments

- `Exten`
- `Context`
- `Hint`
- `Status`
- `Subtype`
- `Message`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueCallerAbandon

QueueCallerAbandon

Synopsis

Raised when a caller abandons the queue.

Description

Syntax

```
Event: QueueCallerAbandon
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Queue: <value>
Position: <value>
OriginalPosition: <value>
HoldTime: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `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.

Class

AGENT

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueCallerJoin

QueueCallerJoin

Synopsis

Raised when a caller joins a Queue.

Description

Syntax

```
Event: QueueCallerJoin
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Queue: <value>
Position: <value>
Count: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Queue` - The name of the queue.
- `Position` - This channel's current position in the queue.
- `Count` - The total number of channels in the queue.

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_QueueCallerLeave
- Asterisk 13 Application_Queue

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueCallerLeave

QueueCallerLeave

Synopsis

Raised when a caller leaves a Queue.

Description

Syntax

```
Event: QueueCallerLeave
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Queue: <value>
Count: <value>
Position: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Queue` - The name of the queue.
- `Count` - The total number of channels in the queue.
- `Position` - This channel's current position in the queue.

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_QueueCallerJoin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueMemberAdded

QueueMemberAdded

Synopsis

Raised when a member is added to the queue.

Description

Syntax

```
Event: QueueMemberAdded
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.
- `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 a 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`
- `Ringinuse`
    - `0`
    - `1`

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_QueueMemberRemoved
- Asterisk 13 Application_AddQueueMember

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueMemberPause

QueueMemberPause

Synopsis

Raised when a member is paused/unpaused in the queue.

Description

Syntax

```
Event: QueueMemberPause
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
Reason: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.
- `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 a 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`
- `Ringinuse`
    - `0`
    - `1`
- `Reason` - The reason a member was paused.

Class

AGENT

See Also

- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnPauseQueueMember

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueMemberPenalty

QueueMemberPenalty

Synopsis

Raised when a member's penalty is changed.

Description

Syntax

```
Event: QueueMemberPenalty
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.
- `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 a 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`
- `Ringinuse`
    - `0`
    - `1`

Class

AGENT

See Also

- Asterisk 13 Function_QUEUE_MEMBER

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueMemberRemoved

QueueMemberRemoved

Synopsis

Raised when a member is removed from the queue.

Description

Syntax

```
Event: QueueMemberRemoved
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.
- `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 a 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`
- `Ringinuse`
    - `0`
    - `1`

Class

AGENT

See Also

- Asterisk 13 ManagerEvent_QueueMemberAdded
- Asterisk 13 Application_RemoveQueueMember

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueMemberRinginuse

QueueMemberRinginuse

Synopsis

Raised when a member's ringinuse setting is changed.

Description

Syntax

```
Event: QueueMemberRinginuse
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.
- `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 a 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`
- `Ringinuse`
  - `0`
  - `1`

Class

AGENT

See Also

- Asterisk 13 Function_QUEUE_MEMBER

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_QueueMemberStatus

QueueMemberStatus

Synopsis

Raised when a Queue member's status has changed.

Description

Syntax

```
Event: QueueMemberStatus
Queue: <value>
MemberName: <value>
Interface: <value>
StateInterface: <value>
Membership: <value>
Penalty: <value>
CallsTaken: <value>
LastCall: <value>
Status: <value>
Paused: <value>
Ringinuse: <value>
```

Arguments

- `Queue` - The name of the queue.
- `MemberName` - The name of the queue member.
- `Interface` - The queue member's channel technology or location.
- `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 a 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`
- `Ringinuse`
    - `0`
    - `1`

Class

AGENT

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_ReceiveFAX

ReceiveFAX

Synopsis

Raised when a receive fax operation has completed.

Description

Syntax

```
Event: ReceiveFAX
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
LocalStationID: <value>
RemoteStationID: <value>
PagesTransferred: <value>
Resolution: <value>
TransferRate: <value>
FileName: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `LocalStationID` - The value of the `LOCALSTATIONID` channel variable
- `RemoteStationID` - The value of the `REMOTESTATIONID` channel variable
- `PagesTransferred` - The number of pages that have been transferred
- `Resolution` - The negotiated resolution
- `TransferRate` - The negotiated transfer rate
- `FileName` - The files being affected by the fax operation

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Registry

Registry

Synopsis

Raised when an outbound registration completes.

Description

Syntax

```
Event: Registry
ChannelType: <value>
Username: <value>
Domain: <value>
Status: <value>
Cause: <value>
```

Arguments

- `ChannelType` - The type of channel that was registered (or not).
- `Username` - The username portion of the registration.
- `Domain` - The address portion of the registration.
- `Status` - The status of the registration request.
    - `Registered`
    - `Unregistered`
    - `Rejected`
    - `Failed`
- `Cause` - What caused the rejection of the request, if available.

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Reload

Reload

Synopsis

Raised when a module has been reloaded in Asterisk.

Description

Syntax

```
Event: Reload
Module: <value>
Status: <value>
```

Arguments

- `Module` - The name of the module that was reloaded, or `All` if all modules were reloaded
- `Status` - The numeric status code denoting the success or failure of the reload request.
    - `0` - Success
    - `1` - Request queued
    - `2` - Module not found
    - `3` - Error
    - `4` - Reload already in progress
    - `5` - Module uninitialized
    - `6` - Reload not supported

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Rename

Rename

Synopsis

Raised when the name of a channel is changed.

Description

Syntax

```
Event: Rename
Channel: <value>
Newname: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `Newname`
- `Uniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_RequestBadFormat

RequestBadFormat

Synopsis

Raised when a request is received with bad formatting.

Description

Syntax

```
Event: RequestBadFormat
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
RequestType: <value>
[Module:] <value>
[SessionTV:] <value>
[AccountID:] <value>
[RequestParams:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
  - `Informational`
  - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `RequestType` - The type of request attempted.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.
- `AccountID` - The account ID associated with the rejected request.
- `RequestParams` - Parameters provided to the rejected request.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_RequestNotAllowed

RequestNotAllowed

Synopsis

Raised when a request is not allowed by the service.

Description

Syntax

```
Event: RequestNotAllowed
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
RequestType: <value>
[Module:] <value>
[SessionTV:] <value>
[RequestParams:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `RequestType` - The type of request attempted.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.
- `RequestParams` - Parameters provided to the rejected request.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_RequestNotSupported

RequestNotSupported

Synopsis

Raised when a request fails due to some aspect of the requested item not being supported by the service.

Description

Syntax

```
Event: RequestNotSupported
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
RequestType: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `RequestType` - The type of request attempted.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_RTCPReceived

RTCPReceived

Synopsis

Raised when an RTCP packet is received.

Description

Syntax

```
Event: RTCPReceived
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
SSRC: <value>
PT: <value>
From: <value>
RTT: <value>
ReportCount: <value>
[SentNTP:] <value>
[SentRTP:] <value>
[SentPackets:] <value>
[SentOctets:] <value>
ReportXSourceSSRC: <value>
ReportXFractionLost: <value>
ReportXCumulativeLost: <value>
ReportXHighestSequence: <value>
ReportXSequenceNumberCycles: <value>
ReportXIAJitter: <value>
ReportXLSR: <value>
ReportXDLSR: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `SSRC` - The SSRC identifier for the remote system
- `PT` - The type of packet for this RTCP report.
    - `200(SR)`
    - `201(RR)`
- `From` - The address the report was received from.
- `RTT` - Calculated Round-Trip Time in seconds

- `ReportCount` - The number of reports that were received.
  The report count determines the number of ReportX headers in the message. The X for each set of report headers will range from 0 to `ReportCount - 1`.
- `SentNTP` - The time the sender generated the report. Only valid when PT is `200(SR)`.
- `SentRTP` - The sender's last RTP timestamp. Only valid when PT is `200(SR)`.
- `SentPackets` - The number of packets the sender has sent. Only valid when PT is `200(SR)`.
- `SentOctets` - The number of bytes the sender has sent. Only valid when PT is `200(SR)`.
- `ReportXSourceSSRC` - The SSRC for the source of this report block.
- `ReportXFractionLost` - The fraction of RTP data packets from `ReportXSourceSSRC` lost since the previous SR or RR report was sent.
- `ReportXCumulativeLost` - The total number of RTP data packets from `ReportXSourceSSRC` lost since the beginning of reception.
- `ReportXHighestSequence` - The highest sequence number received in an RTP data packet from `ReportXSourceSSRC`.
- `ReportXSequenceNumberCycles` - The number of sequence number cycles seen for the RTP data received from `ReportXSourceSSRC`.
- `ReportXIAJitter` - An estimate of the statistical variance of the RTP data packet interarrival time, measured in timestamp units.
- `ReportXLSR` - The last SR timestamp received from `ReportXSourceSSRC`. If no SR has been received from `ReportXSourceSSRC`, then 0.
- `ReportXDLSR` - The delay, expressed in units of 1/65536 seconds, between receiving the last SR packet from `ReportXSourceSSRC` and sending this report.

Class

REPORTING

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_RTCPSent

RTCPSent

Synopsis

Raised when an RTCP packet is sent.

Description

Syntax

```
Event: RTCPSent
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
SSRC: <value>
PT: <value>
To: <value>
ReportCount: <value>
[SentNTP:] <value>
[SentRTP:] <value>
[SentPackets:] <value>
[SentOctets:] <value>
ReportXSourceSSRC: <value>
ReportXFractionLost: <value>
ReportXCumulativeLost: <value>
ReportXHighestSequence: <value>
ReportXSequenceNumberCycles: <value>
ReportXIAJitter: <value>
ReportXLSR: <value>
ReportXDLSR: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `SSRC` - The SSRC identifier for our stream
- `PT` - The type of packet for this RTCP report.
    - `200(SR)`
    - `201(RR)`
- `To` - The address the report is sent to.
- `ReportCount` - The number of reports that were sent.
  The report count determines the number of ReportX headers in the message. The X for each set of report headers will range from 0 to `Re`

portCount - 1.

- SentNTP - The time the sender generated the report. Only valid when PT is 200(SR).
- SentRTP - The sender's last RTP timestamp. Only valid when PT is 200(SR).
- SentPackets - The number of packets the sender has sent. Only valid when PT is 200(SR).
- SentOctets - The number of bytes the sender has sent. Only valid when PT is 200(SR).
- ReportXSourceSSRC - The SSRC for the source of this report block.
- ReportXFractionLost - The fraction of RTP data packets from ReportXSourceSSRC lost since the previous SR or RR report was sent.
- ReportXCumulativeLost - The total number of RTP data packets from ReportXSourceSSRC lost since the beginning of reception.
- ReportXHighestSequence - The highest sequence number received in an RTP data packet from ReportXSourceSSRC.
- ReportXSequenceNumberCycles - The number of sequence number cycles seen for the RTP data received from ReportXSourceSSRC.
- ReportXIAJitter - An estimate of the statistical variance of the RTP data packet interarrival time, measured in timestamp units.
- ReportXLSR - The last SR timestamp received from ReportXSourceSSRC. If no SR has been received from ReportXSourceSSRC, then 0.
- ReportXDLSR - The delay, expressed in units of 1/65536 seconds, between receiving the last SR packet from ReportXSourceSSRC and sending this report.

Class

REPORTING

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SendFAX

SendFAX

Synopsis

Raised when a send fax operation has completed.

Description

Syntax

```
Event: SendFAX
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
LocalStationID: <value>
RemoteStationID: <value>
PagesTransferred: <value>
Resolution: <value>
TransferRate: <value>
FileName: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `LocalStationID` - The value of the `LOCALSTATIONID` channel variable
- `RemoteStationID` - The value of the `REMOTESTATIONID` channel variable
- `PagesTransferred` - The number of pages that have been transferred
- `Resolution` - The negotiated resolution
- `TransferRate` - The negotiated transfer rate
- `FileName` - The files being affected by the fax operation

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SessionLimit

SessionLimit

Synopsis

Raised when a request fails due to exceeding the number of allowed concurrent sessions for that service.

Description

Syntax

```
Event: SessionLimit
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SessionTimeout

SessionTimeout

Synopsis

Raised when a SIP session times out.

Description

Syntax

```
Event: SessionTimeout
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Source: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Source` - The source of the session timeout.
    - `RTPTimeout`
    - `SIPSessionTimer`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Shutdown

Shutdown

Synopsis

Raised when Asterisk is shutdown or restarted.

Description

Syntax

```
Event: Shutdown
Shutdown: <value>
Restart: <value>
```

Arguments

- `Shutdown` - Whether the shutdown is proceeding cleanly (all channels were hungup successfully) or uncleanly (channels will be terminated)
    - `Uncleanly`
    - `Cleanly`
- `Restart` - Whether or not a restart will occur.
    - `True`
    - `False`

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SIPQualifyPeerDone

SIPQualifyPeerDone

Synopsis

Raised when SIPQualifyPeer has finished qualifying the specified peer.

Description

Syntax

```
Event: SIPQualifyPeerDone
Peer: <value>
ActionID: <value>
```

Arguments

- `Peer` - The name of the peer.
- `ActionID` - This is only included if an ActionID Header was sent with the action request, in which case it will be that ActionID.

Class

CALL

See Also

- Asterisk 13 ManagerAction_SIPqualifypeer

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SoftHangupRequest

SoftHangupRequest

Synopsis

Raised when a soft hangup is requested with a specific cause code.

Description

Syntax

```
Event: SoftHangupRequest
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Cause: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Cause` - A numeric cause code for why the channel was hung up.

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SpanAlarm

SpanAlarm

Synopsis

Raised when an alarm is set on a DAHDI span.

Description

Syntax

```
Event: SpanAlarm
Span: <value>
Alarm: <value>
```

Arguments

- `Span` - The span on which the alarm occurred.
- `Alarm` - A textual description of the alarm that occurred.

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SpanAlarmClear

SpanAlarmClear

Synopsis

Raised when an alarm is cleared on a DAHDI span.

Description

Syntax

```
Event: SpanAlarmClear
Span: <value>
```

Arguments

- `Span` - The span on which the alarm was cleared.

Class

SYSTEM

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Status

Status

Synopsis

Raised in response to a Status command.

Description

Syntax

```
Event: Status
[ActionID:] <value>
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Type: <value>
DNID: <value>
TimeToHangup: <value>
BridgeID: <value>
Linkedid: <value>
Application: <value>
Data: <value>
Nativeformats: <value>
Readformat: <value>
Readtrans: <value>
Writeformat: <value>
Writetrans: <value>
Callgroup: <value>
Pickupgroup: <value>
Seconds: <value>
```

Arguments

- `ActionID`
- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Type` - Type of channel
- `DNID` - Dialed number identifier
- `TimeToHangup` - Absolute lifetime of the channel
- `BridgeID` - Identifier of the bridge the channel is in, may be empty if not in one
- `Linkedid`
- `Application` - Application currently executing on the channel

* `Data` - Data given to the currently executing channel
* `Nativeformats` - Media formats the connected party is willing to send or receive
* `Readformat` - Media formats that frames from the channel are received in
* `Readtrans` - Translation path for media received in native formats
* `Writeformat` - Media formats that frames to the channel are accepted in
* `Writetrans` - Translation path for media sent to the connected party
* `Callgroup` - Configured call group on the channel
* `Pickupgroup` - Configured pickup group on the channel
* `Seconds` - Number of seconds the channel has been active

Class

CALL

See Also

* Asterisk 13 ManagerAction_Status

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_SuccessfulAuth

SuccessfulAuth

Synopsis

Raised when a request successfully authenticates with a service.

Description

Syntax

```
Event: SuccessfulAuth
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
UsingPassword: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
  - `Informational`
  - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `UsingPassword` - Whether or not the authentication attempt included a password.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_TransportDetail

TransportDetail

Synopsis

Provide details about an authentication section.

Description

Syntax

```
Event: TransportDetail
ObjectType: <value>
ObjectName: <value>
Protocol: <value>
Bind: <value>
AsycOperations: <value>
CaListFile: <value>
CertFile: <value>
PrivKeyFile: <value>
Password: <value>
ExternalSignalingAddress: <value>
ExternalSignalingPort: <value>
ExternalMediaAddress: <value>
Domain: <value>
VerifyServer: <value>
VerifyClient: <value>
RequireClientCert: <value>
Method: <value>
Cipher: <value>
LocalNet: <value>
Tos: <value>
Cos: <value>
WebsocketWriteTimeout: <value>
EndpointName: <value>
```

Arguments

- `ObjectType` - The object's type. This will always be 'transport'.
- `ObjectName` - The name of this object.
- `Protocol` - Protocol to use for SIP traffic
- `Bind` - IP Address and optional port to bind to for this transport
- `AsycOperations` - Number of simultaneous Asynchronous Operations
- `CaListFile` - File containing a list of certificates to read (TLS ONLY)
- `CertFile` - Certificate file for endpoint (TLS ONLY)
- `PrivKeyFile` - Private key file (TLS ONLY)
- `Password` - Password required for transport
- `ExternalSignalingAddress` - External address for SIP signalling
- `ExternalSignalingPort` - External port for SIP signalling
- `ExternalMediaAddress` - External IP address to use in RTP handling
- `Domain` - Domain the transport comes from
- `VerifyServer` - Require verification of server certificate (TLS ONLY)
- `VerifyClient` - Require verification of client certificate (TLS ONLY)
- `RequireClientCert` - Require client certificate (TLS ONLY)
- `Method` - Method of SSL transport (TLS ONLY)
- `Cipher` - Preferred Cryptography Cipher (TLS ONLY)
- `LocalNet` - Network to consider local (used for NAT purposes).
- `Tos` - Enable TOS for the signalling sent over this transport
- `Cos` - Enable COS for the signalling sent over this transport
- `WebsocketWriteTimeout` - The timeout (in milliseconds) to set on WebSocket connections.
- `EndpointName` - The name of the endpoint associated with this information.

Class

COMMAND

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_UnexpectedAddress

UnexpectedAddress

Synopsis

Raised when a request has a different source address then what is expected for a session already in progress with a service.

Description

Syntax

```
Event: UnexpectedAddress
EventTV: <value>
Severity: <value>
Service: <value>
EventVersion: <value>
AccountID: <value>
SessionID: <value>
LocalAddress: <value>
RemoteAddress: <value>
ExpectedAddress: <value>
[Module:] <value>
[SessionTV:] <value>
```

Arguments

- `EventTV` - The time the event was detected.
- `Severity` - A relative severity of the security event.
    - `Informational`
    - `Error`
- `Service` - The Asterisk service that raised the security event.
- `EventVersion` - The version of this event.
- `AccountID` - The Service account associated with the security event notification.
- `SessionID` - A unique identifier for the session in the service that raised the event.
- `LocalAddress` - The address of the Asterisk service that raised the security event.
- `RemoteAddress` - The remote address of the entity that caused the security event to be raised.
- `ExpectedAddress` - The address that the request was expected to use.
- `Module` - If available, the name of the module that raised the event.
- `SessionTV` - The timestamp reported by the session.

Class

SECURITY

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_Unhold

Unhold

Synopsis

Raised when a channel goes off hold.

Description

Syntax

```
Event: Unhold
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_UnParkedCall

UnParkedCall

Synopsis

Raised when a channel leaves a parking lot because it was retrieved from the parking lot and reconnected.

Description

Syntax

```
Event: UnParkedCall
ParkeeChannel: <value>
ParkeeChannelState: <value>
ParkeeChannelStateDesc: <value>
ParkeeCallerIDNum: <value>
ParkeeCallerIDName: <value>
ParkeeConnectedLineNum: <value>
ParkeeConnectedLineName: <value>
ParkeeAccountCode: <value>
ParkeeContext: <value>
ParkeeExten: <value>
ParkeePriority: <value>
ParkeeUniqueid: <value>
ParkerChannel: <value>
ParkerChannelState: <value>
ParkerChannelStateDesc: <value>
ParkerCallerIDNum: <value>
ParkerCallerIDName: <value>
ParkerConnectedLineNum: <value>
ParkerConnectedLineName: <value>
ParkerAccountCode: <value>
ParkerContext: <value>
ParkerExten: <value>
ParkerPriority: <value>
ParkerUniqueid: <value>
ParkerDialString: <value>
Parkinglot: <value>
ParkingSpace: <value>
ParkingTimeout: <value>
ParkingDuration: <value>
RetrieverChannel: <value>
RetrieverChannelState: <value>
RetrieverChannelStateDesc: <value>
RetrieverCallerIDNum: <value>
RetrieverCallerIDName: <value>
RetrieverConnectedLineNum: <value>
RetrieverConnectedLineName: <value>
RetrieverAccountCode: <value>
RetrieverContext: <value>
RetrieverExten: <value>
RetrieverPriority: <value>
RetrieverUniqueid: <value>
```

Arguments

- `ParkeeChannel`
- `ParkeeChannelState` - A numeric code for the channel's current state, related to ParkeeChannelStateDesc
- `ParkeeChannelStateDesc`
  - `Down`
  - `Rsrvd`
  - `OffHook`
  - `Dialing`
  - `Ring`
  - `Ringing`
  - `Up`
  - `Busy`
  - `Dialing Offhook`
  - `Pre-ring`
  - `Unknown`
- `ParkeeCallerIDNum`
- `ParkeeCallerIDName`
- `ParkeeConnectedLineNum`
- `ParkeeConnectedLineName`
- `ParkeeAccountCode`

* `ParkeeContext`
* `ParkeeExten`
* `ParkeePriority`
* `ParkeeUniqueid`
* `ParkerChannel`
* `ParkerChannelState` - A numeric code for the channel's current state, related to ParkerChannelStateDesc
* `ParkerChannelStateDesc`
    * `Down`
    * `Rsrvd`
    * `OffHook`
    * `Dialing`
    * `Ring`
    * `Ringing`
    * `Up`
    * `Busy`
    * `Dialing Offhook`
    * `Pre-ring`
    * `Unknown`
* `ParkerCallerIDNum`
* `ParkerCallerIDName`
* `ParkerConnectedLineNum`
* `ParkerConnectedLineName`
* `ParkerAccountCode`
* `ParkerContext`
* `ParkerExten`
* `ParkerPriority`
* `ParkerUniqueid`
* `ParkerDialString` - Dial String that can be used to call back the parker on ParkingTimeout.
* `Parkinglot` - Name of the parking lot that the parkee is parked in
* `ParkingSpace` - Parking Space that the parkee is parked in
* `ParkingTimeout` - Time remaining until the parkee is forcefully removed from parking in seconds
* `ParkingDuration` - Time the parkee has been in the parking bridge (in seconds)
* `RetrieverChannel`
* `RetrieverChannelState` - A numeric code for the channel's current state, related to RetrieverChannelStateDesc
* `RetrieverChannelStateDesc`
    * `Down`
    * `Rsrvd`
    * `OffHook`
    * `Dialing`
    * `Ring`
    * `Ringing`
    * `Up`
    * `Busy`
    * `Dialing Offhook`
    * `Pre-ring`
    * `Unknown`
* `RetrieverCallerIDNum`
* `RetrieverCallerIDName`
* `RetrieverConnectedLineNum`
* `RetrieverConnectedLineName`
* `RetrieverAccountCode`
* `RetrieverContext`
* `RetrieverExten`
* `RetrieverPriority`
* `RetrieverUniqueid`

Class

CALL

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_UserEvent

UserEvent

Synopsis

A user defined event raised from the dialplan.

Description

Event may contain additional arbitrary parameters in addition to optional bridge and endpoint snapshots. Multiple snapshots of the same type are prefixed with a numeric value.

Syntax

```
Event: UserEvent
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
UserEvent: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `UserEvent` - The event name, as specified in the dialplan.

Class

USER

See Also

- Asterisk 13 Application_UserEvent

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ManagerEvent_VarSet

VarSet

Synopsis

Raised when a variable local to the gosub stack frame is set due to a subroutine call.

Description

Syntax

```
Event: VarSet
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Variable: <value>
Value: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Variable` - The LOCAL variable being set.

> **Note**
> The variable name will always be enclosed with `LOCAL()`

- `Value` - The new value of the variable.

Class

DIALPLAN

See Also

- Asterisk 13 Application_GoSub
- Asterisk 13 AGICommand_gosub
- Asterisk 13 Function_LOCAL
- Asterisk 13 Function_LOCAL_PEEK

Synopsis

Raised when a variable is shared between channels.

Description

Syntax

```
Event: VarSet
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Variable: <value>
Value: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Variable` - The SHARED variable being set.

> **Note**
> The variable name will always be enclosed with `SHARED()`

- `Value` - The new value of the variable.

Class

DIALPLAN

See Also

- Asterisk 13 Function_SHARED

Synopsis

Raised when a variable is set to a particular value.

Description

Syntax


```
Event: VarSet
Channel: <value>
ChannelState: <value>
ChannelStateDesc: <value>
CallerIDNum: <value>
CallerIDName: <value>
ConnectedLineNum: <value>
ConnectedLineName: <value>
AccountCode: <value>
Context: <value>
Exten: <value>
Priority: <value>
Uniqueid: <value>
Variable: <value>
Value: <value>
```

Arguments

- `Channel`
- `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`
- `CallerIDNum`
- `CallerIDName`
- `ConnectedLineNum`
- `ConnectedLineName`
- `AccountCode`
- `Context`
- `Exten`
- `Priority`
- `Uniqueid`
- `Variable` - The variable being set.
- `Value` - The new value of the variable.

Class

DIALPLAN

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 ARI

Asterisk 13 Applications REST API

Applications

Method	Path	Return Model	Summary
GET	/applications	List[Application]	List all applications.
GET	/applications/{applicationName}	Application	Get details of an application.
POST	/applications/{applicationName}/subscription	Application	Subscribe an application to a event source.
DELETE	/applications/{applicationName}/subscription	Application	Unsubscribe an application from an event source.

GET /applications

List all applications.

GET /applications/{applicationName}

Get details of an application.

Path parameters

- applicationName: string - Application's name

Error Responses

- 404 - Application does not exist.

POST /applications/{applicationName}/subscription

Subscribe an application to a event source. Returns the state of the application after the subscriptions have changed

Path parameters

- applicationName: string - Application's name

Query parameters

- eventSource: string - **(required)** URI for event source (channel:{channelId}, bridge:{bridgeId}, endpoint:{tech}[/{resource}], deviceState:{deviceName}
    - Allows comma separated values.

Error Responses

- 400 - Missing parameter.
- 404 - Application does not exist.
- 422 - Event source does not exist.

DELETE /applications/{applicationName}/subscription

Unsubscribe an application from an event source. Returns the state of the application after the subscriptions have changed

Path parameters

- applicationName: string - Application's name

Query parameters

- eventSource: string - **(required)** URI for event source (channel:{channelId}, bridge:{bridgeId}, endpoint:{tech}[/{resource}], deviceState:{deviceName}

    - Allows comma separated values.

Error Responses

- 400 - Missing parameter; event source scheme not recognized.
- 404 - Application does not exist.
- 409 - Application not subscribed to event source.
- 422 - Event source does not exist.

Asterisk 13 Asterisk REST API

Asterisk

Method	Path	Return Model	Summary
GET	/asterisk/info	AsteriskInfo	Gets Asterisk system information.
GET	/asterisk/variable	Variable	Get the value of a global variable.
POST	/asterisk/variable	void	Set the value of a global variable.

GET /asterisk/info

Gets Asterisk system information.

Query parameters

- only: string - Filter information returned
    - Allows comma separated values.

GET /asterisk/variable

Get the value of a global variable.

Query parameters

- variable: string - **(required)** The variable to get

Error Responses

- 400 - Missing variable parameter.

POST /asterisk/variable

Set the value of a global variable.

Query parameters

- variable: string - **(required)** The variable to set
- value: string - The value to set the variable to

Error Responses

- 400 - Missing variable parameter.

Asterisk 13 Bridges REST API

Bridges

Method	Path	Return Model	Summary
GET	/bridges	List[Bridge]	List all active bridges in Asterisk.
POST	/bridges	Bridge	Create a new bridge.
POST	/bridges/{bridgeId}	Bridge	Create a new bridge or updates an existing one.
GET	/bridges/{bridgeId}	Bridge	Get bridge details.
DELETE	/bridges/{bridgeId}	void	Shut down a bridge.
POST	/bridges/{bridgeId}/addChannel	void	Add a channel to a bridge.
POST	/bridges/{bridgeId}/removeChannel	void	Remove a channel from a bridge.
POST	/bridges/{bridgeId}/moh	void	Play music on hold to a bridge or change the MOH class that is playing.
DELETE	/bridges/{bridgeId}/moh	void	Stop playing music on hold to a bridge.
POST	/bridges/{bridgeId}/play	Playback	Start playback of media on a bridge.
POST	/bridges/{bridgeId}/play/{playbackId}	Playback	Start playback of media on a bridge.
POST	/bridges/{bridgeId}/record	LiveRecording	Start a recording.

GET /bridges

List all active bridges in Asterisk.

POST /bridges

Create a new bridge. This bridge persists until it has been shut down, or Asterisk has been shut down.

Query parameters

- type: string - Comma separated list of bridge type attributes (mixing, holding, dtmf_events, proxy_media).
- bridgeId: string - Unique ID to give to the bridge being created.
- name: string - Name to give to the bridge being created.

POST /bridges/{bridgeId}

Create a new bridge or updates an existing one. This bridge persists until it has been shut down, or Asterisk has been shut down.

Path parameters

- bridgeId: string - Unique ID to give to the bridge being created.

Query parameters

- type: string - Comma separated list of bridge type attributes (mixing, holding, dtmf_events, proxy_media) to set.
- name: string - Set the name of the bridge.

GET /bridges/{bridgeId}

Get bridge details.

Path parameters

* bridgeId: string - Bridge's id

Error Responses

* 404 - Bridge not found

DELETE /bridges/{bridgeId}

Shut down a bridge. If any channels are in this bridge, they will be removed and resume whatever they were doing beforehand.

Path parameters

* bridgeId: string - Bridge's id

Error Responses

* 404 - Bridge not found

POST /bridges/{bridgeId}/addChannel

Add a channel to a bridge.

Path parameters

* bridgeId: string - Bridge's id

Query parameters

* channel: string - **(required)** Ids of channels to add to bridge
    * Allows comma separated values.
* role: string - Channel's role in the bridge

Error Responses

* 400 - Channel not found
* 404 - Bridge not found
* 409 - Bridge not in Stasis application; Channel currently recording
* 422 - Channel not in Stasis application

POST /bridges/{bridgeId}/removeChannel

Remove a channel from a bridge.

Path parameters

* bridgeId: string - Bridge's id

Query parameters

* channel: string - **(required)** Ids of channels to remove from bridge
    * Allows comma separated values.

Error Responses

* 400 - Channel not found
* 404 - Bridge not found
* 409 - Bridge not in Stasis application
* 422 - Channel not in this bridge

POST /bridges/{bridgeId}/moh

Play music on hold to a bridge or change the MOH class that is playing.

Path parameters

* bridgeId: string - Bridge's id

Query parameters

* mohClass: string - Channel's id

Error Responses

* 404 - Bridge not found
* 409 - Bridge not in Stasis application

DELETE /bridges/{bridgeId}/moh

Stop playing music on hold to a bridge. This will only stop music on hold being played via POST bridges/{bridgeId}/moh.

Path parameters

* bridgeId: string - Bridge's id

Error Responses

* 404 - Bridge not found
* 409 - Bridge not in Stasis application

POST /bridges/{bridgeId}/play

Start playback of media on a bridge. The media URI may be any of a number of URI's. Currently sound:, recording:, number:, digits:, characters:, and tone: URI's are supported. This operation creates a playback resource that can be used to control the playback of media (pause, rewind, fast forward, etc.)

Path parameters

* bridgeId: string - Bridge's id

Query parameters

* media: string - **(required)** Media's URI to play.
* lang: string - For sounds, selects language for sound.
* offsetms: int - Number of media to skip before playing.
* skipms: int = 3000 - Number of milliseconds to skip for forward/reverse operations.
* playbackId: string - Playback Id.

Error Responses

* 404 - Bridge not found
* 409 - Bridge not in a Stasis application

POST /bridges/{bridgeId}/play/{playbackId}

Start playback of media on a bridge. The media URI may be any of a number of URI's. Currently sound: and recording: URI's are supported. This operation creates a playback resource that can be used to control the playback of media (pause, rewind, fast forward, etc.)

Path parameters

* bridgeId: string - Bridge's id
* playbackId: string - Playback ID.

Query parameters

* media: string - **(required)** Media's URI to play.
* lang: string - For sounds, selects language for sound.
* offsetms: int - Number of media to skip before playing.
* skipms: int = 3000 - Number of milliseconds to skip for forward/reverse operations.

Error Responses

- 404 - Bridge not found
- 409 - Bridge not in a Stasis application

POST /bridges/{bridgeId}/record

Start a recording. This records the mixed audio from all channels participating in this bridge.

Path parameters

- bridgeId: string - Bridge's id

Query parameters

- name: string - **(required)** Recording's filename
- format: string - **(required)** Format to encode audio in
- maxDurationSeconds: int - Maximum duration of the recording, in seconds. 0 for no limit.
- maxSilenceSeconds: int - Maximum duration of silence, in seconds. 0 for no limit.
- ifExists: string = fail - Action to take if a recording with the same name already exists.
- beep: boolean - Play beep when recording begins
- terminateOn: string = none - DTMF input to terminate recording.

Error Responses

- 400 - Invalid parameters
- 404 - Bridge not found
- 409 - Bridge is not in a Stasis application; A recording with the same name already exists on the system and can not be overwritten because it is in progress or ifExists=fail
- 422 - The format specified is unknown on this system

Asterisk 13 Channels REST API

Channels

Method	Path	Return Model	Summary
GET	/channels	List[Channel]	List all active channels in Asterisk.
POST	/channels	Channel	Create a new channel (originate).
GET	/channels/{channelId}	Channel	Channel details.
POST	/channels/{channelId}	Channel	Create a new channel (originate with id).
DELETE	/channels/{channelId}	void	Delete (i.e. hangup) a channel.
POST	/channels/{channelId}/continue	void	Exit application; continue execution in the dialplan.
POST	/channels/{channelId}/answer	void	Answer a channel.
POST	/channels/{channelId}/ring	void	Indicate ringing to a channel.
DELETE	/channels/{channelId}/ring	void	Stop ringing indication on a channel if locally generated.
POST	/channels/{channelId}/dtmf	void	Send provided DTMF to a given channel.
POST	/channels/{channelId}/mute	void	Mute a channel.
DELETE	/channels/{channelId}/mute	void	Unmute a channel.
POST	/channels/{channelId}/hold	void	Hold a channel.
DELETE	/channels/{channelId}/hold	void	Remove a channel from hold.
POST	/channels/{channelId}/moh	void	Play music on hold to a channel.
DELETE	/channels/{channelId}/moh	void	Stop playing music on hold to a channel.
POST	/channels/{channelId}/silence	void	Play silence to a channel.
DELETE	/channels/{channelId}/silence	void	Stop playing silence to a channel.
POST	/channels/{channelId}/play	Playback	Start playback of media.
POST	/channels/{channelId}/play/{playbackId}	Playback	Start playback of media and specify the playbackId.
POST	/channels/{channelId}/record	LiveRecording	Start a recording.
GET	/channels/{channelId}/variable	Variable	Get the value of a channel variable or function.
POST	/channels/{channelId}/variable	void	Set the value of a channel variable or function.
POST	/channels/{channelId}/snoop	Channel	Start snooping.
POST	/channels/{channelId}/snoop/{snoopId}	Channel	Start snooping.

GET /channels

List all active channels in Asterisk.

POST /channels

Create a new channel (originate). The new channel is created immediately and a snapshot of it returned. If a Stasis application is provided it will be automatically subscribed to the originated channel for further events and updates.

Query parameters

- endpoint: string - **(required)** Endpoint to call.
- extension: string - The extension to dial after the endpoint answers
- context: string - The context to dial after the endpoint answers. If omitted, uses 'default'
- priority: long - The priority to dial after the endpoint answers. If omitted, uses 1
- app: string - The application that is subscribed to the originated channel, and passed to the Stasis application.
- appArgs: string - The application arguments to pass to the Stasis application.
- callerId: string - CallerID to use when dialing the endpoint or extension.
- timeout: int = 30 - Timeout (in seconds) before giving up dialing, or -1 for no timeout.
- channelId: string - The unique id to assign the channel on creation.
- otherChannelId: string - The unique id to assign the second channel when using local channels.

Body parameter

- variables: containers - The "variables" key in the body object holds variable key/value pairs to set on the channel on creation. Other keys in the body object are interpreted as query parameters. Ex. { "endpoint": "SIP/Alice", "variables": { "CALLERID(name)": "Alice" } }

Error Responses

- 400 - Invalid parameters for originating a channel.

GET /channels/{channelId}

Channel details.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found

POST /channels/{channelId}

Create a new channel (originate with id). The new channel is created immediately and a snapshot of it returned. If a Stasis application is provided it will be automatically subscribed to the originated channel for further events and updates.

Path parameters

- channelId: string - The unique id to assign the channel on creation.

Query parameters

- endpoint: string - **(required)** Endpoint to call.
- extension: string - The extension to dial after the endpoint answers
- context: string - The context to dial after the endpoint answers. If omitted, uses 'default'
- priority: long - The priority to dial after the endpoint answers. If omitted, uses 1
- app: string - The application that is subscribed to the originated channel, and passed to the Stasis application.
- appArgs: string - The application arguments to pass to the Stasis application.
- callerId: string - CallerID to use when dialing the endpoint or extension.
- timeout: int = 30 - Timeout (in seconds) before giving up dialing, or -1 for no timeout.
- otherChannelId: string - The unique id to assign the second channel when using local channels.

Body parameter

- variables: containers - The "variables" key in the body object holds variable key/value pairs to set on the channel on creation. Other keys in the body object are interpreted as query parameters. Ex. { "endpoint": "SIP/Alice", "variables": { "CALLERID(name)": "Alice" } }

Error Responses

- 400 - Invalid parameters for originating a channel.

DELETE /channels/{channelId}

Delete (i.e. hangup) a channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- reason: string - Reason for hanging up the channel

Error Responses

- 400 - Invalid reason for hangup provided
- 404 - Channel not found

POST /channels/{channelId}/continue

Exit application; continue execution in the dialplan.

Path parameters

- channelId: string - Channel's id

Query parameters

- context: string - The context to continue to.
- extension: string - The extension to continue to.
- priority: int - The priority to continue to.

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/answer

Answer a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/ring

Indicate ringing to a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/ring

Stop ringing indication on a channel if locally generated.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/dtmf

Send provided DTMF to a given channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- dtmf: string - DTMF To send.
- before: int - Amount of time to wait before DTMF digits (specified in milliseconds) start.
- between: int = 100 - Amount of time in between DTMF digits (specified in milliseconds).
- duration: int = 100 - Length of each DTMF digit (specified in milliseconds).
- after: int - Amount of time to wait after DTMF digits (specified in milliseconds) end.

Error Responses

- 400 - DTMF is required
- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/mute

Mute a channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- direction: string = both - Direction in which to mute audio

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/mute

Unmute a channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- direction: string = both - Direction in which to unmute audio

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/hold

Hold a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/hold

Remove a channel from hold.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/moh

Play music on hold to a channel. Using media operations such as /play on a channel playing MOH in this manner will suspend MOH without resuming automatically. If continuing music on hold is desired, the stasis application must reinitiate music on hold.

Path parameters

- channelId: string - Channel's id

Query parameters

- mohClass: string - Music on hold class to use

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/moh

Stop playing music on hold to a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/silence

Play silence to a channel. Using media operations such as /play on a channel playing silence in this manner will suspend silence without resuming automatically.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

DELETE /channels/{channelId}/silence

Stop playing silence to a channel.

Path parameters

- channelId: string - Channel's id

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/play

Start playback of media. The media URI may be any of a number of URI's. Currently sound:, recording:, number:, digits:, characters:, and tone: URI's are supported. This operation creates a playback resource that can be used to control the playback of media (pause, rewind, fast forward, etc.)

Path parameters

- channelId: string - Channel's id

Query parameters

- media: string - **(required)** Media's URI to play.
- lang: string - For sounds, selects language for sound.
- offsetms: int - Number of media to skip before playing.
- skipms: int = 3000 - Number of milliseconds to skip for forward/reverse operations.
- playbackId: string - Playback ID.

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/play/{playbackId}

Start playback of media and specify the playbackId. The media URI may be any of a number of URI's. Currently sound: and recording: URI's are supported. This operation creates a playback resource that can be used to control the playback of media (pause, rewind, fast forward, etc.)

Path parameters

- channelId: string - Channel's id
- playbackId: string - Playback ID.

Query parameters

- media: string - **(required)** Media's URI to play.
- lang: string - For sounds, selects language for sound.
- offsetms: int - Number of media to skip before playing.
- skipms: int = 3000 - Number of milliseconds to skip for forward/reverse operations.

Error Responses

- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/record

Start a recording. Record audio from a channel. Note that this will not capture audio sent to the channel. The bridge itself has a record feature if that's what you want.

Path parameters

- channelId: string - Channel's id

Query parameters

- name: string - **(required)** Recording's filename
- format: string - **(required)** Format to encode audio in
- maxDurationSeconds: int - Maximum duration of the recording, in seconds. 0 for no limit
- maxSilenceSeconds: int - Maximum duration of silence, in seconds. 0 for no limit
- ifExists: string = fail - Action to take if a recording with the same name already exists.
- beep: boolean - Play beep when recording begins
- terminateOn: string = none - DTMF input to terminate recording

Error Responses

- 400 - Invalid parameters
- 404 - Channel not found
- 409 - Channel is not in a Stasis application; the channel is currently bridged with other hcannels; A recording with the same name already exists on the system and can not be overwritten because it is in progress or ifExists=fail
- 422 - The format specified is unknown on this system

GET /channels/{channelId}/variable

Get the value of a channel variable or function.

Path parameters

- channelId: string - Channel's id

Query parameters

- variable: string - **(required)** The channel variable or function to get

Error Responses

- 400 - Missing variable parameter.
- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/variable

Set the value of a channel variable or function.

Path parameters

- channelId: string - Channel's id

Query parameters

- variable: string - **(required)** The channel variable or function to set
- value: string - The value to set the variable to

Error Responses

- 400 - Missing variable parameter.
- 404 - Channel not found
- 409 - Channel not in a Stasis application

POST /channels/{channelId}/snoop

Start snooping. Snoop (spy/whisper) on a specific channel.

Path parameters

- channelId: string - Channel's id

Query parameters

- spy: string = none - Direction of audio to spy on
- whisper: string = none - Direction of audio to whisper into
- app: string - **(required)** Application the snooping channel is placed into
- appArgs: string - The application arguments to pass to the Stasis application
- snoopId: string - Unique ID to assign to snooping channel

Error Responses

- 400 - Invalid parameters
- 404 - Channel not found

POST /channels/{channelId}/snoop/{snoopId}

Start snooping. Snoop (spy/whisper) on a specific channel.

Path parameters

- channelId: string - Channel's id
- snoopId: string - Unique ID to assign to snooping channel

Query parameters

- spy: string = none - Direction of audio to spy on
- whisper: string = none - Direction of audio to whisper into
- app: string - **(required)** Application the snooping channel is placed into
- appArgs: string - The application arguments to pass to the Stasis application

Error Responses

- 400 - Invalid parameters
- 404 - Channel not found

Asterisk 13 Devicestates REST API

Devicestates

Method	Path	Return Model	Summary
GET	/deviceStates	List[DeviceState]	List all ARI controlled device states.
GET	/deviceStates/{deviceName}	DeviceState	Retrieve the current state of a device.
PUT	/deviceStates/{deviceName}	void	Change the state of a device controlled by ARI. (Note - implicitly creates the device state).
DELETE	/deviceStates/{deviceName}	void	Destroy a device-state controlled by ARI.

GET /deviceStates

List all ARI controlled device states.

GET /deviceStates/{deviceName}

Retrieve the current state of a device.

Path parameters

- deviceName: string - Name of the device

PUT /deviceStates/{deviceName}

Change the state of a device controlled by ARI. (Note - implicitly creates the device state).

Path parameters

- deviceName: string - Name of the device

Query parameters

- deviceState: string - **(required)** Device state value

Error Responses

- 404 - Device name is missing
- 409 - Uncontrolled device specified

DELETE /deviceStates/{deviceName}

Destroy a device-state controlled by ARI.

Path parameters

- deviceName: string - Name of the device

Error Responses

- 404 - Device name is missing
- 409 - Uncontrolled device specified

Asterisk 13 Endpoints REST API

Endpoints

Method	Path	Return Model	Summary
GET	/endpoints	List[Endpoint]	List all endpoints.
PUT	/endpoints/sendMessage	void	Send a message to some technology URI or endpoint.
GET	/endpoints/{tech}	List[Endpoint]	List available endoints for a given endpoint technology.
GET	/endpoints/{tech}/{resource}	Endpoint	Details for an endpoint.
PUT	/endpoints/{tech}/{resource}/sendMessage	void	Send a message to some endpoint in a technology.

GET /endpoints

List all endpoints.

PUT /endpoints/sendMessage

Send a message to some technology URI or endpoint.

Query parameters

- to: string - **(required)** The endpoint resource or technology specific URI to send the message to. Valid resources are sip, pjsip, and xmpp.
- from: string - **(required)** The endpoint resource or technology specific identity to send this message from. Valid resources are sip, pjsip, and xmpp.
- body: string - The body of the message

Body parameter

- variables: containers -

Error Responses

- 404 - Endpoint not found

GET /endpoints/{tech}

List available endoints for a given endpoint technology.

Path parameters

- tech: string - Technology of the endpoints (sip,iax2,...)

Error Responses

- 404 - Endpoints not found

GET /endpoints/{tech}/{resource}

Details for an endpoint.

Path parameters

- tech: string - Technology of the endpoint
- resource: string - ID of the endpoint

Error Responses

- 400 - Invalid parameters for sending a message.
- 404 - Endpoints not found

PUT /endpoints/{tech}/{resource}/sendMessage

Send a message to some endpoint in a technology.

Path parameters

- tech: string - Technology of the endpoint
- resource: string - ID of the endpoint

Query parameters

- from: string - **(required)** The endpoint resource or technology specific identity to send this message from. Valid resources are sip, pjsip, and xmpp.
- body: string - The body of the message

Body parameter

- variables: containers -

Error Responses

- 400 - Invalid parameters for sending a message.
- 404 - Endpoint not found

Asterisk 13 Events REST API

Events

Method	Path	Return Model	Summary
GET	/events	Message	WebSocket connection for events.
POST	/events/user/{eventName}	void	Generate a user event.

GET /events

WebSocket connection for events.

Query parameters

- app: string - **(required)** Applications to subscribe to.
    - Allows comma separated values.

POST /events/user/{eventName}

Generate a user event.

Path parameters

- eventName: string - Event name

Query parameters

- application: string - **(required)** The name of the application that will receive this event
- source: string - URI for event source (channel:{channelId}, bridge:{bridgeId}, endpoint:{tech}/{resource}, deviceState:{deviceName}
    - Allows comma separated values.

Body parameter

- variables: containers - The "variables" key in the body object holds custom key/value pairs to add to the user event. Ex. { "variables": { "key": "value" } }

Error Responses

- 404 - Application does not exist.
- 422 - Event source not found.
- 400 - Invalid even tsource URI or userevent data.

Asterisk 13 Mailboxes REST API

Mailboxes

Method	Path	Return Model	Summary
GET	/mailboxes	List[Mailbox]	List all mailboxes.
GET	/mailboxes/{mailboxName}	Mailbox	Retrieve the current state of a mailbox.
PUT	/mailboxes/{mailboxName}	void	Change the state of a mailbox. (Note - implicitly creates the mailbox).
DELETE	/mailboxes/{mailboxName}	void	Destroy a mailbox.

GET /mailboxes

List all mailboxes.

GET /mailboxes/{mailboxName}

Retrieve the current state of a mailbox.

Path parameters

- mailboxName: string - Name of the mailbox

Error Responses

- 404 - Mailbox not found

PUT /mailboxes/{mailboxName}

Change the state of a mailbox. (Note - implicitly creates the mailbox).

Path parameters

- mailboxName: string - Name of the mailbox

Query parameters

- oldMessages: int - **(required)** Count of old messages in the mailbox
- newMessages: int - **(required)** Count of new messages in the mailbox

Error Responses

- 404 - Mailbox not found

DELETE /mailboxes/{mailboxName}

Destroy a mailbox.

Path parameters

- mailboxName: string - Name of the mailbox

Error Responses

- 404 - Mailbox not found

Asterisk 13 Playbacks REST API

Playbacks

Method	Path	Return Model	Summary
GET	/playbacks/{playbackId}	Playback	Get a playback's details.
DELETE	/playbacks/{playbackId}	void	Stop a playback.
POST	/playbacks/{playbackId}/control	void	Control a playback.

GET /playbacks/{playbackId}

Get a playback's details.

Path parameters

- playbackId: string - Playback's id

Error Responses

- 404 - The playback cannot be found

DELETE /playbacks/{playbackId}

Stop a playback.

Path parameters

- playbackId: string - Playback's id

Error Responses

- 404 - The playback cannot be found

POST /playbacks/{playbackId}/control

Control a playback.

Path parameters

- playbackId: string - Playback's id

Query parameters

- operation: string - **(required)** Operation to perform on the playback.

Error Responses

- 400 - The provided operation parameter was invalid
- 404 - The playback cannot be found
- 409 - The operation cannot be performed in the playback's current state

Asterisk 13 Recordings REST API

Recordings

Method	Path	Return Model	Summary
GET	/recordings/stored	List[StoredRecording]	List recordings that are complete.
GET	/recordings/stored/{recordingName}	StoredRecording	Get a stored recording's details.
DELETE	/recordings/stored/{recordingName}	void	Delete a stored recording.
POST	/recordings/stored/{recordingName}/copy	StoredRecording	Copy a stored recording.
GET	/recordings/live/{recordingName}	LiveRecording	List live recordings.
DELETE	/recordings/live/{recordingName}	void	Stop a live recording and discard it.
POST	/recordings/live/{recordingName}/stop	void	Stop a live recording and store it.
POST	/recordings/live/{recordingName}/pause	void	Pause a live recording.
DELETE	/recordings/live/{recordingName}/pause	void	Unpause a live recording.
POST	/recordings/live/{recordingName}/mute	void	Mute a live recording.
DELETE	/recordings/live/{recordingName}/mute	void	Unmute a live recording.

GET /recordings/stored

List recordings that are complete.

GET /recordings/stored/{recordingName}

Get a stored recording's details.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

DELETE /recordings/stored/{recordingName}

Delete a stored recording.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

POST /recordings/stored/{recordingName}/copy

Copy a stored recording.

Path parameters

- recordingName: string - The name of the recording to copy

Query parameters

- destinationRecordingName: string - **(required)** The destination name of the recording

Error Responses

- 404 - Recording not found
- 409 - A recording with the same name already exists on the system

GET /recordings/live/{recordingName}

List live recordings.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

DELETE /recordings/live/{recordingName}

Stop a live recording and discard it.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

POST /recordings/live/{recordingName}/stop

Stop a live recording and store it.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found

POST /recordings/live/{recordingName}/pause

Pause a live recording. Pausing a recording suspends silence detection, which will be restarted when the recording is unpaused. Paused time is not included in the accounting for maxDurationSeconds.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

DELETE /recordings/live/{recordingName}/pause

Unpause a live recording.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

POST /recordings/live/{recordingName}/mute

Mute a live recording. Muting a recording suspends silence detection, which will be restarted when the recording is unmuted.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

DELETE /recordings/live/{recordingName}/mute

Unmute a live recording.

Path parameters

- recordingName: string - The name of the recording

Error Responses

- 404 - Recording not found
- 409 - Recording not in session

Asterisk 13 REST Data Models

AsteriskInfo

Asterisk system information

Expand source

```
{
  "properties": {
    "status": {
      "required": false,
      "type": "StatusInfo",
      "description": "Info about Asterisk status"
    },
    "config": {
      "required": false,
      "type": "ConfigInfo",
      "description": "Info about Asterisk configuration"
    },
    "build": {
      "required": false,
      "type": "BuildInfo",
      "description": "Info about how Asterisk was built"
    },
    "system": {
      "required": false,
      "type": "SystemInfo",
      "description": "Info about the system running Asterisk"
    }
  },
  "id": "AsteriskInfo",
  "description": "Asterisk system information"
}
```

- build: BuildInfo *(optional)* - Info about how Asterisk was built
- config: ConfigInfo *(optional)* - Info about Asterisk configuration
- status: StatusInfo *(optional)* - Info about Asterisk status
- system: SystemInfo *(optional)* - Info about the system running Asterisk

BuildInfo

Info about how Asterisk was built

Expand source


```
{
  "properties": {
    "kernel": {
      "required": true,
      "type": "string",
      "description": "Kernel version Asterisk was built on."
    },
    "machine": {
      "required": true,
      "type": "string",
      "description": "Machine architecture (x86_64, i686, ppc, etc.)"
    },
    "user": {
      "required": true,
      "type": "string",
      "description": "Username that build Asterisk"
    },
    "date": {
      "required": true,
      "type": "string",
      "description": "Date and time when Asterisk was built."
    },
    "os": {
      "required": true,
      "type": "string",
      "description": "OS Asterisk was built on."
    },
    "options": {
      "required": true,
      "type": "string",
      "description": "Compile time options, or empty string if default."
    }
  },
  "id": "BuildInfo",
  "description": "Info about how Asterisk was built"
}
```


- date: string - Date and time when Asterisk was built.
- kernel: string - Kernel version Asterisk was built on.
- machine: string - Machine architecture (x86_64, i686, ppc, etc.)
- options: string - Compile time options, or empty string if default.
- os: string - OS Asterisk was built on.
- user: string - Username that build Asterisk

ConfigInfo

Info about Asterisk configuration

Expand source


```
{
  "properties": {
    "name": {
      "required": true,
      "type": "string",
      "description": "Asterisk system name."
    },
    "default_language": {
      "required": true,
      "type": "string",
      "description": "Default language for media playback."
    },
    "max_load": {
      "required": false,
      "type": "double",
      "description": "Maximum load avg on system."
    },
    "setid": {
      "required": true,
      "type": "SetId",
      "description": "Effective user/group id for running Asterisk."
    },
    "max_open_files": {
      "required": false,
      "type": "int",
      "description": "Maximum number of open file handles (files, sockets)."
    },
    "max_channels": {
      "required": false,
      "type": "int",
      "description": "Maximum number of simultaneous channels."
    }
  },
  "id": "ConfigInfo",
  "description": "Info about Asterisk configuration"
}
```


- default_language: string - Default language for media playback.
- max_channels: int *(optional)* - Maximum number of simultaneous channels.
- max_load: double *(optional)* - Maximum load avg on system.
- max_open_files: int *(optional)* - Maximum number of open file handles (files, sockets).
- name: string - Asterisk system name.
- setid: SetId - Effective user/group id for running Asterisk.

SetId

Effective user/group id

Expand source


```
{
  "properties": {
    "group": {
      "required": true,
      "type": "string",
      "description": "Effective group id."
    },
    "user": {
      "required": true,
      "type": "string",
      "description": "Effective user id."
    }
  },
  "id": "SetId",
  "description": "Effective user/group id"
}
```


- group: string - Effective group id.
- user: string - Effective user id.

StatusInfo

Info about Asterisk status

Expand source


```
{
  "properties": {
    "last_reload_time": {
      "required": true,
      "type": "Date",
      "description": "Time when Asterisk was last reloaded."
    },
    "startup_time": {
      "required": true,
      "type": "Date",
      "description": "Time when Asterisk was started."
    }
  },
  "id": "StatusInfo",
  "description": "Info about Asterisk status"
}
```


- last_reload_time: Date - Time when Asterisk was last reloaded.
- startup_time: Date - Time when Asterisk was started.

SystemInfo

Info about Asterisk

Expand source


```
{
  "properties": {
    "entity_id": {
      "required": true,
      "type": "string",
      "description": ""
    },
    "version": {
      "required": true,
      "type": "string",
      "description": "Asterisk version."
    }
  },
  "id": "SystemInfo",
  "description": "Info about Asterisk"
}
```


- entity_id: string
- version: string - Asterisk version.

Variable

The value of a channel variable

Expand source


```
{
  "properties": {
    "value": {
      "required": true,
      "type": "string",
      "description": "The value of the variable requested"
    }
  },
  "id": "Variable",
  "description": "The value of a channel variable"
}
```


- value: string - The value of the variable requested

Endpoint

An external device that may offer/accept calls to/from Asterisk.

Unlike most resources, which have a single unique identifier, an endpoint is uniquely identified by the technology/resource pair.

Expand source


```
{
  "properties": {
    "resource": {
      "required": true,
      "type": "string",
      "description": "Identifier of the endpoint, specific to the given technology."
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "unknown",
          "offline",
          "online"
        ]
      },
      "required": false,
      "type": "string",
      "description": "Endpoint's state"
    },
    "technology": {
      "required": true,
      "type": "string",
      "description": "Technology of the endpoint"
    },
    "channel_ids": {
      "required": true,
      "type": "List[string]",
      "description": "Id's of channels associated with this endpoint"
    }
  },
  "id": "Endpoint",
  "description": "An external device that may offer/accept calls to/from
Asterisk.\n\nUnlike most resources, which have a single unique identifier, an endpoint is
uniquely identified by the technology/resource pair."
}
```


- channel_ids: List[string] - Id's of channels associated with this endpoint
- resource: string - Identifier of the endpoint, specific to the given technology.
- state: string *(optional)* - Endpoint's state
- technology: string - Technology of the endpoint

TextMessage

A text message.

Expand source


```
{
  "properties": {
    "body": {
      "required": true,
      "type": "string",
      "description": "The text of the message."
    },
    "to": {
      "required": true,
      "type": "string",
      "description": "A technology specific URI specifying the destination of the message. Valid technologies include sip, pjsip, and xmp. The destination of a message should be an endpoint."
    },
    "variables": {
      "required": false,
      "type": "List[TextMessageVariable]",
      "description": "Technology specific key/value pairs associated with the message."
    },
    "from": {
      "required": true,
      "type": "string",
      "description": "A technology specific URI specifying the source of the message. For sip and pjsip technologies, any SIP URI can be specified. For xmpp, the URI must correspond to the client connection being used to send the message."
    }
  },
  "id": "TextMessage",
  "description": "A text message."
}
```


- body: string - The text of the message.
- from: string - A technology specific URI specifying the source of the message. For sip and pjsip technologies, any SIP URI can be specified. For xmpp, the URI must correspond to the client connection being used to send the message.
- to: string - A technology specific URI specifying the destination of the message. Valid technologies include sip, pjsip, and xmp. The destination of a message should be an endpoint.
- variables: List[TextMessageVariable] *(optional)* - Technology specific key/value pairs associated with the message.

TextMessageVariable

A key/value pair variable in a text message.

Expand source

```
{
  "properties": {
    "value": {
      "required": true,
      "type": "string",
      "description": "The value of the variable."
    },
    "key": {
      "required": true,
      "type": "string",
      "description": "A unique key identifying the variable."
    }
  },
  "id": "TextMessageVariable",
  "description": "A key/value pair variable in a text message."
}
```

- key: string - A unique key identifying the variable.
- value: string - The value of the variable.

CallerID

Caller identification

Expand source

```
{
  "properties": {
    "name": {
      "required": true,
      "type": "string"
    },
    "number": {
      "required": true,
      "type": "string"
    }
  },
  "id": "CallerID",
  "description": "Caller identification"
}
```

- name: string
- number: string

Channel

A specific communication connection between Asterisk and an Endpoint.

Expand source

```
{
  "properties": {
    "accountcode": {
```



```
      "required": true,
      "type": "string"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of the channel (i.e. SIP/foo-0000a7e3)"
    },
    "caller": {
      "required": true,
      "type": "CallerID"
    },
    "creationtime": {
      "required": true,
      "type": "Date",
      "description": "Timestamp when channel was created"
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "Down",
          "Rsrved",
          "OffHook",
          "Dialing",
          "Ring",
          "Ringing",
          "Up",
          "Busy",
          "Dialing Offhook",
          "Pre-ring",
          "Unknown"
        ]
      },
      "required": true,
      "type": "string"
    },
    "connected": {
      "required": true,
      "type": "CallerID"
    },
    "dialplan": {
      "required": true,
      "type": "DialplanCEP",
      "description": "Current location in the dialplan"
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "Unique identifier of the channel.\n\nThis is the same as the
Uniqueid field in AMI."
    }
  },
  "id": "Channel",
```



```
  "description": "A specific communication connection between Asterisk and an Endpoint."
}
```

- accountcode: string
- caller: CallerID
- connected: CallerID
- creationtime: Date - Timestamp when channel was created
- dialplan: DialplanCEP - Current location in the dialplan
- id: string - Unique identifier of the channel.

This is the same as the Uniqueid field in AMI.

- name: string - Name of the channel (i.e. SIP/foo-0000a7e3)
- state: string

Dialed

Dialed channel information.

Expand source

```
{
  "properties": {},
  "id": "Dialed",
  "description": "Dialed channel information."
}
```

DialplanCEP

Dialplan location (context/extension/priority)

Expand source

```
{
  "properties": {
    "priority": {
      "required": true,
      "type": "long",
      "description": "Priority in the dialplan"
    },
    "exten": {
      "required": true,
      "type": "string",
      "description": "Extension in the dialplan"
    },
    "context": {
      "required": true,
      "type": "string",
      "description": "Context in the dialplan"
    }
  },
  "id": "DialplanCEP",
  "description": "Dialplan location (context/extension/priority)"
}
```

- context: string - Context in the dialplan
- exten: string - Extension in the dialplan
- priority: long - Priority in the dialplan

Bridge

The merging of media from one or more channels.

Everyone on the bridge receives the same audio.

Expand source


```
{
  "properties": {
    "bridge_type": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "mixing",
          "holding"
        ]
      },
      "required": true,
      "type": "string",
      "description": "Type of bridge technology"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name the creator gave the bridge"
    },
    "creator": {
      "required": true,
      "type": "string",
      "description": "Entity that created the bridge"
    },
    "channels": {
      "required": true,
      "type": "List[string]",
      "description": "Ids of channels participating in this bridge"
    },
    "bridge_class": {
      "required": true,
      "type": "string",
      "description": "Bridging class"
    },
    "technology": {
      "required": true,
      "type": "string",
      "description": "Name of the current bridging technology"
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "Unique identifier for this bridge"
    }
  },
  "id": "Bridge",
  "description": "The merging of media from one or more channels.\n\nEveryone on the
bridge receives the same audio."
}
```

- bridge_class: string - Bridging class
- bridge_type: string - Type of bridge technology
- channels: List[string] - Ids of channels participating in this bridge
- creator: string - Entity that created the bridge
- id: string - Unique identifier for this bridge
- name: string - Name the creator gave the bridge
- technology: string - Name of the current bridging technology

LiveRecording

A recording that is in progress

Expand source


```
{
  "properties": {
    "talking_duration": {
      "required": false,
      "type": "int",
      "description": "Duration of talking, in seconds, detected in the recording. This is
only available if the recording was initiated with a non-zero maxSilenceSeconds."
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Base name for the recording"
    },
    "target_uri": {
      "required": true,
      "type": "string",
      "description": "URI for the channel or bridge being recorded"
    },
    "format": {
      "required": true,
      "type": "string",
      "description": "Recording format (wav, gsm, etc.)"
    },
    "cause": {
      "required": false,
      "type": "string",
      "description": "Cause for recording failure if failed"
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "queued",
          "recording",
          "paused",
          "done",
          "failed",
          "canceled"
        ]
      },
      "required": true,
      "type": "string"
    },
    "duration": {
      "required": false,
```

```
      "type": "int",
      "description": "Duration in seconds of the recording"
    },
    "silence_duration": {
      "required": false,
      "type": "int",
      "description": "Duration of silence, in seconds, detected in the recording. This is only available if the recording was initiated with a non-zero maxSilenceSeconds."
    }
  },
  "id": "LiveRecording",
```



```
  "description": "A recording that is in progress"
}
```


- cause: string *(optional)* - Cause for recording failure if failed
- duration: int *(optional)* - Duration in seconds of the recording
- format: string - Recording format (wav, gsm, etc.)
- name: string - Base name for the recording
- silence_duration: int *(optional)* - Duration of silence, in seconds, detected in the recording. This is only available if the recording was initiated with a non-zero maxSilenceSeconds.
- state: string
- talking_duration: int *(optional)* - Duration of talking, in seconds, detected in the recording. This is only available if the recording was initiated with a non-zero maxSilenceSeconds.
- target_uri: string - URI for the channel or bridge being recorded

StoredRecording

A past recording that may be played back.

Expand source


```
{
  "properties": {
    "name": {
      "required": true,
      "type": "string"
    },
    "format": {
      "required": true,
      "type": "string"
    }
  },
  "id": "StoredRecording",
  "description": "A past recording that may be played back."
}
```


- format: string
- name: string

FormatLangPair

Identifies the format and language of a sound file

Expand

source


```
{
  "properties": {
    "language": {
      "required": true,
      "type": "string"
    },
    "format": {
      "required": true,
      "type": "string"
    }
  },
  "id": "FormatLangPair",
  "description": "Identifies the format and language of a sound file"
}
```


- format: string
- language: string

Sound

A media file that may be played back.

Expand

source


```
{
  "properties": {
    "text": {
      "required": false,
      "type": "string",
      "description": "Text description of the sound, usually the words spoken."
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "Sound's identifier."
    },
    "formats": {
      "required": true,
      "type": "List[FormatLangPair]",
      "description": "The formats and languages in which this sound is available."
    }
  },
  "id": "Sound",
  "description": "A media file that may be played back."
}
```


- formats: List[FormatLangPair] - The formats and languages in which this sound is available.
- id: string - Sound's identifier.
- text: string *(optional)* - Text description of the sound, usually the words spoken.

Playback

Object representing the playback of media to a channel

Expand source


```
{
  "properties": {
    "language": {
      "type": "string",
      "description": "For media types that support multiple languages, the language
requested for playback."
    },
    "media_uri": {
      "required": true,
      "type": "string",
      "description": "URI for the media to play back."
    },
    "id": {
      "required": true,
      "type": "string",
      "description": "ID for this playback operation"
    },
    "target_uri": {
      "required": true,
      "type": "string",
      "description": "URI for the channel or bridge to play the media on"
    },
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "queued",
          "playing",
          "complete"
        ]
      },
      "required": true,
      "type": "string",
      "description": "Current state of the playback operation."
    }
  },
  "id": "Playback",
  "description": "Object representing the playback of media to a channel"
}
```


- id: string - ID for this playback operation
- language: string *(optional)* - For media types that support multiple languages, the language requested for playback.
- media_uri: string - URI for the media to play back.
- state: string - Current state of the playback operation.
- target_uri: string - URI for the channel or bridge to play the media on

DeviceState

Represents the state of a device.

Expand source


```
{
  "properties": {
    "state": {
      "allowableValues": {
        "valueType": "LIST",
        "values": [
          "UNKNOWN",
          "NOT_INUSE",
          "INUSE",
          "BUSY",
          "INVALID",
          "UNAVAILABLE",
          "RINGING",
          "RINGINUSE",
          "ONHOLD"
        ]
      },
      "required": true,
      "type": "string",
      "description": "Device's state"
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of the device."
    }
  },
  "id": "DeviceState",
  "description": "Represents the state of a device."
}
```


- name: string - Name of the device.
- state: string - Device's state

Mailbox

Represents the state of a mailbox.

Expand source


```
{
  "properties": {
    "old_messages": {
      "required": true,
      "type": "int",
      "description": "Count of old messages in the mailbox."
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of the mailbox."
    },
    "new_messages": {
      "required": true,
      "type": "int",
      "description": "Count of new messages in the mailbox."
    }
  },
  "id": "Mailbox",
  "description": "Represents the state of a mailbox."
}
```


- name: string - Name of the mailbox.
- new_messages: int - Count of new messages in the mailbox.
- old_messages: int - Count of old messages in the mailbox.

ApplicationReplaced

Base type: Event

Notification that another WebSocket has taken over for an application.

An application may only be subscribed to by a single WebSocket at a time. If multiple WebSockets attempt to subscribe to the same application, the newer WebSocket wins, and the older one receives this event.

Expand source


```
{
  "properties": {},
  "id": "ApplicationReplaced",
  "description": "Notification that another WebSocket has taken over for an
application.\n\nAn application may only be subscribed to by a single WebSocket at a time.
If multiple WebSockets attempt to subscribe to the same application, the newer WebSocket
wins, and the older one receives this event."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.

BridgeAttendedTransfer

Base type: Event

Notification that an attended transfer has occurred.

Expand source


```
{
  "properties": {
    "replace_channel": {
      "required": false,
      "type": "Channel",
      "description": "The channel that is replacing transferer_first_leg in the swap"
    },
    "is_external": {
      "required": true,
      "type": "boolean",
      "description": "Whether the transfer was externally initiated or not"
    },
    "transferer_second_leg_bridge": {
      "type": "Bridge",
      "description": "Bridge the transferer second leg is in"
    },
    "destination_bridge": {
      "type": "string",
      "description": "Bridge that survived the merge result"
    },
    "transferer_second_leg": {
      "required": true,
      "type": "Channel",
      "description": "Second leg of the transferer"
    },
    "destination_link_second_leg": {
      "type": "Channel",
      "description": "Second leg of a link transfer result"
    },
    "destination_threeway_channel": {
      "type": "Channel",
      "description": "Transferer channel that survived the threeway result"
    },
    "transfer_target": {
      "required": false,
      "type": "Channel",
      "description": "The channel that is being transferred to"
    },
    "result": {
      "required": true,
      "type": "string",
      "description": "The result of the transfer attempt"
    },
    "destination_type": {
      "required": true,
      "type": "string",
      "description": "How the transfer was accomplished"
    },
    "destination_application": {
      "type": "string",
      "description": "Application that has been transferred into"
    },
    "destination_threeway_bridge": {
      "type": "Bridge",
      "description": "Bridge that survived the threeway result"
    },
```

```
    "destination_link_first_leg": {
      "type": "Channel",
      "description": "First leg of a link transfer result"
    },
    "transferee": {
      "required": false,
      "type": "Channel",
      "description": "The channel that is being transferred"
    },
    "transferer_first_leg": {
      "required": true,
      "type": "Channel",
      "description": "First leg of the transferer"
    },
    "transferer_first_leg_bridge": {
      "type": "Bridge",
      "description": "Bridge the transferer first leg is in"
    }
  },
  "id": "BridgeAttendedTransfer",
```

```
  "description": "Notification that an attended transfer has occurred."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- destination_application: string *(optional)* - Application that has been transferred into
- destination_bridge: string *(optional)* - Bridge that survived the merge result
- destination_link_first_leg: Channel *(optional)* - First leg of a link transfer result
- destination_link_second_leg: Channel *(optional)* - Second leg of a link transfer result
- destination_threeway_bridge: Bridge *(optional)* - Bridge that survived the threeway result
- destination_threeway_channel: Channel *(optional)* - Transferer channel that survived the threeway result
- destination_type: string - How the transfer was accomplished
- is_external: boolean - Whether the transfer was externally initiated or not
- replace_channel: Channel *(optional)* - The channel that is replacing transferer_first_leg in the swap
- result: string - The result of the transfer attempt
- transfer_target: Channel *(optional)* - The channel that is being transferred to
- transferee: Channel *(optional)* - The channel that is being transferred
- transferer_first_leg: Channel - First leg of the transferer
- transferer_first_leg_bridge: Bridge *(optional)* - Bridge the transferer first leg is in
- transferer_second_leg: Channel - Second leg of the transferer
- transferer_second_leg_bridge: Bridge *(optional)* - Bridge the transferer second leg is in

BridgeBlindTransfer

Base type: Event

Notification that a blind transfer has occurred.

Expand source


```
{
  "properties": {
    "bridge": {
      "type": "Bridge",
      "description": "The bridge being transferred"
    },
    "is_external": {
      "required": true,
      "type": "boolean",
      "description": "Whether the transfer was externally initiated or not"
    },
    "exten": {
      "required": true,
      "type": "string",
      "description": "The extension transferred to"
    },
    "result": {
      "required": true,
      "type": "string",
      "description": "The result of the transfer attempt"
    },
    "context": {
      "required": true,
      "type": "string",
      "description": "The context transferred to"
    },
    "transferee": {
      "required": false,
      "type": "Channel",
      "description": "The channel that is being transferred"
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel performing the blind transfer"
    }
  },
  "id": "BridgeBlindTransfer",
  "description": "Notification that a blind transfer has occurred."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge *(optional)* - The bridge being transferred
- channel: Channel - The channel performing the blind transfer
- context: string - The context transferred to
- exten: string - The extension transferred to
- is_external: boolean - Whether the transfer was externally initiated or not
- result: string - The result of the transfer attempt
- transferee: Channel *(optional)* - The channel that is being transferred

BridgeCreated

Base type: Event

Notification that a bridge has been created.

Expand source


```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    }
  },
  "id": "BridgeCreated",
  "description": "Notification that a bridge has been created."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge

BridgeDestroyed

Base type: Event

Notification that a bridge has been destroyed.

Expand source


```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    }
  },
  "id": "BridgeDestroyed",
  "description": "Notification that a bridge has been destroyed."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge

BridgeMerged

Base type: Event

Notification that one bridge has merged into another.

Expand

source


```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    },
    "bridge_from": {
      "required": true,
      "type": "Bridge"
    }
  },
  "id": "BridgeMerged",
  "description": "Notification that one bridge has merged into another."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge
- bridge_from: Bridge

ChannelCallerId

Base type: Event

Channel changed Caller ID.

Expand

source


```
{
  "properties": {
    "caller_presentation_txt": {
      "required": true,
      "type": "string",
      "description": "The text representation of the Caller Presentation value."
    },
    "caller_presentation": {
      "required": true,
      "type": "int",
      "description": "The integer representation of the Caller Presentation value."
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel that changed Caller ID."
    }
  },
  "id": "ChannelCallerId",
  "description": "Channel changed Caller ID."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- caller_presentation: int - The integer representation of the Caller Presentation value.
- caller_presentation_txt: string - The text representation of the Caller Presentation value.

- channel: Channel - The channel that changed Caller ID.

ChannelCreated

Base type: Event

Notification that a channel has been created.

Expand source


```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelCreated",
  "description": "Notification that a channel has been created."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel

ChannelDestroyed

Base type: Event

Notification that a channel has been destroyed.

Expand source


```
{
  "properties": {
    "cause": {
      "required": true,
      "type": "int",
      "description": "Integer representation of the cause of the hangup"
    },
    "cause_txt": {
      "required": true,
      "type": "string",
      "description": "Text representation of the cause of the hangup"
    },
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelDestroyed",
  "description": "Notification that a channel has been destroyed."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- cause: int - Integer representation of the cause of the hangup

* cause_txt: string - Text representation of the cause of the hangup
* channel: Channel

ChannelDialplan

Base type: Event

Channel changed location in the dialplan.

Expand source


```
{
  "properties": {
    "dialplan_app_data": {
      "required": true,
      "type": "string",
      "description": "The data to be passed to the application."
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel that changed dialplan location."
    },
    "dialplan_app": {
      "required": true,
      "type": "string",
      "description": "The application about to be executed."
    }
  },
  "id": "ChannelDialplan",
  "description": "Channel changed location in the dialplan."
}
```


* type: string - Indicates the type of this message.
* application: string - Name of the application receiving the event.
* timestamp: Date *(optional)* - Time at which this event was created.
* channel: Channel - The channel that changed dialplan location.
* dialplan_app: string - The application about to be executed.
* dialplan_app_data: string - The data to be passed to the application.

ChannelDtmfReceived

Base type: Event

DTMF received on a channel.

This event is sent when the DTMF ends. There is no notification about the start of DTMF

Expand source


```
{
  "properties": {
    "duration_ms": {
      "required": true,
      "type": "int",
      "description": "Number of milliseconds DTMF was received"
    },
    "digit": {
      "required": true,
      "type": "string",
      "description": "DTMF digit received (0-9, A-E, # or *)"
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel on which DTMF was received"
    }
  },
  "id": "ChannelDtmfReceived",
  "description": "DTMF received on a channel.\n\nThis event is sent when the DTMF ends.
There is no notification about the start of DTMF"
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel - The channel on which DTMF was received
- digit: string - DTMF digit received (0-9, A-E, # or *)
- duration_ms: int - Number of milliseconds DTMF was received

ChannelEnteredBridge

Base type: Event

Notification that a channel has entered a bridge.

Expand source


```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    },
    "channel": {
      "type": "Channel"
    }
  },
  "id": "ChannelEnteredBridge",
  "description": "Notification that a channel has entered a bridge."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge

* channel: Channel *(optional)*

ChannelHangupRequest

Base type: Event

A hangup was requested on the channel.

Expand source


```
{
  "properties": {
    "soft": {
      "type": "boolean",
      "description": "Whether the hangup request was a soft hangup request."
    },
    "cause": {
      "type": "int",
      "description": "Integer representation of the cause of the hangup."
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel on which the hangup was requested."
    }
  },
  "id": "ChannelHangupRequest",
  "description": "A hangup was requested on the channel."
}
```


* type: string - Indicates the type of this message.
* application: string - Name of the application receiving the event.
* timestamp: Date *(optional)* - Time at which this event was created.
* cause: int *(optional)* - Integer representation of the cause of the hangup.
* channel: Channel - The channel on which the hangup was requested.
* soft: boolean *(optional)* - Whether the hangup request was a soft hangup request.

ChannelLeftBridge

Base type: Event

Notification that a channel has left a bridge.

Expand source


```
{
  "properties": {
    "bridge": {
      "required": true,
      "type": "Bridge"
    },
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelLeftBridge",
  "description": "Notification that a channel has left a bridge."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge
- channel: Channel

ChannelStateChange

Base type: Event

Notification of a channel's state change.

Expand source


```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "ChannelStateChange",
  "description": "Notification of a channel's state change."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel

ChannelTalkingFinished

Base type: Event

Talking is no longer detected on the channel.

Expand source


```
{
  "properties": {
    "duration": {
      "required": true,
      "type": "int",
      "description": "The length of time, in milliseconds, that talking was detected on the channel"
    },
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel on which talking completed."
    }
  },
  "id": "ChannelTalkingFinished",
  "description": "Talking is no longer detected on the channel."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel - The channel on which talking completed.
- duration: int - The length of time, in milliseconds, that talking was detected on the channel

ChannelTalkingStarted

Base type: Event

Talking was detected on the channel.

Expand source


```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel",
      "description": "The channel on which talking started."
    }
  },
  "id": "ChannelTalkingStarted",
  "description": "Talking was detected on the channel."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel - The channel on which talking started.

ChannelUserevent

Base type: Event

User-generated event with additional user-defined fields in the object.

Expand source


```
{
  "properties": {
    "eventname": {
      "required": true,
      "type": "string",
      "description": "The name of the user event."
    },
    "bridge": {
      "required": false,
      "type": "Bridge",
      "description": "A bridge that is signaled with the user event."
    },
    "userevent": {
      "required": true,
      "type": "object",
      "description": "Custom Userevent data"
    },
    "endpoint": {
      "required": false,
      "type": "Endpoint",
      "description": "A endpoint that is signaled with the user event."
    },
    "channel": {
      "required": false,
      "type": "Channel",
      "description": "A channel that is signaled with the user event."
    }
  },
  "id": "ChannelUserevent",
  "description": "User-generated event with additional user-defined fields in the
object."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- bridge: Bridge *(optional)* - A bridge that is signaled with the user event.
- channel: Channel *(optional)* - A channel that is signaled with the user event.
- endpoint: Endpoint *(optional)* - A endpoint that is signaled with the user event.
- eventname: string - The name of the user event.
- userevent: object - Custom Userevent data

ChannelVarset

Base type: Event

Channel variable changed.

Expand source


```
{
  "properties": {
    "variable": {
      "required": true,
      "type": "string",
      "description": "The variable that changed."
    },
    "channel": {
      "required": false,
      "type": "Channel",
      "description": "The channel on which the variable was set.\n\nIf missing, the
variable is a global variable."
    },
    "value": {
      "required": true,
      "type": "string",
      "description": "The new value of the variable."
    }
  },
  "id": "ChannelVarset",
  "description": "Channel variable changed."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel *(optional)* - The channel on which the variable was set.

If missing, the variable is a global variable.

- value: string - The new value of the variable.
- variable: string - The variable that changed.

DeviceStateChanged

Base type: Event

Notification that a device state has changed.

Expand source


```
{
  "properties": {
    "device_state": {
      "required": true,
      "type": "DeviceState",
      "description": "Device state object"
    }
  },
  "id": "DeviceStateChanged",
  "description": "Notification that a device state has changed."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.

- device_state: DeviceState - Device state object

Dial

Base type: Event

Dialing state has changed.

Expand source


```
{
  "properties": {
    "forwarded": {
      "required": false,
      "type": "Channel",
      "description": "Channel that the caller has been forwarded to."
    },
    "caller": {
      "required": false,
      "type": "Channel",
      "description": "The calling channel."
    },
    "dialstatus": {
      "required": true,
      "type": "string",
      "description": "Current status of the dialing attempt to the peer."
    },
    "forward": {
      "required": false,
      "type": "string",
      "description": "Forwarding target requested by the original dialed channel."
    },
    "dialstring": {
      "required": false,
      "type": "string",
      "description": "The dial string for calling the peer channel."
    },
    "peer": {
      "required": true,
      "type": "Channel",
      "description": "The dialed channel."
    }
  },
  "id": "Dial",
  "description": "Dialing state has changed."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- caller: Channel *(optional)* - The calling channel.
- dialstatus: string - Current status of the dialing attempt to the peer.
- dialstring: string *(optional)* - The dial string for calling the peer channel.
- forward: string *(optional)* - Forwarding target requested by the original dialed channel.
- forwarded: Channel *(optional)* - Channel that the caller has been forwarded to.
- peer: Channel - The dialed channel.

EndpointStateChange

Base type: Event

Endpoint state changed.

Expand

source


```
{
  "properties": {
    "endpoint": {
      "required": true,
      "type": "Endpoint"
    }
  },
  "id": "EndpointStateChange",
  "description": "Endpoint state changed."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- endpoint: Endpoint

Event

Base type: Message
Subtypes: ApplicationReplaced BridgeAttendedTransfer BridgeBlindTransfer BridgeCreated BridgeDestroyed BridgeMerged ChannelCallerId ChannelCreated ChannelDestroyed ChannelDialplan ChannelDtmfReceived ChannelEnteredBridge ChannelHangupRequest ChannelLeftBridge ChannelStateChange ChannelTalkingFinished ChannelTalkingStarted ChannelUserevent ChannelVarset DeviceStateChanged Dial EndpointStateChange PlaybackFinished PlaybackStarted RecordingFailed RecordingFinished RecordingStarted StasisEnd StasisStart TextMessageReceived

Base type for asynchronous events from Asterisk.

Expand source


```
{
  "subTypes": [
    "DeviceStateChanged",
    "PlaybackStarted",
    "PlaybackFinished",
    "RecordingStarted",
    "RecordingFinished",
    "RecordingFailed",
    "ApplicationReplaced",
    "BridgeCreated",
    "BridgeDestroyed",
    "BridgeMerged",
    "BridgeBlindTransfer",
    "BridgeAttendedTransfer",
    "ChannelCreated",
    "ChannelDestroyed",
    "ChannelEnteredBridge",
    "ChannelLeftBridge",
    "ChannelStateChange",
    "ChannelDtmfReceived",
    "ChannelDialplan",
    "ChannelCallerId",
    "ChannelUserevent",
    "ChannelHangupRequest",
    "ChannelVarset",
    "ChannelTalkingStarted",
    "ChannelTalkingFinished",
    "EndpointStateChange",
    "Dial",
    "StasisEnd",
    "StasisStart",
    "TextMessageReceived"
  ],
  "properties": {
    "application": {
      "required": true,
      "type": "string",
      "description": "Name of the application receiving the event."
    },
    "timestamp": {
      "required": false,
      "type": "Date",
      "description": "Time at which this event was created."
    }
  },
  "id": "Event",
  "description": "Base type for asynchronous events from Asterisk."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.

Message

Subtypes: ApplicationReplaced BridgeAttendedTransfer BridgeBlindTransfer BridgeCreated BridgeDestroyed BridgeMerged ChannelCallerId ChannelCreated ChannelDestroyed ChannelDialplan ChannelDtmfReceived ChannelEnteredBridge ChannelHangupRequest ChannelLeftBridge ChannelStateChange ChannelTalkingFinished ChannelTalkingStarted ChannelUserevent ChannelVarset DeviceStateChanged Dial EndpointStateChange Event MissingParams PlaybackFinished PlaybackStarted RecordingFailed RecordingFinished RecordingStarted StasisEnd StasisStart TextMessageReceived

Base type for errors and events

Expand

source


```
{
  "discriminator": "type",
  "properties": {
    "type": {
      "required": true,
      "type": "string",
      "description": "Indicates the type of this message."
    }
  },
  "subTypes": [
    "MissingParams",
    "Event"
  ],
  "id": "Message",
  "description": "Base type for errors and events"
}
```


- type: string - Indicates the type of this message.

MissingParams

Base type: Message

Error event sent when required params are missing.

Expand

source


```
{
  "properties": {
    "params": {
      "required": true,
      "type": "List[string]",
      "description": "A list of the missing parameters"
    }
  },
  "id": "MissingParams",
  "description": "Error event sent when required params are missing."
}
```


- type: string - Indicates the type of this message.
- params: List[string] - A list of the missing parameters

PlaybackFinished

Base type: Event

Event showing the completion of a media playback operation.

Expand source


```
{
  "properties": {
    "playback": {
      "required": true,
      "type": "Playback",
      "description": "Playback control object"
    }
  },
  "id": "PlaybackFinished",
  "description": "Event showing the completion of a media playback operation."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- playback: Playback - Playback control object

PlaybackStarted

Base type: Event

Event showing the start of a media playback operation.

Expand source


```
{
  "properties": {
    "playback": {
      "required": true,
      "type": "Playback",
      "description": "Playback control object"
    }
  },
  "id": "PlaybackStarted",
  "description": "Event showing the start of a media playback operation."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- playback: Playback - Playback control object

RecordingFailed

Base type: Event

Event showing failure of a recording operation.

Expand

source


```
{
  "properties": {
    "recording": {
      "required": true,
      "type": "LiveRecording",
      "description": "Recording control object"
    }
  },
  "extends": "Event",
  "id": "RecordingFailed",
  "description": "Event showing failure of a recording operation."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- recording: LiveRecording - Recording control object

RecordingFinished

Base type: Event

Event showing the completion of a recording operation.

Expand

source


```
{
  "properties": {
    "recording": {
      "required": true,
      "type": "LiveRecording",
      "description": "Recording control object"
    }
  },
  "extends": "Event",
  "id": "RecordingFinished",
  "description": "Event showing the completion of a recording operation."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- recording: LiveRecording - Recording control object

RecordingStarted

Base type: Event

Event showing the start of a recording operation.

Expand source

```
{
  "properties": {
    "recording": {
      "required": true,
      "type": "LiveRecording",
      "description": "Recording control object"
    }
  },
  "extends": "Event",
  "id": "RecordingStarted",
  "description": "Event showing the start of a recording operation."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- recording: LiveRecording - Recording control object

StasisEnd

Base type: Event

Notification that a channel has left a Stasis application.

Expand source

```
{
  "properties": {
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "StasisEnd",
  "description": "Notification that a channel has left a Stasis application."
}
```

- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- channel: Channel

StasisStart

Base type: Event

Notification that a channel has entered a Stasis application.

Expand source


```
{
  "properties": {
    "args": {
      "required": true,
      "type": "List[string]",
      "description": "Arguments to the application"
    },
    "replace_channel": {
      "required": false,
      "type": "Channel"
    },
    "channel": {
      "required": true,
      "type": "Channel"
    }
  },
  "id": "StasisStart",
  "description": "Notification that a channel has entered a Stasis application."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- args: List[string] - Arguments to the application
- channel: Channel
- replace_channel: Channel *(optional)*

TextMessageReceived

Base type: Event

A text message was received from an endpoint.

Expand source


```
{
  "properties": {
    "message": {
      "required": true,
      "type": "TextMessage"
    },
    "endpoint": {
      "required": false,
      "type": "Endpoint"
    }
  },
  "id": "TextMessageReceived",
  "description": "A text message was received from an endpoint."
}
```


- type: string - Indicates the type of this message.
- application: string - Name of the application receiving the event.
- timestamp: Date *(optional)* - Time at which this event was created.
- endpoint: Endpoint *(optional)*
- message: TextMessage

Application

Details of a Stasis application

Expand source


```
{
  "properties": {
    "endpoint_ids": {
      "required": true,
      "type": "List[string]",
      "description": "{tech}/{resource} for endpoints subscribed to."
    },
    "channel_ids": {
      "required": true,
      "type": "List[string]",
      "description": "Id's for channels subscribed to."
    },
    "bridge_ids": {
      "required": true,
      "type": "List[string]",
      "description": "Id's for bridges subscribed to."
    },
    "device_names": {
      "required": true,
      "type": "List[string]",
      "description": "Names of the devices subscribed to."
    },
    "name": {
      "required": true,
      "type": "string",
      "description": "Name of this application"
    }
  },
  "id": "Application",
  "description": "Details of a Stasis application"
}
```


- bridge_ids: List[string] - Id's for bridges subscribed to.
- channel_ids: List[string] - Id's for channels subscribed to.
- device_names: List[string] - Names of the devices subscribed to.
- endpoint_ids: List[string] - {tech}/{resource} for endpoints subscribed to.
- name: string - Name of this application

Asterisk 13 Sounds REST API

Sounds

Method	Path	Return Model	Summary
GET	/sounds	List[Sound]	List all sounds.
GET	/sounds/{soundId}	Sound	Get a sound's details.

GET /sounds

List all sounds.

Query parameters

- lang: string - Lookup sound for a specific language.
- format: string - Lookup sound in a specific format.

GET /sounds/{soundId}

Get a sound's details.

Path parameters

- soundId: string - Sound's id

Asterisk 13 Dialplan Applications

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_GetCPEID
- `adsi.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_AgentLogin

AgentLogin()

Synopsis

Login an agent.

Description

Login an agent to the system. Any agent authentication is assumed to already be done by dialplan. While logged in, the agent can receive calls and will hear the sound file specified by the config option custom_beep when a new call comes in for the agent. Login failures will continue in the dialplan with `AGENT_STATUS` set.

Before logging in, you can setup on the real agent channel the CHANNEL(dtmf-features) an agent will have when talking to a caller and you can setup on the channel running this application the CONNECTEDLINE() information the agent will see while waiting for a caller.

AGENT_STATUS enumeration values:

- `INVALID` - The specified agent is invalid.
- `ALREADY_LOGGED_IN` - The agent is already logged in.

> **Note**
> The Agents:*AgentId* device state is available to monitor the status of the agent.

Syntax

```
AgentLogin(AgentId,[options])
```

Arguments

- `AgentId`
- `options`
    - `s` - silent login - do not announce the login ok segment after agent logged on.

See Also

- Asterisk 13 Application_Authenticate
- Asterisk 13 Application_Queue
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_AGENT
- Asterisk 13 Function_CHANNEL(dtmf-features)
- Asterisk 13 Function_CONNECTEDLINE()
- `agents.conf`
- `queues.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_AgentRequest

AgentRequest()

Synopsis

Request an agent to connect with the channel.

Description

Request an agent to connect with the channel. Failure to find, alert the agent, or acknowledge the call will continue in the dialplan with `AGENT_STATUS` set.

AGENT_STATUS enumeration values:

- `INVALID` - The specified agent is invalid.
- `NOT_LOGGED_IN` - The agent is not available.
- `BUSY` - The agent is on another call.
- `NOT_CONNECTED` - The agent did not connect with the call. The agent most likely did not acknowledge the call.
- `ERROR` - Alerting the agent failed.

Syntax

```
AgentRequest(AgentId)
```

Arguments

- `AgentId`

See Also

- Asterisk 13 Application_AgentLogin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(command,arg1,[arg2[,...]])
```

Arguments

- `command`
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 13 Application_EAGI
- Asterisk 13 Application_DeadAGI

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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**
> Few Ademco DTMF signalling formats are detected automaticaly: Contact ID, Express 4+1, Express 4+2, High Speed and Super Fast.

The application is affected by the following variables:

- `ALARMRECEIVER_CALL_LIMIT` - Maximum call time, in milliseconds.
  If set, this variable causes application to exit after the specified time.
- `ALARMRECEIVER_RETRIES_LIMIT` - Maximum number of retries per call.
  If set, this variable causes application to exit after the specified number of messages.

Syntax

```
AlarmReceiver()
```

Arguments

See Also

- `alarmreceiver.conf`

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_WaitForSilence
- Asterisk 13 Application_WaitForNoise

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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])
```

Arguments

- `delay` - Asterisk will wait this number of milliseconds before returning to the dialplan after answering the call.

See Also

- Asterisk 13 Application_Hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_VMAuthenticate
- Asterisk 13 Application_DISA

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_ControlPlayback
- Asterisk 13 Application_WaitExten
- Asterisk 13 Application_BackgroundDetect
- Asterisk 13 Function_TIMEOUT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_BridgeWait

BridgeWait()

Synopsis

Put a call into the holding bridge.

Description

This application places the incoming channel into a holding bridge. The channel will then wait in the holding bridge until some event occurs which removes it from the holding bridge.

> **Note**
> This application will answer calls which haven't already been answered.

Syntax

```
BridgeWait([name,[role,[options]]])
```

Arguments

- `name` - Name of the holding bridge to join. This is a handle for `BridgeWait` only and does not affect the actual bridges that are created. If not provided, the reserved name `default` will be used.
- `role` - Defines the channel's purpose for entering the holding bridge. Values are case sensitive.
  - `participant` - The channel will enter the holding bridge to be placed on hold until it is removed from the bridge for some reason. (default)
  - `announcer` - The channel will enter the holding bridge to make announcements to channels that are currently in the holding bridge. While an announcer is present, holding for the participants will be suspended.
- `options`
  - `m` - The specified MOH class will be used/suggested for music on hold operations. This option will only be useful for entertainment modes that use it (m and h).
    - `class`
  - `e` - Which entertainment mechanism should be used while on hold in the holding bridge. Only the first letter is read.
    - `m` - Play music on hold (default)
    - `r` - Ring without pause
    - `s` - Generate silent audio
    - `h` - Put the channel on hold
    - `n` - No entertainment
  - `S` - Automatically exit the bridge and return to the PBX after **duration** seconds.
    - `duration`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Congestion
- Asterisk 13 Application_Progress
- Asterisk 13 Application_Playtones
- Asterisk 13 Application_Hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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' or 'u' options are 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.
  - `u` - The `chanprefix` parameter is a channel uniqueid or fully specified channel name.
  - `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 while actively spying on a channel. If there is no channel being spied on, the DTMF digit will be ignored.
- `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 13 Application_ExtenSpy
- Asterisk 13 ManagerEvent_ChanSpyStart
- Asterisk 13 ManagerEvent_ChanSpyStop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

This application sets the following channel variable upon completion:

- `CONFBRIDGE_RESULT`
  - FAILED - The channel encountered an error and could not enter the conference.
  - HANGUP - The channel exited the conference by hanging up.
  - KICKED - The channel was kicked from the conference.
  - ENDMARKED - The channel left the conference as a result of the last marked user leaving.
  - DTMF - The channel pressed a DTMF sequence to exit the conference.

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. When left blank, a dynamically built menu profile created by the CONFBRIDGE dialplan function is searched for on the channel and used. If no dynamic profile is present, the 'default_menu' profile found in confbridge.conf is used.

See Also

- Asterisk 13 Application_ConfBridge
- Asterisk 13 Function_CONFBRIDGE
- Asterisk 13 Function_CONFBRIDGE_INFO

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Busy
- Asterisk 13 Application_Progress
- Asterisk 13 Application_Playtones
- Asterisk 13 Application_Hangup

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_While
- Asterisk 13 Application_EndWhile
- Asterisk 13 Application_ExitWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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
  - REMOTESTOPPED
  - 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

- Asterisk 13 ManagerEvent_ChanSpyStart
- Asterisk 13 ManagerEvent_ChanSpyStop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_DB_DELETE
- Asterisk 13 Application_DBdeltree
- Asterisk 13 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_DB_DELETE
- Asterisk 13 Application_DBdel
- Asterisk 13 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(command,arg1,[arg2[,...]])
```

Arguments

- `command`
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 13 Application_AGI
- Asterisk 13 Application_EAGI

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 to the called party immediately after receiving a PROGRESS message.
  See SendDTMF for valid digits.
    - `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 `musi conhold.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** c hannel. 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 the indications.conf tonezone currently in use.
    - `R` - Default: Indicate ringing to the calling party, even if the called party isn't actually ringing. Allow interruption of the ringback if early media is received on the channel.
    - `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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

This application will set the following channel variable before completion:

- `DIRECTORY_RESULT` - Reason Directory application exited.
    - OPERATOR - User requested operator
    - ASSISTANT - User requested assistant
    - TIMEOUT - User allowed DTMF wait duration to pass without sending DTMF
    - HANGUP - The channel hung up before the application finished
    - SELECTED - User selected a user to call from the directory
    - USEREXIT - User exited with '#' during selection
    - FAILED - The application failed

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
    - a - Allow the caller to additionally enter an alias for a user in the directory. This option must be specified in addition to the `f`, `l`, or b option.
    - `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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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,[cid,mailbox@[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 13 Application_Authenticate
- Asterisk 13 Application_VMAuthenticate

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - Minimum verbose level

See Also

- Asterisk 13 Application_NoOp
- Asterisk 13 Application_Verbose

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(command,arg1,[arg2[,...]])
```

Arguments

- `command`
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 13 Application_AGI
- Asterisk 13 Application_DeadAGI

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_EndWhile

EndWhile()

Synopsis

End a while loop.

Description

Return to the previous called `While()`.

Syntax

```
EndWhile()
```

Arguments

See Also

- Asterisk 13 Application_While
- Asterisk 13 Application_ExitWhile
- Asterisk 13 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(appname(arguments))
```

Arguments

- `appname` - Application name and arguments of the dialplan application to execute.
    - `arguments`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(expression?appiftrue:[appiffalse])
```

Arguments

- `expression`
- `execapp`
    - `appiftrue`
        - `args`
    - `appiffalse`
        - `args`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(times,weekdays,mdays,months,[timezone]?appname[(appargs]))
```

Arguments

- `day_condition`
    - `times`
    - `weekdays`
    - `mdays`
    - `months`
    - `timezone`
- `appname`
    - `appargs`

See Also

- Asterisk 13 Application_Exec
- Asterisk 13 Application_ExecIf
- Asterisk 13 Application_TryExec
- Asterisk 13 Application_GotoIfTime

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_While
- Asterisk 13 Application_EndWhile
- Asterisk 13 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 while actively spying on a channel. If there is no channel being spied on, the DTMF digit will be ignored.
- `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 13 Application_ChanSpy
- Asterisk 13 ManagerEvent_ChanSpyStart
- Asterisk 13 ManagerEvent_ChanSpyStop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(command|ivr://host([arg1,[arg2[,...]]]),[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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SendDTMF

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_ForkCDR

ForkCDR()

Synopsis

Forks the current Call Data Record for this channel.

Description

Causes the Call Data Record engine to fork a new CDR starting from the time the application is executed. The forked CDR will be linked to the end of the CDRs associated with the channel.

Syntax

```
ForkCDR([options])
```

Arguments

- `options`
  - `a` - If the channel is answered, set the answer time on the forked CDR to the current time. If this option is not used, the answer time on the forked CDR will be the answer time on the original CDR. If the channel is not answered, this option has no effect.
    Note that this option is implicitly assumed if the `r` option is used.
  - `e` - End (finalize) the original CDR.
  - `r` - Reset the start and answer times on the forked CDR. This will set the start and answer times (if the channel is answered) to be set to the current time.
    Note that this option implicitly assumes the a option.
  - `v` - Do not copy CDR variables and attributes from the original CDR to the forked CDR.

See Also

- Asterisk 13 Function_CDR
- Asterisk 13 Application_NoCDR
- Asterisk 13 Application_ResetCDR

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_Gosub

Gosub()

Synopsis

Jump to label, saving return address.

Description

Jumps to the label specified, saving the return address.

Syntax

```
Gosub([context,[exten,]]priority[(arg1,[...][argN]]))
```

Arguments

- `context`
- `exten`
- `priority`
    - `arg1`
    - `argN`

See Also

- Asterisk 13 Application_GosubIf
- Asterisk 13 Application_Macro
- Asterisk 13 Application_Goto
- Asterisk 13 Application_Return
- Asterisk 13 Application_StackPop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(condition?[labeliftrue:[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 13 Application_Gosub
- Asterisk 13 Application_Return
- Asterisk 13 Application_MacroIf
- Asterisk 13 Function_IF
- Asterisk 13 Application_GotoIf
- Asterisk 13 Application_Goto

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_GotoIf
- Asterisk 13 Application_GotoIfTime
- Asterisk 13 Application_Gosub
- Asterisk 13 Application_Macro

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(condition?[labeliftrue:[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 13 Application_Goto
- Asterisk 13 Application_GotoIfTime
- Asterisk 13 Application_GosubIf
- Asterisk 13 Application_MacroIf

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(times,weekdays,mdays,months,[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 13 Application_GotoIf
- Asterisk 13 Application_Goto
- Asterisk 13 Function_IFTIME
- Asterisk 13 Function_TESTTIME

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Answer
- Asterisk 13 Application_Busy
- Asterisk 13 Application_Congestion

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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).

Syntax

See Also

- Asterisk 13 Function_HANGUPCAUSE
- Asterisk 13 Function_HANGUPCAUSE_KEYS

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(newvar=channelname,variable)
```

Arguments

- `newvar`
- `vardata`
    - `channelname`
    - `variable`

See Also

- Asterisk 13 Application_Set

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_JABBER_STATUS_res_xmpp
- Asterisk 13 Function_JABBER_RECEIVE_res_xmpp

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(name,arg1,[arg2[,...]])
```

Arguments

- `name` - The name of the macro
- `args`
    - `arg1`
    - `arg2`

See Also

- Asterisk 13 Application_MacroExit
- Asterisk 13 Application_Goto
- Asterisk 13 Application_Gosub

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Macro

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Macro

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(expr?macroiftrue:[macroiffalse])
```

Arguments

- `expr`
- `destination`
    - `macroiftrue`
        - `macroiftrue`
        - `arg1`
    - `macroiffalse`
        - `macroiffalse`
        - `arg1`

See Also

- Asterisk 13 Application_GotoIf
- Asterisk 13 Application_GosubIf
- Asterisk 13 Function_IF

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_VM_INFO

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MeetMeCount
- Asterisk 13 Application_MeetMeAdmin
- Asterisk 13 Application_MeetMeChannelAdmin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MeetMe

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MeetMe

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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: PJSIP**
      Specifying a prefix of `pjsip:` will send the message as a SIP MESSAGE request.
    - **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: PJSIP**
      The `from` parameter can be a configured endpoint or in the form of "display-name" <URI>.
    - **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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

- `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.
  - `B` - Play a periodic beep while this call is being recorded.
    - `interval` - Interval, in seconds. Default is 15.
  - `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`
  - `p` - Play a beep on the channel that starts the recording.
  - `P` - Play a beep on the channel that stops the recording.
  - `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 13 Application_Monitor
- Asterisk 13 Application_StopMixMonitor
- Asterisk 13 Application_PauseMonitor
- Asterisk 13 Application_UnpauseMonitor
- Asterisk 13 Function_AUDIOHOOK_INHERIT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.
  - `B` - Play a periodic beep while this call is being recorded.
    - `interval` - Interval, in seconds. Default is 15.
  - `i` - Skip recording of input stream (disables `m` option).
  - `o` - Skip recording of output stream (disables `m` option).

See Also

- Asterisk 13 Application_StopMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SayAlpha
- Asterisk 13 Application_SayPhonetic

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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=value1,name2=value2)
```

Arguments

- `set1`
    - `name1`
    - `value1`
- `set2`
    - `name2`
    - `value2`

See Also

- Asterisk 13 Application_Set

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_NoCDR

NoCDR()

Synopsis

Tell Asterisk to not maintain a CDR for this channel.

Description

This application will tell Asterisk not to maintain a CDR for the current channel. This does **NOT** mean that information is not tracked; rather, if the channel is hung up no CDRs will be created for that channel.

If a subsequent call to ResetCDR occurs, all non-finalized CDRs created for the channel will be enabled.

> **Note**
> This application is deprecated. Please use the CDR_PROP function to disable CDRs on a channel.

Syntax

```
NoCDR()
```

Arguments

See Also

- Asterisk 13 Application_ResetCDR
- Asterisk 13 Function_CDR_PROP

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Verbose
- Asterisk 13 Application_Log

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_OSPLookup
- Asterisk 13 Application_OSPNext
- Asterisk 13 Application_OSPFinish

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_OSPAuth
- Asterisk 13 Application_OSPLookup
- Asterisk 13 Application_OSPNext

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_OSPAuth
- Asterisk 13 Application_OSPNext
- Asterisk 13 Application_OSPFinish

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

Syntax

See Also

- Asterisk 13 Application_OSPAuth
- Asterisk 13 Application_OSPLookup
- Asterisk 13 Application_OSPFinish

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`
    - `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`
    - `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 13 Application_ConfBridge

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_Park

Park()

Synopsis

Park yourself.

Description

Used to park yourself (typically in combination with an attended 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 in use then execution will continue at the next priority.

Syntax

```
Park([parking_lot_name,[options]])
```

Arguments

- `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 to this application
  2) `PARKINGLOT` variable
  3) `CHANNEL(parkinglot)` function (Possibly preset by the channel driver.)
  4) Default parking lot.
- `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.
    - `c` - If the parking times out, go to this place in the dialplan instead of where the parking lot defines the call should go.
        - `context`
        - `extension`
        - `priority`
    - `t` - Use a timeout of `duration` seconds instead of the timeout specified by the parking lot.
        - `duration`

See Also

- Asterisk 13 Application_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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([parking_lot_name,[options,announce:[announce1[:...]],]]dial)
```

Arguments

- `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 to this application
  2) `PARKINGLOT` variable
  3) `CHANNEL(parkinglot)` function (Possibly preset by the channel driver.)
  4) Default parking lot.
- `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.
  - `c` - If the parking times out, go to this place in the dialplan instead of where the parking lot defines the call should go.
    - `context`
    - `extension`
    - `priority`
  - `t` - Use a timeout of `duration` seconds instead of the timeout specified by the parking lot.
    - `duration`
- `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`
- `dial` - The app_dial style resource to call to make the announcement. Console/dsp calls the console.

See Also

- Asterisk 13 Application_Park
- Asterisk 13 Application_ParkedCall

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_ParkedCall

ParkedCall()

Synopsis

Retrieve a parked call.

Description

Used to retrieve a parked call from a parking lot.

> **Note**
> If a parking lot's parkext option is set, then Parking lots will automatically create and manage dialplan extensions in the parking lot context. If that is the case then you will not need to manage parking extensions yourself, just include the parking context of the parking lot.

Syntax

```
ParkedCall([parking_lot_name,[parking_space]])
```

Arguments

- `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.
- `parking_space` - Parking space to retrieve a parked call from. If not provided then the first available parked call in the parking lot will be retrieved.

See Also

- Asterisk 13 Application_Park

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_UnpauseMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_PickupChan

PickupChan()

Synopsis

Pickup a ringing channel.

Description

Pickup a specified *channel* if ringing.

Syntax

```
PickupChan(channel&[channel2[&...]],[options])
```

Arguments

- `channel` - ** `channel`
    - `channel2`
      List of channel names or channel uniqueids to pickup if ringing. For example, a channel name could be `SIP/bob` or `SIP/bob-00000000` to find `SIP/bob-00000000`.
- `options`
    - p - Supplied channel names are prefixes. For example, `SIP/bob` will match `SIP/bob-00000000` and `SIP/bobby-00000000`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

- Asterisk 13 Application_Background
- Asterisk 13 Application_WaitExten
- Asterisk 13 Application_ControlPlayback
- Asterisk 13 AGICommand_stream file
- Asterisk 13 AGICommand_control stream file
- Asterisk 13 ManagerAction_ControlPlayback

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_StopPlayTones

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Zapateller

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_Proceeding

Proceeding()

Synopsis

Indicate proceeding.

Description

This application will request that a proceeding message be provided to the calling channel.

Syntax

```
Proceeding()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Busy
- Asterisk 13 Application_Congestion
- Asterisk 13 Application_Ringing
- Asterisk 13 Application_Playtones

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 called party's channel (the queue member) once the parties are connected.

- `gosub` - Will run a gosub on the called party's channel (the queue member) once the parties are connected.
- `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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_Exception

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(variable,filename&[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 13 Application_SendDTMF

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_FAXOPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.
    - `o` - Exit when 0 is pressed, setting the variable `RECORD_STATUS` to `OPERATOR` instead of `DTMF`
    - `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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_ResetCDR

ResetCDR()

Synopsis

Resets the Call Data Record.

Description

This application causes the Call Data Record to be reset. Depending on the flags passed in, this can have several effects. With no options, a reset does the following:

1. The `start` time is set to the current time.

2. If the channel is answered, the `answer` time is set to the current time.

3. All variables are wiped from the CDR. Note that this step can be prevented with the v option.

On the other hand, if the e option is specified, the effects of the NoCDR application will be lifted. CDRs will be re-enabled for this channel.

> **Note**
> The e option is deprecated. Please use the CDR_PROP function instead.

Syntax

```
ResetCDR([options])
```

Arguments

- `options`
    - v - Save the CDR variables during the reset.
    - e - Enable the CDRs for this channel only (negate effects of NoCDR).

See Also

- Asterisk 13 Application_ForkCDR
- Asterisk 13 Application_NoCDR
- Asterisk 13 Function_CDR_PROP

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Gosub
- Asterisk 13 Application_StackPop

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Busy
- Asterisk 13 Application_Congestion
- Asterisk 13 Application_Progress
- Asterisk 13 Application_Playtones

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_SayAlpha

SayAlpha()

Synopsis

Say Alpha.

Description

This application will play the sounds that correspond to the letters of the given *string*. If the channel variable `SAY_DTMF_INTERRUPT` is set to 'true' (case insensitive), then this application will react to DTMF in thesame way as `Background`.

Syntax

```
SayAlpha(string)
```

Arguments

- `string`

See Also

- Asterisk 13 Application_SayDigits
- Asterisk 13 Application_SayNumber
- Asterisk 13 Application_SayPhonetic
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_SayAlphaCase

SayAlphaCase()

Synopsis

Say Alpha.

Description

This application will play the sounds that correspond to the letters of the given *string*. Optionally, a *casetype* may be specified. This will be used for case-insensitive or case-sensitive pronunciations. If the channel variable `SAY_DTMF_INTERRUPT` is set to 'true' (case insensitive), then this application will react to DTMF in the same way as `Background`.

Syntax

```
SayAlphaCase(casetype,string)
```

Arguments

- `casetype`
    - a - Case sensitive (all) pronunciation. (Ex: SayAlphaCase(a,aBc); - lowercase a uppercase b lowercase c).
    - `l` - Case sensitive (lower) pronunciation. (Ex: SayAlphaCase(l,aBc); - lowercase a b lowercase c).
    - n - Case insensitive pronunciation. Equivalent to SayAlpha. (Ex: SayAlphaCase(n,aBc) - a b c).
    - u - Case sensitive (upper) pronunciation. (Ex: SayAlphaCase(u,aBc); - a uppercase b c).
- `string`

See Also

- Asterisk 13 Application_SayDigits
- Asterisk 13 Application_SayNumber
- Asterisk 13 Application_SayPhonetic
- Asterisk 13 Application_SayAlpha
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SayCountedNoun
- Asterisk 13 Application_SayNumber

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SayCountedAdj
- Asterisk 13 Application_SayNumber

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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. If the channel variable `SAY_DTMF_INTERRUPT` is set to 'true' (case insensitive), then this application will react to DTMF in the same way as `Background`.

Syntax

```
SayDigits(digits)
```

Arguments

- `digits`

See Also

- Asterisk 13 Application_SayAlpha
- Asterisk 13 Application_SayNumber
- Asterisk 13 Application_SayPhonetic
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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. If the channel variable `SAY_DTMF_INTERRUPT` is set to 'true' (case insensitive), then this application will react to DTMF in the same way as `Background`.

Syntax

```
SayNumber(digits,[gender])
```

Arguments

- `digits`
- `gender`

See Also

- Asterisk 13 Application_SayAlpha
- Asterisk 13 Application_SayDigits
- Asterisk 13 Application_SayPhonetic
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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*. If the channel variable `SAY_DTMF_INTERRUPT` is set to 'true' (case insensitive), then this application will react to DTMF in the same way as `Background`.

Syntax

```
SayPhonetic(string)
```

Arguments

- `string`

See Also

- Asterisk 13 Application_SayAlpha
- Asterisk 13 Application_SayDigits
- Asterisk 13 Application_SayNumber

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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. This option is automatically enabled if `SAY_DTMF_INTERRUPT` is present on the channel and set to 'true' (case insensitive)

See Also

- Asterisk 13 Function_STRFTIME
- Asterisk 13 Function_STRPTIME
- Asterisk 13 Function_IFTIME

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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, W for a one 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 13 Application_Read

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_FAXOPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SendText
- Asterisk 13 Application_SendURL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SendImage
- Asterisk 13 Application_SendURL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_SendImage
- Asterisk 13 Application_SendText

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MSet
- Asterisk 13 Function_GLOBAL
- Asterisk 13 Function_SET
- Asterisk 13 Function_ENV

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_SetAMAFlags

SetAMAFlags()

Synopsis

Set the AMA Flags.

Description

This application will set the channel's AMA Flags for billing purposes.

> **Warning**
> This application is deprecated. Please use the CHANNEL function instead.

Syntax

```
SetAMAFlags([flag])
```

Arguments

- `flag`

See Also

- Asterisk 13 Function_CDR
- Asterisk 13 Function_CHANNEL

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Return
- Asterisk 13 Application_Gosub

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_Stasis

Stasis()

Synopsis

Invoke an external Stasis application.

Description

Invoke a Stasis application.

Syntax

```
Stasis(app_name,[args])
```

Arguments

- `app_name` - Name of the application to invoke.
- `args` - Optional comma-delimited arguments for the application invocation.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MixMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_StopMusicOnHold

StopMusicOnHold()

Synopsis

Stop playing Music On Hold.

Description

Stops playing music on hold.

Syntax

```
StopMusicOnHold()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_StopPlayTones

StopPlayTones()

Synopsis

Stop playing a tone list.

Description

Stop playing a tone list, initiated by PlayTones().

Syntax

```
StopPlayTones()
```

Arguments

See Also

- Asterisk 13 Application_PlayTones

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_TestServer

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_TestClient

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 transferred. 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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(appname(arguments))
```

Arguments

- `appname`
    - `arguments`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_PauseMonitor

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Application_UserEvent

UserEvent()

Synopsis

Send an arbitrary user-defined event to parties interested in a channel (AMI users and relevant res_stasis applications).

Description

Sends an arbitrary event to interested parties, with an optional *body* representing additional arguments. The *body* may be specified as a `,` delimited list of key:value pairs.

For AMI, each additional argument will be placed on a new line in the event and 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.

For res_stasis applications, the event will be provided as a JSON blob with additional arguments appearing as keys in the object and the *eventname* under the *eventname* key.

Syntax

```
UserEvent(eventname,[body])
```

Arguments

- `eventname`
- `body`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_VoiceMailMain

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_VoiceMail

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Background
- Asterisk 13 Function_TIMEOUT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_WaitForSilence

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_WaitForNoise

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_EndWhile
- Asterisk 13 Application_ExitWhile
- Asterisk 13 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Dialplan Functions

Asterisk 13 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 13 Function_AES_ENCRYPT
- Asterisk 13 Function_BASE64_ENCODE
- Asterisk 13 Function_BASE64_DECODE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_AES_DECRYPT
- Asterisk 13 Function_BASE64_ENCODE
- Asterisk 13 Function_BASE64_DECODE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` - Deprecated. The dialplan handles any agent authentication.
  - `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)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_AST_SORCERY

AST_SORCERY()

Synopsis

Get a field from a sorcery object

Description

Syntax

```
AST_SORCERY(module_name,object_type,object_id,field_name[,retrieval_method[,retrieval_details]])
```

Arguments

- `module_name` - The name of the module owning the sorcery instance.
- `object_type` - The type of object to query.
- `object_id` - The id of the object to query.
- `field_name` - The name of the field.
- `retrieval_method` - Fields that have multiple occurrences may be retrieved in two ways.
    - `concat` - Returns all matching fields concatenated in a single string separated by *separator* which defaults to `,`.
    - `single` - Returns the nth occurrence of the field as specified by *occurrence_number* which defaults to `1`.
      The default is `concat` with separator `,`.
- `retrieval_details` - Specifies either the separator for `concat` or the occurrence number for `single`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_AUDIOHOOK_INHERIT

AUDIOHOOK_INHERIT()

Synopsis

DEPRECATED: Used to set whether an audiohook may be inherited to another channel. Due to architectural changes in Asterisk 12, audiohook inheritance is performed automatically and this function now lacks function.

Description

Prior to Asterisk 12, masquerades would occur under all sorts of situations which were hard to predict. In Asterisk 12, masquerades only occur as a result of a small set of operations for which inheriting all audiohooks from the original channel is now safe. So in Asterisk 12.5+, all audiohooks are inherited without needing other controls expressing which audiohooks should be inherited under which conditions.

Syntax

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_BASE64_ENCODE
- Asterisk 13 Function_AES_DECRYPT
- Asterisk 13 Function_AES_ENCRYPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_BASE64_DECODE
- Asterisk 13 Function_AES_DECRYPT
- Asterisk 13 Function_AES_ENCRYPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CALENDAR_EVENT
- Asterisk 13 Function_CALENDAR_QUERY
- Asterisk 13 Function_CALENDAR_QUERY_RESULT
- Asterisk 13 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CALENDAR_BUSY
- Asterisk 13 Function_CALENDAR_QUERY
- Asterisk 13 Function_CALENDAR_QUERY_RESULT
- Asterisk 13 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CALENDAR_BUSY
- Asterisk 13 Function_CALENDAR_EVENT
- Asterisk 13 Function_CALENDAR_QUERY_RESULT
- Asterisk 13 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CALENDAR_BUSY
- Asterisk 13 Function_CALENDAR_EVENT
- Asterisk 13 Function_CALENDAR_QUERY
- Asterisk 13 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CALENDAR_BUSY
- Asterisk 13 Function_CALENDAR_EVENT
- Asterisk 13 Function_CALENDAR_QUERY
- Asterisk 13 Function_CALENDAR_QUERY_RESULT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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**
> 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.

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` - The final state of the CDR.
        - `0` - NO ANSWER
        - `1` - NO ANSWER (NULL record)
        - `2` - FAILED
        - `4` - BUSY
        - `8` - ANSWERED
        - `16` - CONGESTION
    - `src` - Source.
    - `start` - Time the call started.
    - `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

        > **Warning**
        > Accessing this setting is deprecated in CDR. Please use the CHANNEL function instead.
    - `dst` - Destination.
    - `answer` - Time the call was answered.
    - `accountcode` - The channel's account code.

    > **Warning**
    > Accessing this setting is deprecated in CDR. Please use the CHANNEL function instead.
    - `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.

* 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_CDR_PROP

CDR_PROP()

Synopsis

Set a property on a channel's CDR.

Description

This function sets a property on a channel's CDR. Properties alter the behavior of how the CDR operates for that channel.

Syntax

```
CDR_PROP(name)
```

Arguments

- `name` - The property to set on the CDR.
    - `party_a` - Set this channel as the preferred Party A when channels are associated together.
      Write-Only
    - `disable` - Disable CDRs for this channel.
      Write-Only

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 *item* 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.
    - `dtmf_features` - R/W The channel's DTMF bridge features. May include one or more of 'T' 'K' 'H' 'W' and 'X' in a similar manner to options in the `Dial` application. When setting it, the features string must be all upper case.
    - `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)
    - `after_bridge_goto` - R/W the parseable goto string indicating where the channel is expected to return to in the PBX after exiting the next bridge it joins on the condition that it doesn't hang up. The parseable goto string uses the same syntax as the `Goto` application.
    - `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.
- `recvport` - R/O Get the source port 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.
- **Technology: PJSIP**
  - `rtp` - R/O Retrieve media related information.
    - `type` - When *rtp* is specified, the `type` parameter must be provided. It specifies which RTP parameter to read.
      - `src` - Retrieve the local address for RTP.
      - `dest` - Retrieve the remote address for RTP.
      - `direct` - If direct media is enabled, this address is the remote address used for RTP.
      - `secure` - Whether or not the media stream is encrypted.
        - `0` - The media stream is not encrypted.
        - `1` - The media stream is encrypted.
      - `hold` - Whether or not the media stream is currently restricted due to a call hold.
        - `0` - The media stream is not held.
        - `1` - The media stream is held.
    - `media_type` - When *rtp* is specified, the `media_type` parameter may be provided. It specifies which media stream the chosen RTP parameter should be retrieved from.
      - `audio` - Retrieve information from the audio media stream.

> **Note**
> If not specified, `audio` is used by default.

    - `video` - Retrieve information from the video media stream.
- `rtcp` - R/O Retrieve RTCP statistics.
  - `statistic` - When *rtcp* is specified, the `statistic` parameter must be provided. It specifies which RTCP statistic parameter to read.
    - `all` - Retrieve a summary of all RTCP statistics.
      The following data items are returned in a semi-colon delineated list:
      - `ssrc` - Our Synchronization Source identifier
      - `themssrc` - Their Synchronization Source identifier
      - `lp` - Our lost packet count
      - `rxjitter` - Received packet jitter
      - `rxcount` - Received packet count
      - `txjitter` - Transmitted packet jitter
      - `txcount` - Transmitted packet count
      - `rlp` - Remote lost packet count
      - `rtt` - Round trip time
    - `all_jitter` - Retrieve a summary of all RTCP Jitter statistics.
      The following data items are returned in a semi-colon delineated list:
      - `minrxjitter` - Our minimum jitter
      - `maxrxjitter` - Our max jitter
      - `avgrxjitter` - Our average jitter
      - `stdevrxjitter` - Our jitter standard deviation
      - `reported_minjitter` - Their minimum jitter
      - `reported_maxjitter` - Their max jitter
      - `reported_avgjitter` - Their average jitter
      - `reported_stdevjitter` - Their jitter standard deviation
    - `all_loss` - Retrieve a summary of all RTCP packet loss statistics.
      The following data items are returned in a semi-colon delineated list:
      - `minrxlost` - Our minimum lost packets
      - `maxrxlost` - Our max lost packets
      - `avgrxlost` - Our average lost packets
      - `stdevrxlost` - Our lost packets standard deviation
      - `reported_minlost` - Their minimum lost packets
      - `reported_maxlost` - Their max lost packets
      - `reported_avglost` - Their average lost packets
      - `reported_stdevlost` - Their lost packets standard deviation
    - `all_rtt` - Retrieve a summary of all RTCP round trip time information.
      The following data items are returned in a semi-colon delineated list:
      - `minrtt` - Minimum round trip time
      - `maxrtt` - Maximum round trip time
      - `avgrtt` - Average round trip time
      - `stdevrtt` - Standard deviation round trip time
    - `txcount` - Transmitted packet count
    - `rxcount` - Received packet count
    - `txjitter` - Transmitted packet jitter
    - `rxjitter` - Received packet jitter
    - `remote_maxjitter` - Their max jitter
    - `remote_minjitter` - Their minimum jitter
    - `remote_normdevjitter` - Their average jitter
    - `remote_stdevjitter` - Their jitter standard deviation
    - `local_maxjitter` - Our max jitter
    - `local_minjitter` - Our minimum jitter
    - `local_normdevjitter` - Our average jitter
    - `local_stdevjitter` - Our jitter standard deviation
    - `txploss` - Transmitted packet loss
    - `rxploss` - Received packet loss
    - `remote_maxrxploss` - Their max lost packets
    - `remote_minrxploss` - Their minimum lost packets
    - `remote_normdevrxploss` - Their average lost packets
    - `remote_stdevrxploss` - Their lost packets standard deviation
    - `local_maxrxploss` - Our max lost packets
    - `local_minrxploss` - Our minimum lost packets
    - `local_normdevrxploss` - Our average lost packets
    - `local_stdevrxploss` - Our lost packets standard deviation
    - `rtt` - Round trip time

            - `maxrtt` - Maximum round trip time
            - `minrtt` - Minimum round trip time
            - `normdevrtt` - Average round trip time
            - `stdevrtt` - Standard deviation round trip time
            - `local_ssrc` - Our Synchronization Source identifier
            - `remote_ssrc` - Their Synchronization Source identifier
        - `media_type` - When *rtcp* is specified, the `media_type` parameter may be provided. It specifies which media stream the chosen RTCP parameter should be retrieved from.
            - `audio` - Retrieve information from the audio media stream.

              > **Note**
              > If not specified, `audio` is used by default.

            - `video` - Retrieve information from the video media stream.
    - `endpoint` - R/O The name of the endpoint associated with this channel. Use the *PJSIP_ENDPOINT* function to obtain further endpoint related information.
    - `pjsip` - R/O Obtain information about the current PJSIP channel and its session.
        - `type` - When *pjsip* is specified, the `type` parameter must be provided. It specifies which signalling parameter to read.
            - `secure` - Whether or not the signalling uses a secure transport.
                - `0` - The signalling uses a non-secure transport.
                - `1` - The signalling uses a secure transport.
            - `target_uri` - The request URI of the `INVITE` request associated with the creation of this channel.
            - `local_uri` - The local URI.
            - `remote_uri` - The remote URI.
            - `t38state` - The current state of any T.38 fax on this channel.
                - `DISABLED` - T.38 faxing is disabled on this channel.
                - `LOCAL_REINVITE` - Asterisk has sent a `re-INVITE` to the remote end to initiate a T.38 fax.
                - `REMOTE_REINVITE` - The remote end has sent a `re-INVITE` to Asterisk to initiate a T.38 fax.
                - `ENABLED` - A T.38 fax session has been enabled.
                - `REJECTED` - A T.38 fax session was attempted but was rejected.
            - `local_addr` - On inbound calls, the full IP address and port number that the `INVITE` request was received on. On outbound calls, the full IP address and port number that the `INVITE` request was transmitted from.
            - `remote_addr` - On inbound calls, the full IP address and port number that the `INVITE` request was received from. On outbound calls, the full IP address and port number that the `INVITE` request was transmitted to.
              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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_CONFBRIDGE

CONFBRIDGE()

Synopsis

Set a custom dynamic bridge, user, or menu 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`, `user`, or `menu`.
- `option` - Option refers to `confbridge.conf` option that is being set dynamically on this channel, or `clear` to remove already applied options from the channel.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CURLOPT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_CURL
- Asterisk 13 Function_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 &)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_DBdel
- Asterisk 13 Function_DB_DELETE
- Asterisk 13 Application_DBdeltree
- Asterisk 13 Function_DB_EXISTS

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to no, this function can only be read from the dialplan, and not directly from external protocols. It can, however, be executed as a write operation (`DB_DELETE(family, key)=ignored`)

Syntax

```
DB_DELETE(family/key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 13 Application_DBdel
- Asterisk 13 Function_DB
- Asterisk 13 Application_DBdeltree

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_DB

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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)

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_RaiseException

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_EXISTS

EXISTS()

Synopsis

Test the existence of a value.

Description

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

Syntax

```
EXISTS(data)
```

Arguments

- `data`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_ReceiveFax
- Asterisk 13 Application_SendFax

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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:
  - `inherit` - Inherit feature settings made in FEATURE or FEATUREMAP to child channels.
  - `featuredigittimeout` - Milliseconds allowed between digit presses when entering a feature code.
  - `transferdigittimeout` - Seconds allowed between digit presses when dialing a transfer destination
  - `atxfernoanswertimeout` - Seconds to wait for attended transfer destination to answer
  - `atxferdropcall` - Hang up the call entirely if the attended transfer fails
  - `atxferloopdelay` - Seconds to wait between attempts to re-dial transfer destination
  - `atxfercallbackretries` - Number of times to re-attempt dialing a transfer destination
  - `xfersound` - Sound to play to during transfer and transfer-like operations.
  - `xferfailsound` - Sound to play to a transferee when a transfer fails
  - `atxferabort` - Digits to dial to abort an attended transfer attempt
  - `atxfercomplete` - Digits to dial to complete an attended transfer
  - `atxferthreeway` - Digits to dial to change an attended transfer into a three-way call
  - `pickupexten` - Digits used for picking up ringing calls
  - `pickupsound` - Sound to play to picker when a call is picked up
  - `pickupfailsound` - Sound to play to picker when a call cannot be picked up
  - `courtesytone` - Sound to play when automon or automixmon is activated
  - `recordingfailsound` - Sound to play when automon or automixmon is attempted but fails to start

See Also

- Asterisk 13 Function_FEATUREMAP

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_FEATURE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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)

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

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 13 Function_FILE_COUNT_LINE
- Asterisk 13 Function_FILE_FORMAT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

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 13 Function_FILE
- Asterisk 13 Function_FILE_FORMAT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

Note
If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
FILE_FORMAT(filename)
```

Arguments

- `filename`

See Also

- Asterisk 13 Function_FILE
- Asterisk 13 Function_FILE_COUNT_LINE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_HANGUPCAUSE_KEYS
- Asterisk 13 Application_HangupCauseClear

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

Syntax

See Also

- Asterisk 13 Function_HANGUPCAUSE
- Asterisk 13 Application_HangupCauseClear

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_SIPPEER

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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*

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_IMPORT

IMPORT()

Synopsis

Retrieve the value of a variable from another channel.

Description

Syntax

```
IMPORT(channel,variable)
```

Arguments

- `channel`
- `variable`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_ISNULL

ISNULL()

Synopsis

Check if a value is NULL.

Description

Returns `1` if NULL or `0` otherwise.

Syntax

```
ISNULL(data)
```

Arguments

- `data`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_JABBER_STATUS_res_xmpp
- Asterisk 13 Application_JabberSend_res_xmpp

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_JABBER_RECEIVE_res_xmpp
- Asterisk 13 Application_JabberSend_res_xmpp

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

Jitterbuffers are constructed in two different ways. The first always take three arguments: *max_size*, *resync_threshold*, and *target_extra*. Alternatively, a single argument of `default` can be provided, which will construct the default jitterbuffer for the given *jitterbuffer type*.

The arguments are:

max_size: Length in milliseconds of the buffer. Defaults to 200 ms.

resync_threshold: The length in milliseconds over which a timestamp difference will result in resyncing the jitterbuffer. Defaults to 1000ms.

target_extra: This option only affects the adaptive jitterbuffer. It represents the amount time in milliseconds by which the new jitter buffer will pad its size. Defaults to 40ms.

Example: Fixed with defaults

```
exten => 1,1,Set(JITTERBUFFER(fixed)=default)
```

Example: Fixed with 200ms max size

```
exten => 1,1,Set(JITTERBUFFER(fixed)=200)
```

Example: Fixed with 200ms max size, resync threshold 1500

```
exten => 1,1,Set(JITTERBUFFER(fixed)=200,1500)
```

Example: Adaptive with defaults

```
exten => 1,1,Set(JITTERBUFFER(adaptive)=default)
```

Example: Adaptive with 200ms max size, 60ms target extra

```
exten => 1,1,Set(JITTERBUFFER(adaptive)=200,,60)
```


Example: Set a fixed jitterbuffer with defaults; then remove it

```
exten => 1,1,Set(JITTERBUFFER(fixed)=default)
exten => 1,n,Set(JITTERBUFFER(disabled)=)
```

> **Note**
> If a channel specifies a jitterbuffer due to channel driver configuration and the JITTERBUFFER function has set a jitterbuffer for that channel, the jitterbuffer set by the JITTERBUFFER function will take priority and the jitterbuffer set by the channel configuration will not be applied.

Syntax

```
JITTERBUFFER(jitterbuffer type)
```

Arguments

- `jitterbuffer type`
    - `fixed` - Set a fixed jitterbuffer on the channel.
    - `adaptive` - Set an adaptive jitterbuffer on the channel.
    - `disabled` - Remove a previously set jitterbuffer from the channel.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Function_KEYPADHASH

KEYPADHASH()

Synopsis

Hash the letters in string into equivalent keypad numbers.

Description

Example: ${KEYPADHASH(Les)} returns "537"

Syntax

```
KEYPADHASH(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_LEN

LEN()

Synopsis

Return the length of the string given.

Description

Example: ${LEN(example)} returns 7

Syntax

```
LEN(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Gosub
- Asterisk 13 Application_GosubIf
- Asterisk 13 Application_Return

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Gosub
- Asterisk 13 Application_GosubIf
- Asterisk 13 Application_Return

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
LOCK(lockname)
```

Arguments

- `lockname`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_VM_INFO

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

Syntax

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_MD5

MD5()

Synopsis

Computes an MD5 digest.

Description

Computes an MD5 digest.

Syntax

```
MD5(data)
```

Arguments

- `data`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MeetMe
- Asterisk 13 Application_MeetMeCount
- Asterisk 13 Application_MeetMeAdmin
- Asterisk 13 Application_MeetMeChannelAdmin

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MessageSend

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MessageSend

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MinivmRecord
- Asterisk 13 Application_MinivmGreet
- Asterisk 13 Application_MinivmNotify
- Asterisk 13 Application_MinivmDelete
- Asterisk 13 Application_MinivmAccMess
- Asterisk 13 Application_MinivmMWI
- Asterisk 13 Function_MINIVMCOUNTER

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_MinivmRecord
- Asterisk 13 Application_MinivmGreet
- Asterisk 13 Application_MinivmNotify
- Asterisk 13 Application_MinivmDelete
- Asterisk 13 Application_MinivmAccMess
- Asterisk 13 Application_MinivmMWI
- Asterisk 13 Function_MINIVMACCOUNT

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_MIXMONITOR

MIXMONITOR()

Synopsis

Retrieve data pertaining to specific instances of MixMonitor on a channel.

Description

Syntax

```
MIXMONITOR(id,key)
```

Arguments

- `id` - The unique ID of the MixMonitor instance. The unique ID can be retrieved through the channel variable used as an argument to the *i* option to MixMonitor.
- `key` - The piece of data to retrieve from the MixMonitor.
    - `filename`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_PERIODIC_HOOK

PERIODIC_HOOK()

Synopsis

Execute a periodic dialplan hook into the audio of a call.

Description

For example, you could use this function to enable playing a periodic `beep` sound in a call.

To turn on:

Set(BEEPID=${PERIODIC_HOOK(hooks,beep,180)})

To turn off:

Set(PERIODIC_HOOK(${BEEPID})=off)

To turn back on again later:

Set(PERIODIC_HOOK(${BEEPID})=on)

It is important to note that the hook does not actually run on the channel itself. It runs asynchronously on a new channel. Any audio generated by the hook gets injected into the call for the channel PERIODIC_HOOK() was set on.

The hook dialplan will have two variables available. `HOOK_CHANNEL` is the channel the hook is enabled on. `HOOK_ID` is the hook ID for enabling or disabling the hook.

Syntax

```
PERIODIC_HOOK(context,extension,interval,hook_id)
```

Arguments

- `context` - (On Read Only) Context for the hook extension.
- `extension` - (On Read Only) The hook extension.
- `interval` - (On Read Only) Number of seconds in between hook runs. Whole seconds only.
- `hook_id` - (On Write Only) The hook ID.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_PJSIP_DIAL_CONTACTS

PJSIP_DIAL_CONTACTS()

Synopsis

Return a dial string for dialing all contacts on an AOR.

Description

Returns a properly formatted dial string for dialing all contacts on an AOR.

Syntax

```
PJSIP_DIAL_CONTACTS(endpoint[,aor[,request_user]])
```

Arguments

- `endpoint` - Name of the endpoint
- `aor` - Name of an AOR to use, if not specified the configured AORs on the endpoint are used
- `request_user` - Optional request user to use in the request URI

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_PJSIP_ENDPOINT

PJSIP_ENDPOINT()

Synopsis

Get information about a PJSIP endpoint

Description

Syntax

```
PJSIP_ENDPOINT(name,field)
```

Arguments

- `name` - The name of the endpoint to query.
- `field` - The configuration option for the endpoint to query for. Supported options are those fields on the *endpoint* object in `pjsip.conf`.
    - `100rel` - Allow support for RFC3262 provisional ACK tags
    - `aggregate_mwi` - Condense MWI notifications into a single NOTIFY.
    - `allow` - Media Codec(s) to allow
    - `aors` - AoR(s) to be used with the endpoint
    - `auth` - Authentication Object(s) associated with the endpoint
    - `callerid` - CallerID information for the endpoint
    - `callerid_privacy` - Default privacy level
    - `callerid_tag` - Internal id_tag for the endpoint
    - `context` - Dialplan context for inbound sessions
    - `direct_media_glare_mitigation` - Mitigation of direct media (re)INVITE glare
    - `direct_media_method` - Direct Media method type
    - `connected_line_method` - Connected line method type
    - `direct_media` - Determines whether media may flow directly between endpoints.
    - `disable_direct_media_on_nat` - Disable direct media session refreshes when NAT obstructs the media session
    - `disallow` - Media Codec(s) to disallow
    - `dtmf_mode` - DTMF mode
    - `media_address` - IP address used in SDP for media handling
    - `force_rport` - Force use of return port
    - `ice_support` - Enable the ICE mechanism to help traverse NAT
    - `identify_by` - Way(s) for Endpoint to be identified
    - `redirect_method` - How redirects received from an endpoint are handled
    - `mailboxes` - NOTIFY the endpoint when state changes for any of the specified mailboxes
    - `moh_suggest` - Default Music On Hold class
    - `outbound_auth` - Authentication object used for outbound requests
    - `outbound_proxy` - Proxy through which to send requests, a full SIP URI must be provided
    - `rewrite_contact` - Allow Contact header to be rewritten with the source IP address-port
    - `rtp_ipv6` - Allow use of IPv6 for RTP traffic
    - `rtp_symmetric` - Enforce that RTP must be symmetric
    - `send_diversion` - Send the Diversion header, conveying the diversion information to the called user agent
    - `send_pai` - Send the P-Asserted-Identity header
    - `send_rpid` - Send the Remote-Party-ID header
    - `timers_min_se` - Minimum session timers expiration period
    - `timers` - Session timers for SIP packets
    - `timers_sess_expires` - Maximum session timer expiration period
    - `transport` - Desired transport configuration
    - `trust_id_inbound` - Accept identification information received from this endpoint
    - `trust_id_outbound` - Send private identification details to the endpoint.
    - `type` - Must be of type 'endpoint'.
    - `use_ptime` - Use Endpoint's requested packetisation interval
    - `use_avpf` - Determines whether res_pjsip will use and enforce usage of AVPF for this endpoint.
    - `force_avp` - Determines whether res_pjsip will use and enforce usage of AVP, regardless of the RTP profile in use for this endpoint.
    - `media_use_received_transport` - Determines whether res_pjsip will use the media transport received in the offer SDP in the corresponding answer SDP.
    - `media_encryption` - Determines whether res_pjsip will use and enforce usage of media encryption for this endpoint.
    - `inband_progress` - Determines whether chan_pjsip will indicate ringing using inband progress.
    - `call_group` - The numeric pickup groups for a channel.
    - `pickup_group` - The numeric pickup groups that a channel can pickup.

- named_call_group - The named pickup groups for a channel.
- named_pickup_group - The named pickup groups that a channel can pickup.
- device_state_busy_at - The number of in-use channels which will cause busy to be returned as device state
- t38_udptl - Whether T.38 UDPTL support is enabled or not
- t38_udptl_ec - T.38 UDPTL error correction method
- t38_udptl_maxdatagram - T.38 UDPTL maximum datagram size
- fax_detect - Whether CNG tone detection is enabled
- t38_udptl_nat - Whether NAT support is enabled on UDPTL sessions
- t38_udptl_ipv6 - Whether IPv6 is used for UDPTL Sessions
- tone_zone - Set which country's indications to use for channels created for this endpoint.
- language - Set the default language to use for channels created for this endpoint.
- one_touch_recording - Determines whether one-touch recording is allowed for this endpoint.
- record_on_feature - The feature to enact when one-touch recording is turned on.
- record_off_feature - The feature to enact when one-touch recording is turned off.
- rtp_engine - Name of the RTP engine to use for channels created for this endpoint
- allow_transfer - Determines whether SIP REFER transfers are allowed for this endpoint
- sdp_owner - String placed as the username portion of an SDP origin (o=) line.
- sdp_session - String used for the SDP session (s=) line.
- tos_audio - DSCP TOS bits for audio streams
- tos_video - DSCP TOS bits for video streams
- cos_audio - Priority for audio streams
- cos_video - Priority for video streams
- allow_subscribe - Determines if endpoint is allowed to initiate subscriptions with Asterisk.
- sub_min_expiry - The minimum allowed expiry time for subscriptions initiated by the endpoint.
- from_user - Username to use in From header for requests to this endpoint.
- mwi_from_user - Username to use in From header for unsolicited MWI NOTIFYs to this endpoint.
- from_domain - Domain to user in From header for requests to this endpoint.
- dtls_verify - Verify that the provided peer certificate is valid
- dtls_rekey - Interval at which to renegotiate the TLS session and rekey the SRTP session
- dtls_cert_file - Path to certificate file to present to peer
- dtls_private_key - Path to private key for certificate file
- dtls_cipher - Cipher to use for DTLS negotiation
- dtls_ca_file - Path to certificate authority certificate
- dtls_ca_path - Path to a directory containing certificate authority certificates
- dtls_setup - Whether we are willing to accept connections, connect to the other party, or both.
- srtp_tag_32 - Determines whether 32 byte tags should be used instead of 80 byte tags.
- set_var - Variable set on a channel involving the endpoint.
- message_context - Context to route incoming MESSAGE requests to.
- accountcode - An accountcode to set automatically on any channels created for this endpoint.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_PJSIP_HEADER

PJSIP_HEADER()

Synopsis

Gets, adds, updates or removes the specified SIP header from a PJSIP session.

Description

Examples:

;

; Set 'somevar' to the value of the 'From' header.

exten => 1,1,Set(somevar=${PJSIP_HEADER(read,From)})

;

; Set 'via2' to the value of the 2nd 'Via' header.

exten => 1,1,Set(via2=${PJSIP_HEADER(read,Via,2)})

;

; Add an 'X-Myheader' header with the value of 'myvalue'.

exten => 1,1,Set(PJSIP_HEADER(add,X-MyHeader)=myvalue)

;

; Add an 'X-Myheader' header with an empty value.

exten => 1,1,Set(PJSIP_HEADER(add,X-MyHeader)=)

;

; Update the value of the header named 'X-Myheader' to 'newvalue'.

; 'X-Myheader' must already exist or the call will fail.

exten => 1,1,Set(PJSIP_HEADER(update,X-MyHeader)=newvalue)

;

; Remove all headers whose names exactly match 'X-MyHeader'.

exten => 1,1,Set(PJSIP_HEADER(remove,X-MyHeader)=)

;

; Remove all headers that begin with 'X-My'.

exten => 1,1,Set(PJSIP_HEADER(remove,X-My*)=)

;

; Remove all previously added headers.

exten => 1,1,Set(PJSIP_HEADER(remove,*)=)

;

> **Note**
> The `remove` action can be called by reading **or** writing PJSIP_HEADER.
>
> ;
>
> ; Display the number of headers removed
>
> exten => 1,1,Verbose(Removed ${PJSIP_HEADER(remove,X-MyHeader)} headers)
>
> ;
>
> ; Set a variable to the number of headers removed
>
> exten => 1,1,Set(count=${PJSIP_HEADER(remove,X-MyHeader)})


```
;
; Just remove them ignoring any count
exten => 1,1,Set(=${PJSIP_HEADER(remove,X-MyHeader)})
exten => 1,1,Set(PJSIP_HEADER(remove,X-MyHeader)=)
;
```

Note
If you call PJSIP_HEADER in a normal dialplan context you'll be operating on the **caller's (incoming)** channel which may not be what you want. To operate on the **callee's (outgoing)** channel call PJSIP_HEADER in a pre-dial handler.

Example:

```
;
[handler]
exten => addheader,1,Set(PJSIP_HEADER(add,X-MyHeader)=myvalue)
exten => addheader,2,Set(PJSIP_HEADER(add,X-MyHeader2)=myvalue2)
;
[somecontext]
exten => 1,1,Dial(PJSIP/${EXTEN},,b(handler^addheader^1))
;
```

Syntax

```
PJSIP_HEADER(action,name[,number])
```

Arguments

- `action`
  - `read` - Returns instance *number* of header *name*.
  - `add` - Adds a new header *name* to this session.
  - `update` - Updates instance *number* of header *name* to a new value. The header must already exist.
  - `remove` - Removes all instances of previously added headers whose names match *name*. A {**} may be appended to *name* to remove all headers *beginning with** *name*. *name* may be set to a single {**} to clear *all** previously added headers. In all cases, the number of headers actually removed is returned.
- `name` - The name of the header.
- `number` - If there's more than 1 header with the same name, this specifies which header to read or update. If not specified, defaults to `1` meaning the first matching header. Not valid for `add` or `remove`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_PJSIP_MEDIA_OFFER

PJSIP_MEDIA_OFFER()

Synopsis

Media and codec offerings to be set on an outbound SIP channel prior to dialing.

Description

Returns the codecs offered based upon the media choice

Syntax

```
PJSIP_MEDIA_OFFER(media)
```

Arguments

- `media` - types of media offered

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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. The values will be stored encoded within Asterisk, but all consumers of the presence state (e.g. the SIP presence event package) will receive decoded values.
    On Read - Retrieves unencoded message/subtype in Base64 encoded form.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_Queue
- Asterisk 13 Application_QueueLog
- Asterisk 13 Application_AddQueueMember
- Asterisk 13 Application_RemoveQueueMember
- Asterisk 13 Application_PauseQueueMember
- Asterisk 13 Application_UnpauseQueueMember
- Asterisk 13 Function_QUEUE_VARIABLES
- Asterisk 13 Function_QUEUE_MEMBER
- Asterisk 13 Function_QUEUE_MEMBER_COUNT
- Asterisk 13 Function_QUEUE_EXISTS
- Asterisk 13 Function_QUEUE_WAITING_COUNT
- Asterisk 13 Function_QUEUE_MEMBER_LIST
- Asterisk 13 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_QUOTE

QUOTE()

Synopsis

Quotes a given string, escaping embedded quotes as necessary

Description

Example: ${QUOTE(ab"c"de)} will return ""ab\"c\"de""

Syntax

```
QUOTE(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_REALTIME_STORE
- Asterisk 13 Function_REALTIME_DESTROY
- Asterisk 13 Function_REALTIME_FIELD
- Asterisk 13 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be read from the dialplan, and not directly from external protocols. It can, however, be executed as a write operation (`REALTIME_DESTROY(family, fieldmatch)=ignored`)

Syntax

```
REALTIME_DESTROY(family,fieldmatch,matchvalue,delim1,delim2)
```

Arguments

- `family`
- `fieldmatch`
- `matchvalue`
- `delim1`
- `delim2`

See Also

- Asterisk 13 Function_REALTIME
- Asterisk 13 Function_REALTIME_STORE
- Asterisk 13 Function_REALTIME_FIELD
- Asterisk 13 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_REALTIME
- Asterisk 13 Function_REALTIME_STORE
- Asterisk 13 Function_REALTIME_DESTROY
- Asterisk 13 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_REALTIME
- Asterisk 13 Function_REALTIME_STORE
- Asterisk 13 Function_REALTIME_DESTROY
- Asterisk 13 Function_REALTIME_FIELD

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_REALTIME
- Asterisk 13 Function_REALTIME_DESTROY
- Asterisk 13 Function_REALTIME_FIELD
- Asterisk 13 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_SET

SET()

Synopsis

SET assigns a value to a channel variable.

Description

Syntax

```
SET(varname=value)
```

Arguments

- `varname`
- `value`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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` .

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.
>
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
SHELL(command)
```

Arguments

- `command` - The command that the shell should execute.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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).

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_SMDI_MSG_RETRIEVE

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Function_SMDI_MSG

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_SPEECH_ENGINE

SPEECH_ENGINE()

Synopsis

Get or change a speech engine specific attribute.

Description

Changes a speech engine specific attribute.

Syntax

```
SPEECH_ENGINE(name)
```

Arguments

- `name`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_STAT

STAT()

Synopsis

Does a check on the specified file.

Description

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_TALK_DETECT

TALK_DETECT()

Synopsis

Raises notifications when Asterisk detects silence or talking on a channel.

Description

The TALK_DETECT function enables events on the channel it is applied to. These events can be emited over AMI, ARI, and potentially other Asterisk modules that listen for the internal notification.

The function has two parameters that can optionally be passed when `set` on a channel: *dsp_talking_threshold* and *dsp_silence_threshold*.

dsp_talking_threshold is the time in milliseconds of sound above what the dsp has established as base line silence for a user before a user is considered to be talking. By default, the value of silencethreshold from dsp.conf is used. If this value is set too tight events may be falsely triggered by variants in room noise.

Valid values are 1 through 2^31.

dsp_silence_threshold is the time in milliseconds of sound falling within what the dsp has established as baseline silence before a user is considered be silent. If this value is set too low events indicating the user has stopped talking may get falsely sent out when the user briefly pauses during mid sentence.

The best way to approach this option is to set it slightly above the maximum amount of ms of silence a user may generate during natural speech.

By default this value is 2500ms. Valid values are 1 through 2^31.

Example:

same => n,Set(TALK_DETECT(set)=) ; Enable talk detection

same => n,Set(TALK_DETECT(set)=1200) ; Update existing talk detection's silence threshold to 1200 ms

same => n,Set(TALK_DETECT(remove)=) ; Remove talk detection

same => n,Set(TALK_DETECT(set)=,128) ; Enable and set talk threshold to 128

This function will set the following variables:

> **Note**
> The TALK_DETECT function uses an audiohook to inspect the voice media frames on a channel. Other functions, such as JITTERBUFFER, DENOISE, and AGC use a similar mechanism. Audiohooks are processed in the order in which they are placed on the channel. As such, it typically makes sense to place functions that modify the voice media data prior to placing the TALK_DETECT function, as this will yield better results.
>
> Example:
>
> same => n,Set(DENOISE(rx)=on) ; Denoise received audio
>
> same => n,Set(TALK_DETECT(set)=) ; Perform talk detection on the denoised received audio

Syntax

```
TALK_DETECT(action)
```

Arguments

- `action`
    - `remove` - W/O. Remove talk detection from the channel.
    - `set` - W/O. Enable TALK_DETECT and/or configure talk detection parameters. Can be called multiple times to change parameters on a channel with talk detection already enabled.
        - `dsp_silence_threshold` - The time in milliseconds before which a user is considered silent.
        - `dsp_talking_threshold` - The time in milliseconds after which a user is considered talking.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 13 Application_GotoIfTime

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_TOLOWER

TOLOWER()

Synopsis

Convert string to all lowercase letters.

Description

Example: ${TOLOWER(Example)} returns "example"

Syntax

```
TOLOWER(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Function_TOUPPER

TOUPPER()

Synopsis

Convert string to all uppercase letters.

Description

Example: ${TOUPPER(Example)} returns "EXAMPLE"

Syntax

```
TOUPPER(string)
```

Arguments

- `string`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
TRYLOCK(lockname)
```

Arguments

- `lockname`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

> **Note**
> If `live_dangerously` in `asterisk.conf` is set to `no`, this function can only be executed from the dialplan, and not directly from external protocols.

Syntax

```
UNLOCK(lockname)
```

Arguments

- `lockname`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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`.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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.

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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 mailbox *folder*.

Example: `exten => s,1,Set(foo=${VMCOUNT(125@default)})`

Syntax

```
VMCOUNT(vmbox[,folder])
```

Arguments

- `vmbox`
- `folder` - If not specified, defaults to `INBOX`

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 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

See Also

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Module Configuration

Asterisk 13 Configuration_app_agent_pool

Agent pool applications

This configuration documentation is for functionality provided by `app_agent_pool`.

Overview

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

agents.conf

global

Unused, but reserved.

agent-id

Configure an agent for the pool.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`ackcall`	`Boolean`	`no`	`false`	Enable to require the agent to acknowledge a call.
`acceptdtmf`	`String`	`#`	`false`	DTMF key sequence the agent uses to acknowledge a call.
`autologoff`	`Unsigned Integer`	`0`	`false`	Time the agent has to acknowledge a call before being logged off.
`wrapuptime`	`Unsigned Integer`	`0`	`false`	Minimum time the agent has between calls.
`musiconhold`	`String`	`default`	`false`	Music on hold class the agent listens to between calls.
`recordagentcalls`	`Boolean`	`no`	`false`	Enable to automatically record calls the agent takes.
`custom_beep`	`String`	`beep`	`false`	Sound file played to alert the agent when a call is present.
`fullname`	`String`		`false`	A friendly name for the agent used in log messages.

Configuration Option Descriptions

ackcall

Enable to require the agent to give a DTMF acknowledgement when the agent receives a call.

> **Note**
> The option is overridden by `AGENTACKCALL` on agent login.

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

acceptdtmf

> **Note**
> The option is overridden by `AGENTACCEPTDTMF` on agent login.

> **Note**
> The option is ignored unless the ackcall option is enabled.

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

autologoff

Set how many seconds a call for the agent has to wait for the agent to acknowledge the call before the agent is automatically logged off. If set to zero then the call will wait forever for the agent to acknowledge.

> **Note**
> The option is overridden by `AGENTAUTOLOGOFF` on agent login.

> **Note**
> The option is ignored unless the ackcall option is enabled.

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

wrapuptime

Set the minimum amount of time in milliseconds after disconnecting a call before the agent can receive a new call.

> **Note**
> The option is overridden by `AGENTWRAPUPTIME` on agent login.

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

musiconhold

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

recordagentcalls

Enable recording calls the agent takes automatically by invoking the automixmon DTMF feature when the agent connects to a caller. See `features.conf.sample` for information about the automixmon feature.

> **Note**
> Option changes take effect on agent login or after an agent disconnects from a call.

custom_beep

> **Note**

Option changes take effect on agent login or after an agent disconnects from a call.

fullname

Note
Option changes take effect on agent login or after an agent disconnects from a call.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_app_confbridge

Conference Bridge Application

This configuration documentation is for functionality provided by `app_confbridge`.

confbridge.conf

global

Unused, but reserved.

user_profile

A named profile to apply to specific callers.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`type`	`None`		`false`	Define this configuration category as a user profile.
`admin`	`Boolean`	`no`	`false`	Sets if the user is an admin or not
`marked`	`Boolean`	`no`	`false`	Sets if this is a marked user or not
`startmuted`	`Boolean`	`no`	`false`	Sets if all users should start out muted
`music_on_hold_when_empty`	`Boolean`	`no`	`false`	Play MOH when user is alone or waiting on a marked user
`quiet`	`Boolean`	`no`	`false`	Silence enter/leave prompts and user intros for this user
`announce_user_count`	`Boolean`	`no`	`false`	Sets if the number of users should be announced to the user
`announce_user_count_all`	`Custom`	`no`	`false`	Announce user count to all the other users when this user joins
`announce_only_user`	`Boolean`	`yes`	`false`	Announce to a user when they join an empty conference
`wait_marked`	`Boolean`	`no`	`false`	Sets if the user must wait for a marked user to enter before joining a conference
`end_marked`	`Boolean`	`no`	`false`	Kick the user from the conference when the last marked user leaves
`talk_detection_events`	`Boolean`	`no`	`false`	Set whether or not notifications of when a user begins and ends talking should be sent out as events over AMI
`dtmf_passthrough`	`Boolean`	`no`	`false`	Sets whether or not DTMF should pass through the conference
`announce_join_leave`	`Boolean`	`no`	`false`	Prompt user for their name when joining a conference and play it to the conference when they enter

announce_join_leave_review	Boolean	no	false	Prompt user for their name when joining a conference and play it to the conference when they enter. The user will be asked to review the recording of their name before entering the conference.
pin	String		false	Sets a PIN the user must enter before joining the conference
music_on_hold_class	String		false	The MOH class to use for this user
announcement	String		false	Sound file to play to the user when they join a conference
denoise	Boolean	no	false	Apply a denoise filter to the audio before mixing
dsp_drop_silence	Boolean	no	false	Drop what Asterisk detects as silence from audio sent to the bridge
dsp_silence_threshold	Unsigned Integer	2500	false	The number of milliseconds of detected silence necessary to trigger silence detection
dsp_talking_threshold	Unsigned Integer	160	false	The number of milliseconds of detected non-silence necessary to triger talk detection
jitterbuffer	Boolean	no	false	Place a jitter buffer on the user's audio stream before audio mixing is performed
template	Custom		false	When using the CONFBRIDGE dialplan function, use a user profile as a template for creating a new temporary profile

Configuration Option Descriptions

type

The type parameter determines how a context in the configuration file is interpreted.

- `user` - Configure the context as a *user_profile*
- `bridge` - Configure the context as a *bridge_profile*
- `menu` - Configure the context as a *menu*

announce_user_count_all

Sets if the number of users should be announced to all the other users in the conference when this user joins. This option can be either set to 'yes' or a number. When set to a number, the announcement will only occur once the user count is above the specified number.

denoise

Sets whether or not a denoise filter should be applied to the audio before mixing or not. Off by default. Requires `codec_speex` to be built and installed. Do not confuse this option with *drop_silence*. Denoise is useful if there is a lot of background noise for a user as it attempts to remove the noise while preserving the speech. This option does NOT remove silence from being mixed into the conference and does come at the cost of a slight performance hit.

dsp_drop_silence

This option drops what Asterisk detects as silence from entering into the bridge. Enabling this option will drastically improve performance and help remove the buildup of background noise from the conference. Highly recommended for large conferences due to its performance enhancements.

dsp_silence_threshold

The time in milliseconds of sound falling within the what the dsp has established as baseline silence before a user is considered be silent. This value affects several operations and should not be changed unless the impact on call quality is fully understood.

What this value affects internally:

1. When talk detection AMI events are enabled, this value determines when the user has stopped talking after a period of talking. If this value is set too low AMI events indicating the user has stopped talking may get falsely sent out when the user briefly pauses during mid sentence.

2. The *drop_silence* option depends on this value to determine when the user's audio should begin to be dropped from the conference bridge after the user stops talking. If this value is set too low the user's audio stream may sound choppy to the other participants. This is caused by the user transitioning constantly from silence to talking during mid sentence.

The best way to approach this option is to set it slightly above the maximum amount of ms of silence a user may generate during natural speech.

By default this value is 2500ms. Valid values are 1 through 2^31.

dsp_talking_threshold

The time in milliseconds of sound above what the dsp has established as base line silence for a user before a user is considered to be talking. This value affects several operations and should not be changed unless the impact on call quality is fully understood.

What this value affects internally:

1. Audio is only mixed out of a user's incoming audio stream if talking is detected. If this value is set too loose the user will hear themselves briefly each time they begin talking until the dsp has time to establish that they are in fact talking.

2. When talk detection AMI events are enabled, this value determines when talking has begun which results in an AMI event to fire. If this value is set too tight AMI events may be falsely triggered by variants in room noise.

3. The *drop_silence* option depends on this value to determine when the user's audio should be mixed into the bridge after periods of silence. If this value is too loose the beginning of a user's speech will get cut off as they transition from silence to talking.

By default this value is 160 ms. Valid values are 1 through 2^31

jitterbuffer

Enabling this option places a jitterbuffer on the user's audio stream before audio mixing is performed. This is highly recommended but will add a slight delay to the audio. This option is using the `JITTERBUFFER` dialplan function's default adaptive jitterbuffer. For a more fine tuned jitterbuffer, disable this option and use the `JITTERBUFFER` dialplan function on the user before entering the ConfBridge application.

bridge_profile

A named profile to apply to specific bridges.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`type`	`None`		`false`	Define this configuration category as a bridge profile
`jitterbuffer`	`Boolean`	`no`	`false`	Place a jitter buffer on the conference's audio stream
`internal_sample_rate`	`Unsigned Integer`	`0`	`false`	Set the internal native sample rate for mixing the conference
`language`	`String`	`en`	`false`	The language used for announcements to the conference.
`mixing_interval`	`Custom`	`20`	`false`	Sets the internal mixing interval in milliseconds for the bridge

record_conference	Boolean	no	false	Record the conference starting with the first active user's entrance and ending with the last active user's exit
record_file	String	confbridge-name of conference bridge-start time.wav	false	The filename of the conference recording
record_file_append	Boolean	yes	false	Append record file when starting/stopping on same conference recording
video_mode	Custom		false	Sets how confbridge handles video distribution to the conference participants
max_members	Unsigned Integer	0	false	Limit the maximum number of participants for a single conference
sound_	Custom		true	Override the various conference bridge sound files
template	Custom		false	When using the CONFBRIDGE dialplan function, use a bridge profile as a template for creating a new temporary profile

Configuration Option Descriptions

type

The type parameter determines how a context in the configuration file is interpreted.

- `user` - Configure the context as a *user_profile*
- `bridge` - Configure the context as a *bridge_profile*
- `menu` - Configure the context as a *menu*

internal_sample_rate

Sets the internal native sample rate the conference is mixed at. This is set to automatically adjust the sample rate to the best quality by default. Other values can be anything from 8000-192000. If a sample rate is set that Asterisk does not support, the closest sample rate Asterisk does support to the one requested will be used.

language

By default, announcements to a conference use English. Which means the prompts played to all users within the conference will be English. By changing the language of a bridge, this will change the language of the prompts played to all users.

mixing_interval

Sets the internal mixing interval in milliseconds for the bridge. This number reflects how tight or loose the mixing will be for the conference. In order to improve performance a larger mixing interval such as 40ms may be chosen. Using a larger mixing interval comes at the cost of introducing larger amounts of delay into the bridge. Valid values here are 10, 20, 40, or 80.

record_conference

Records the conference call starting when the first user enters the room, and ending when the last user exits the room. The default recorded filename is `'confbridge-${name of conference bridge}-${start time}.wav'` and the default format is 8khz slinear. This file will be located in the configured monitoring directory in `asterisk.conf`.

record_file

When *record_conference* is set to yes, the specific name of the record file can be set using this option. Note that since multiple conferences may use the same bridge profile, this may cause issues depending on the configuration. It is recommended to only use this option dynamically with the `CONFBRIDGE()` dialplan function. This allows the record name to be specified and a unique name to be chosen. By default, the record_file is stored in Asterisk's spool/monitor directory with a unique filename starting with the 'confbridge' prefix.

record_file_append

When *record_file_append* is set to yes, stopping and starting recording on a conference adds the new portion to end of current record_file. When this is set to no, a new *record_file* is generated every time you start then stop recording on a conference.

video_mode

Sets how confbridge handles video distribution to the conference participants. Note that participants wanting to view and be the source of a video feed **MU ST** be sharing the same video codec. Also, using video in conjunction with with the jitterbuffer currently results in the audio being slightly out of sync with the video. This is a result of the jitterbuffer only working on the audio stream. It is recommended to disable the jitterbuffer when video is used.

- `none` - No video sources are set by default in the conference. It is still possible for a user to be set as a video source via AMI or DTMF action at any time.
- `follow_talker` - The video feed will follow whoever is talking and providing video.
- `last_marked` - The last marked user to join the conference with video capabilities will be the single source of video distributed to all participants. If multiple marked users are capable of video, the last one to join is always the source, when that user leaves it goes to the one who joined before them.
- `first_marked` - The first marked user to join the conference with video capabilities is the single source of video distribution among all participants. If that user leaves, the marked user to join after them becomes the source.

max_members

This option limits the number of participants for a single conference to a specific number. By default conferences have no participant limit. After the limit is reached, the conference will be locked until someone leaves. Note however that an Admin user will always be alowed to join the conference regardless if this limit is reached or not.

sound_

All sounds in the conference are customizable using the bridge profile options below. Simply state the option followed by the filename or full path of the filename after the option. Example: `sound_had_joined=conf-hasjoin` This will play the `conf-hasjoin` sound file found in the sounds directory when announcing someone's name is joining the conference.

- `sound_join` - The sound played to everyone when someone enters the conference.
- `sound_leave` - The sound played to everyone when someone leaves the conference.
- `sound_has_joined` - The sound played before announcing someone's name has joined the conference. This is used for user intros. Example `"_____ has joined the conference"`
- `sound_has_left` - The sound played when announcing someone's name has left the conference. This is used for user intros. Example `"_____ has left the conference"`
- `sound_kicked` - The sound played to a user who has been kicked from the conference.
- `sound_muted` - The sound played when the mute option it toggled on.
- `sound_unmuted` - The sound played when the mute option it toggled off.
- `sound_only_person` - The sound played when the user is the only person in the conference.
- `sound_only_one` - The sound played to a user when there is only one other person is in the conference.
- `sound_there_are` - The sound played when announcing how many users there are in a conference.
- `sound_other_in_party` - This file is used in conjunction with `sound_there_are` when announcing how many users there are in the conference. The sounds are stringed together like this. `"sound_there_are" ${number of participants} "sound_other_in_party"`
- `sound_place_into_conference` - The sound played when someone is placed into the conference after waiting for a marked user.
- `sound_wait_for_leader` - The sound played when a user is placed into a conference that can not start until a marked user enters.
- `sound_leader_has_left` - The sound played when the last marked user leaves the conference.
- `sound_get_pin` - The sound played when prompting for a conference pin number.
- `sound_invalid_pin` - The sound played when an invalid pin is entered too many times.
- `sound_locked` - The sound played to a user trying to join a locked conference.
- `sound_locked_now` - The sound played to an admin after toggling the conference to locked mode.
- `sound_unlocked_now` - The sound played to an admin after toggling the conference to unlocked mode.
- `sound_error_menu` - The sound played when an invalid menu option is entered.

menu

A conference user menu

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
type	None		false	Define this configuration category as a menu
template	Custom		false	When using the CONFBRIDGE dialplan function, use a menu profile as a template for creating a new temporary profile
0-9A-D*#	Custom		true	DTMF sequences to assign various confbridge actions to

Configuration Option Descriptions

type

The type parameter determines how a context in the configuration file is interpreted.

- `user` - Configure the context as a *user_profile*
- `bridge` - Configure the context as a *bridge_profile*
- `menu` - Configure the context as a *menu*

0-9A-D*#

The ConfBridge application also has the ability to apply custom DTMF menus to each channel using the application. Like the User and Bridge profiles a menu is passed in to ConfBridge as an argument in the dialplan.

Below is a list of menu actions that can be assigned to a DTMF sequence.

> **Note**
> To have the first DTMF digit in a sequence be the '#' character, you need to escape it. If it is not escaped then normal config file processing will think it is a directive like #include. For example: The mute setting is toggled when `#1` is pressed.
>
> #1=toggle_mute

> **Note**
> A single DTMF sequence can have multiple actions associated with it. This is accomplished by stringing the actions together and using a `,` as the delimiter. Example: Both listening and talking volume is reset when `5` is pressed. `5=reset_talking_volume, reset_listening_volume`

- `playback(filename&filename2&...)` - `playback` will play back an audio file to a channel and then immediately return to the conference. This file can not be interupted by DTMF. Multiple files can be chained together using the `&` character.
- `playback_and_continue(filename&filename2&...)` - `playback_and_continue` will play back a prompt while continuing to collect the dtmf sequence. This is useful when using a menu prompt that describes all the menu options. Note however that any DTMF during this action will terminate the prompts playback. Prompt files can be chained together using the `&` character as a delimiter.
- `toggle_mute` - Toggle turning on and off mute. Mute will make the user silent to everyone else, but the user will still be able to listen in.
- `no_op` - This action does nothing (No Operation). Its only real purpose exists for being able to reserve a sequence in the config as a menu exit sequence.
- `decrease_listening_volume` - Decreases the channel's listening volume.
- `increase_listening_volume` - Increases the channel's listening volume.
- `reset_listening_volume` - Reset channel's listening volume to default level.
- `decrease_talking_volume` - Decreases the channel's talking volume.
- `increase_talking_volume` - Increases the channel's talking volume.
- `reset_talking_volume` - Reset channel's talking volume to default level.
- `dialplan_exec(context,exten,priority)` - The `dialplan_exec` action allows a user to escape from the conference and execute commands in the dialplan. Once the dialplan exits the user will be put back into the conference. The possibilities are endless!
- `leave_conference` - This action allows a user to exit the conference and continue execution in the dialplan.
- `admin_kick_last` - This action allows an Admin to kick the last participant from the conference. This action will only work for admins which allows a single menu to be used for both users and admins.
- `admin_toggle_conference_lock` - This action allows an Admin to toggle locking and unlocking the conference. Non admins can not use this action even if it is in their menu.
- `set_as_single_video_src` - This action allows any user to set themselves as the single video source distributed to all participants. This will make the video feed stick to them regardless of what the `video_mode` is set to.
- `release_as_single_video_src` - This action allows a user to release themselves as the video source. If `video_mode` is not set to n

`one` this action will result in the conference returning to whatever video mode the bridge profile is using.
Note that this action will have no effect if the user is not currently the video source. Also, the user is not guaranteed by using this action that they will not become the video source again. The bridge will return to whatever operation the `video_mode` option is set to upon release of the video src.

- `admin_toggle_mute_participants` - This action allows an administrator to toggle the mute state for all non-admins within a conference. All admin users are unaffected by this option. Note that all users, regardless of their admin status, are notified that the conference is muted.
- `participant_count` - This action plays back the number of participants currently in a conference

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_app_skel

This configuration documentation is for functionality provided by `app_skel`.

app_skel.conf

globals

Options that apply globally to app_skel

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`games`				The number of games a single execution of SkelGuessNumber will play
`cheat`				Should the computer cheat?

Configuration Option Descriptions

cheat

If enabled, the computer will ignore winning guesses.

sounds

Prompts for SkelGuessNumber to play

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`prompt`		`please-enter-yournumberqueue-less-than`		A prompt directing the user to enter a number less than the max number
`wrong_guess`		`vm-pls-try-again`		The sound file to play when a wrong guess is made
`right_guess`		`auth-thankyou`		The sound file to play when a correct guess is made
`too_low`				The sound file to play when a guess is too low
`too_high`				The sound file to play when a guess is too high
`lose`		`vm-goodbye`		The sound file to play when a player loses

level

Defined levels for the SkelGuessNumber game

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`max_number`				The maximum in the range of numbers to guess (1 is the implied minimum)
`max_guesses`				The maximum number of guesses before a game is considered lost

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_cdr

Call Detail Record configuration

This configuration documentation is for functionality provided by `cdr`.

Overview

CDR is Call Detail Record, which provides logging services via a variety of pluggable backend modules. Detailed call information can be recorded to databases, files, etc. Useful for billing, fraud prevention, compliance with Sarbanes-Oxley aka The Enron Act, QOS evaluations, and more.

cdr.conf

general

Global settings applied to the CDR engine.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`debug`	`Boolean`		`false`	Enable/disable verbose CDR debugging.
`enable`	`Boolean`	`1`	`false`	Enable/disable CDR logging.
`unanswered`	`Boolean`	`0`	`false`	Log calls that are never answered.
`congestion`	`Boolean`		`false`	Log congested calls.
`endbeforehexten`	`Boolean`	`1`	`false`	Don't produce CDRs while executing hangup logic
`initiatedseconds`	`Boolean`	`0`	`false`	Count microseconds for billsec purposes
`batch`	`Boolean`	`0`	`false`	Submit CDRs to the backends for processing in batches
`size`	`Unsigned Integer`	`100`	`false`	The maximum number of CDRs to accumulate before triggering a batch
`time`	`Unsigned Integer`	`300`	`false`	The maximum time to accumulate CDRs before triggering a batch
`scheduleronly`	`Boolean`	`0`	`false`	Post batched CDRs on their own thread instead of the scheduler
`safeshutdown`	`Boolean`	`1`	`false`	Block shutdown of Asterisk until CDRs are submitted

Configuration Option Descriptions

debug

When set to `True`, verbose updates of changes in CDR information will be logged. Note that this is only of use when debugging CDR behavior.

enable

Define whether or not to use CDR logging. Setting this to "no" will override any loading of backend CDR modules. Default is "yes".

unanswered

Define whether or not to log unanswered calls. Setting this to "yes" will report every attempt to ring a phone in dialing attempts, when it was not answered. For example, if you try to dial 3 extensions, and this option is "yes", you will get 3 CDR's, one for each phone that was rung. Some find this information horribly useless. Others find it very valuable. Note, in "yes" mode, you will see one CDR, with one of the call targets on one side, and the originating channel on the other, and then one CDR for each channel attempted. This may seem redundant, but cannot be helped.

In brief, this option controls the reporting of unanswered calls which only have an A party. Calls which get offered to an outgoing line, but are unanswered, are still logged, and that is the intended behavior. (It also results in some B side CDRs being output, as they have the B side channel as their source channel, and no destination channel.)

congestion

Define whether or not to log congested calls. Setting this to "yes" will report each call that fails to complete due to congestion conditions.

endbeforehexten

As each CDR for a channel is finished, its end time is updated and the CDR is finalized. When a channel is hung up and hangup logic is present (in the form of a hangup handler or the `h` extension), a new CDR is generated for the channel. Any statistics are gathered from this new CDR. By enabling this option, no new CDR is created for the dialplan logic that is executed in `h` extensions or attached hangup handler subroutines. The default value is `yes`, indicating that a CDR will be generated during hangup logic.

initiatedseconds

Normally, the `billsec` field logged to the CDR backends is simply the end time (hangup time) minus the answer time in seconds. Internally, asterisk stores the time in terms of microseconds and seconds. By setting initiatedseconds to `yes`, you can force asterisk to report any seconds that were initiated (a sort of round up method). Technically, this is when the microsecond part of the end time is greater than the microsecond part of the answer time, then the billsec time is incremented one second.

batch

Define the CDR batch mode, where instead of posting the CDR at the end of every call, the data will be stored in a buffer to help alleviate load on the asterisk server.

> **Warning**
> Use of batch mode may result in data loss after unsafe asterisk termination, i.e., software crash, power failure, kill -9, etc.

size

Define the maximum number of CDRs to accumulate in the buffer before posting them to the backend engines. batch must be set to `yes`.

time

Define the maximum time to accumulate CDRs before posting them in a batch to the backend engines. If this time limit is reached, then it will post the records, regardless of the value defined for size. batch must be set to `yes`.

> **Note**
> Time is expressed in seconds.

scheduleronly

The CDR engine uses the internal asterisk scheduler to determine when to post records. Posting can either occur inside the scheduler thread, or a new thread can be spawned for the submission of every batch. For small batches, it might be acceptable to just use the scheduler thread, so set this to `yes`. For large batches, say anything over size=10, a new thread is recommended, so set this to `no`.

safeshutdown

When shutting down asterisk, you can block until the CDRs are submitted. If you don't, then data will likely be lost. You can always check the size of the CDR batch buffer with the CLI `cdr status` command. To enable blocking on submission of CDR data during asterisk shutdown, set this to `yes`.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_cel

This configuration documentation is for functionality provided by `cel`.

cel.conf

general

Options that apply globally to Channel Event Logging (CEL)

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`enable`	`Boolean`	`no`	`false`	Determines whether CEL is enabled
`dateformat`	`String`		`false`	The format to be used for dates when logging
`apps`	`Custom`		`false`	List of apps for CEL to track
`events`	`Custom`		`false`	List of events for CEL to track

Configuration Option Descriptions

apps

A case-insensitive, comma-separated list of applications to track when one or both of APP_START and APP_END events are flagged for tracking

events

A case-sensitive, comma-separated list of event names to track. These event names do not include the leading `AST_CEL`.

- `ALL` - Special value which tracks all events.
- `CHAN_START`
- `CHAN_END`
- `ANSWER`
- `HANGUP`
- `APP_START`
- `APP_END`
- `PARK_START`
- `PARK_END`
- `USER_DEFINED`
- `BRIDGE_ENTER`
- `BRIDGE_EXIT`
- `BLINDTRANSFER`
- `ATTENDEDTRANSFER`
- `PICKUP`
- `FORWARD`
- `LINKEDID_END`
- `LOCAL_OPTIMIZE`

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_chan_motif

Jingle Channel Driver

This configuration documentation is for functionality provided by `chan_motif`.

Overview

Transports

There are three different transports and protocol derivatives supported by `chan_motif`. They are in order of preference: Jingle using ICE-UDP, Google Jingle, and Google-V1.

Jingle as defined in XEP-0166 supports the widest range of features. It is referred to as `ice-udp`. This is the specification that Jingle clients implement.

Google Jingle follows the Jingle specification for signaling but uses a custom transport for media. It is supported by the Google Talk Plug-in in Gmail and by some other Jingle clients. It is referred to as `google` in this file.

Google-V1 is the original Google Talk signaling protocol which uses an initial preliminary version of Jingle. It also uses the same custom transport as Google Jingle for media. It is supported by Google Voice, some other Jingle clients, and the Windows Google Talk client. It is referred to as `google-v1` in this file.

Incoming sessions will automatically switch to the correct transport once it has been determined.

Outgoing sessions are capable of determining if the target is capable of Jingle or a Google transport if the target is in the roster. Unfortunately it is not possible to differentiate between a Google Jingle or Google-V1 capable resource until a session initiate attempt occurs. If a resource is determined to use a Google transport it will initially use Google Jingle but will fall back to Google-V1 if required.

If an outgoing session attempt fails due to failure to support the given transport `chan_motif` will fall back in preference order listed previously until all transports have been exhausted.

Dialing and Resource Selection Strategy

Placing a call through an endpoint can be accomplished using the following dial string:

Motif/endpoint name/target

When placing an outgoing call through an endpoint the requested target is searched for in the roster list. If present the first Jingle or Google Jingle capable resource is specifically targeted. Since the capabilities of the resource are known the outgoing session initiation will disregard the configured transport and use the determined one.

If the target is not found in the roster the target will be used as-is and a session will be initiated using the transport specified in this configuration file. If no transport has been specified the endpoint defaults to `ice-udp`.

Video Support

Support for video does not need to be explicitly enabled. Configuring any video codec on your endpoint will automatically enable it.

DTMF

The only supported method for DTMF is RFC2833. This is always enabled on audio streams and negotiated if possible.

Incoming Calls

Incoming calls will first look for the extension matching the name of the endpoint in the configured context. If no such extension exists the call will automatically fall back to the `s` extension.

CallerID

The incoming caller id number is populated with the username of the caller and the name is populated with the full identity of the caller. If you would like to perform authentication or filtering of incoming calls it is recommended that you use these fields to do so.

Outgoing caller id can **not** be set.

> **Warning**
> Multiple endpoints using the same connection is **NOT** supported. Doing so may result in broken calls.

motif.conf

endpoint

The configuration for an endpoint.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`context`	`String`	`default`	`false`	Default dialplan context that incoming sessions will be routed to
`callgroup`	`Custom`		`false`	A callgroup to assign to this endpoint.
`pickupgroup`	`Custom`		`false`	A pickup group to assign to this endpoint.
`language`	`String`		`false`	The default language for this endpoint.
`musicclass`	`String`		`false`	Default music on hold class for this endpoint.
`parkinglot`	`String`		`false`	Default parking lot for this endpoint.
`accountcode`	`String`		`false`	Accout code for CDR purposes
`allow`	`Codec`	`ulaw,alaw`	`false`	Codecs to allow
`disallow`	`Codec`	`all`	`false`	Codecs to disallow
`connection`	`Custom`		`false`	Connection to accept traffic on and on which to send traffic out
`transport`	`Custom`		`false`	The transport to use for the endpoint.
`maxicecandidates`	`Unsigned Integer`	`10`	`false`	Maximum number of ICE candidates to offer
`maxpayloads`	`Unsigned Integer`	`30`	`false`	Maximum number of pyaloads to offer

Configuration Option Descriptions

transport

The default outbound transport for this endpoint. Inbound messages are inferred. Allowed transports are `ice-udp`, `google`, or `google-v1`. Note that `chan_motif` will fall back to transport preference order if the transport value chosen here fails.

- `ice-udp` - The Jingle protocol, as defined in XEP 0166.
- `google` - The Google Jingle protocol, which follows the Jingle specification for signaling but uses a custom transport for media.
- `google-v1` - Google-V1 is the original Google Talk signaling protocol which uses an initial preliminary version of Jingle. It also uses the same custom transport as `google` for media.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_core

Bucket file API

This configuration documentation is for functionality provided by `core`.

bucket

bucket

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`scheme`	`String`		`false`	Scheme in use for bucket
`created`	`Custom`		`false`	Time at which the bucket was created
`modified`	`Custom`		`false`	Time at which the bucket was last modified

file

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`scheme`	`String`		`false`	Scheme in use for file
`created`	`Custom`		`false`	Time at which the file was created
`modified`	`Custom`		`false`	Time at which the file was last modified

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_features

Features Configuration

This configuration documentation is for functionality provided by `features`.

features.conf

globals

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`featuredigittimeout`	`Custom`	`1000`	`false`	Milliseconds allowed between digit presses when entering a feature code.
`courtesytone`	`Custom`		`false`	Sound to play when automon or automixmon is activated
`recordingfailsound`	`Custom`		`false`	Sound to play when automon or automixmon is attempted but fails to start
`transferdigittimeout`	`Custom`	`3`	`false`	Seconds allowed between digit presses when dialing a transfer destination
`atxfernoanswertimeout`	`Custom`	`15`	`false`	Seconds to wait for attended transfer destination to answer
`atxferdropcall`	`Custom`	`0`	`false`	Hang up the call entirely if the attended transfer fails
`atxferloopdelay`	`Custom`	`10`	`false`	Seconds to wait between attempts to re-dial transfer destination
`atxfercallbackretries`	`Custom`	`2`	`false`	Number of times to re-attempt dialing a transfer destination
`xfersound`	`Custom`	`beep`	`false`	Sound to play to during transfer and transfer-like operations.
`xferfailsound`	`Custom`	`beeperr`	`false`	Sound to play to a transferee when a transfer fails
`atxferabort`	`Custom`	`*1`	`false`	Digits to dial to abort an attended transfer attempt
`atxfercomplete`	`Custom`	`*2`	`false`	Digits to dial to complete an attended transfer
`atxferthreeway`	`Custom`	`*3`	`false`	Digits to dial to change an attended transfer into a three-way call
`atxferswap`	`Custom`	`*4`	`false`	Digits to dial to toggle who the transferrer is currently bridged to during an attended transfer
`pickupexten`	`Custom`	`*8`	`false`	Digits used for picking up ringing calls
`pickupsound`	`Custom`		`false`	Sound to play to picker when a call is picked up

pickupfailsound	Custom		false	Sound to play to picker when a call cannot be picked up

Configuration Option Descriptions

atxferdropcall

When this option is set to `no`, then Asterisk will attempt to re-call the transferrer if the call to the transfer target fails. If the call to the transferrer fails, then Asterisk will wait *atxferloopdelay* milliseconds and then attempt to dial the transfer target again. This process will repeat until *atxfercallbackretries* attempts to re-call the transferrer have occurred.

When this option is set to `yes`, then Asterisk will not attempt to re-call the transferrer if the call to the transfer target fails. Asterisk will instead hang up all channels involved in the transfer.

xfersound

This sound will play to the transferrer and transfer target channels when an attended transfer completes. This sound is also played to channels when performing an AMI `Bridge` action.

atxferabort

This option is only available to the transferrer during an attended transfer operation. Aborting a transfer results in the transfer being cancelled and the original parties in the call being re-bridged.

atxfercomplete

This option is only available to the transferrer during an attended transfer operation. Completing the transfer with a DTMF sequence is functionally equivalent to hanging up the transferrer channel during an attended transfer. The result is that the transfer target and transferees are bridged.

atxferthreeway

This option is only available to the transferrer during an attended transfer operation. Pressing this DTMF sequence will result in the transferrer, the transferees, and the transfer target all being in a single bridge together.

atxferswap

This option is only available to the transferrer during an attended transfer operation. Pressing this DTMF sequence will result in the transferrer swapping which party he is bridged with. For instance, if the transferrer is currently bridged with the transfer target, then pressing this DTMF sequence will cause the transferrer to be bridged with the transferees.

pickupexten

In order for the pickup attempt to be successful, the party attempting to pick up the call must either have a *namedpickupgroup* in common with a ringing party's *namedcallgroup* or must have a *pickupgroup* in common with a ringing party's *callgroup*.

featuremap

DTMF options that can be triggered during bridged calls

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
atxfer	Custom		false	DTMF sequence to initiate an attended transfer
blindxfer	Custom	#	false	DTMF sequence to initiate a blind transfer
disconnect	Custom	*	false	DTMF sequence to disconnect the current call
parkcall	Custom		false	DTMF sequence to park a call

automon	`Custom`		`false`	DTMF sequence to start or stop monitoring a call
automixmon	`Custom`		`false`	DTMF sequence to start or stop mixmonitoring a call

Configuration Option Descriptions

atxfer

The transferee parties will be placed on hold and the transferrer may dial an extension to reach a transfer target. During an attended transfer, the transferrer may consult with the transfer target before completing the transfer. Once the transferrer has hung up or pressed the *atxfercomplete* DTMF sequence, then the transferees and transfer target will be bridged.

blindxfer

The transferee parties will be placed on hold and the transferrer may dial an extension to reach a transfer target. During a blind transfer, as soon as the transfer target is dialed, the transferrer is hung up.

disconnect

Entering this DTMF sequence will cause the bridge to end, no matter the number of parties present

parkcall

The parking lot used to park the call is determined by using either the *PARKINGLOT* channel variable or a configured value on the channel (provided by the channel driver) if the variable is not present. If no configured value on the channel is present, then `"default"` is used. The call is parked in the next available space in the parking lot.

automon

This will cause the channel that pressed the DTMF sequence to be monitored by the `Monitor` application. The format for the recording is determined by the *TOUCH_MONITOR_FORMAT* channel variable. If this variable is not specified, then `wav` is the default. The filename is constructed in the following manner:

prefix-timestamp-filename

where prefix is either the value of the *TOUCH_MONITOR_PREFIX* channel variable or `auto` if the variable is not set. The timestamp is a UNIX timestamp. The filename is either the value of the *TOUCH_MONITOR* channel variable or the callerID of the channels if the variable is not set.

automixmon

Operation of the automixmon is similar to the {{ automon }} feature, with the following exceptions: *TOUCH_MIXMONITOR* is used in place of *TOUCH_MONITOR* *TOUCH_MIXMONITOR_FORMAT* is used in place of *TOUCH_MIXMONITOR* There is no equivalent for *TOUCH_MONITOR_PREFIX*. `"auto"` is always how the filename begins.

applicationmap

Section for defining custom feature invocations during a call

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
.*	`Custom`		`true`	A custom feature to invoke during a bridged call

Configuration Option Descriptions

.*

Each item listed here is a comma-separated list of parameters that determine how a feature may be invoked during a call

Example:

eggs = *5,self,Playback(hello-world),default

This would create a feature called `eggs` that could be invoked during a call by pressing the `*5`. The party that presses the DTMF sequence would then trigger the `Playback` application to play the `hello-world` file. The application invocation would happen on the party that pressed the DTMF sequence since `self` is specified. The other parties in the bridge would hear the `default` music on hold class during the playback.

In addition to the syntax outlined in this documentation, a backwards-compatible alternative is also allowed. The following applicationmap lines are functionally identical:

eggs = *5,self,Playback(hello-world),default

eggs = *5,self,Playback,hello-world,default

eggs = *5,self,Playback,"hello-world",default

featuregroup

Groupings of items from the applicationmap

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`.*`	`Custom`		`true`	Applicationmap item to place in the feature group

Configuration Option Descriptions

.*

Each item here must be a name of an item in the applicationmap. The argument may either be a new DTMF sequence to use for the item or it may be left blank in order to use the DTMF sequence specified in the applicationmap. For example:

eggs => *1

bacon =>

would result in the applicationmap items `eggs` and `bacon` being in the featuregroup. The former would have its default DTMF trigger overridden with `*1` an d the latter would have the DTMF value specified in the applicationmap.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_named_acl

This configuration documentation is for functionality provided by `named_acl`.

named_acl.conf

named_acl

Options for configuring a named ACL

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`permit`	`ACL`		`false`	An address/subnet from which to allow access
`deny`	`ACL`		`false`	An address/subnet from which to disallow access

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_res_ari

HTTP binding for the Stasis API

This configuration documentation is for functionality provided by `res_ari`.

ari.conf

general

General configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`enabled`	`Boolean`	`yes`	`false`	Enable/disable the ARI module
`websocket_write_timeout`	`Integer`	`100`	`false`	The timeout (in milliseconds) to set on WebSocket connections.
`pretty`	`Custom`	`no`	`false`	Responses from ARI are formatted to be human readable
`auth_realm`	`String`	`Asterisk REST Interface`	`false`	Realm to use for authentication. Defaults to Asterisk REST Interface.
`allowed_origins`	`String`		`false`	Comma separated list of allowed origins, for Cross-Origin Resource Sharing. May be set to * to allow all origins.

Configuration Option Descriptions

enabled

This option enables or disables the ARI module.

> **Note**
> ARI uses Asterisk's HTTP server, which must also be enabled in `http.conf`.

websocket_write_timeout

If a websocket connection accepts input slowly, the timeout for writes to it can be increased to keep it from being disconnected. Value is in milliseconds; default is 100 ms.

user

Per-user configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`type`	`None`		`false`	Define this configuration section as a user.
`read_only`	`Boolean`	`no`	`false`	When set to yes, user is only authorized for read-only requests

password	String		false	Crypted or plaintext password (see password_format)
password_format	Custom	plain	false	password_format may be set to plain (the default) or crypt. When set to crypt, crypt(3) is used to validate the password. A crypted password can be generated using mkpasswd -m sha-512. When set to plain, the password is in plaintext

Configuration Option Descriptions

type

- `user` - Configure this section as a *user*

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_hep

Resource for integration with Homer using HEPv3

This configuration documentation is for functionality provided by `res_hep`.

hep.conf

general

General settings.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`enabled`		`yes`		Enable or disable packet capturing.
`capture_address`		`192.168.1.1:9061`		The address and port of the Homer server to send packets to.
`capture_password`				If set, the authentication password to send to Homer.
`capture_id`		`0`		The ID for this capture agent.

Configuration Option Descriptions

enabled

- `no`
- `yes`

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_mwi_external

Core external MWI support

This configuration documentation is for functionality provided by `res_mwi_external`.

sorcery.conf

mailboxes

Persistent cache of external MWI Mailboxes.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_parking

This configuration documentation is for functionality provided by `res_parking`.

res_parking.conf

globals

Options that apply to every parking lot

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
parkeddynamic	Boolean	no	false	Enables dynamically created parkinglots.

parking_lot

Defined parking lots for res_parking to use to park calls on

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
context	String	parkedcalls	false	The name of the context where calls are parked and picked up from.
parkext	String		false	Extension to park calls to this parking lot.
parkext_exclusive	Boolean	no	false	If yes, the extension registered as parkext will park exclusively to this parking lot.
parkpos	Custom	701-750	false	Numerical range of parking spaces which can be used to retrieve parked calls.
parkinghints	Boolean	no	false	If yes, this parking lot will add hints automatically for parking spaces.
parkingtime	Unsigned Integer	45	false	Amount of time a call will remain parked before giving up (in seconds).
parkedmusicclass	String		false	Which music class to use for parked calls. They will use the default if unspecified.
comebacktoorigin	Boolean	yes	false	Determines what should be done with the parked channel if no one picks it up before it times out.
comebackdialtime	Unsigned Integer	30	false	Timeout for the Dial extension created to call back the parker when a parked call times out.
comebackcontext	String	parkedcallstimeout	false	Context where parked calls will enter the PBX on timeout when comebacktoorigin=no
courtesytone	String		false	If the name of a sound file is provided, use this as the courtesy tone

parkedplay	Custom	caller	false	Who we should play the courtesytone to on the pickup of a parked call from this lot
parkedcalltransfers	Custom	no	false	Who to apply the DTMF transfer features to when parked calls are picked up or timeout.
parkedcallreparking	Custom	no	false	Who to apply the DTMF parking feature to when parked calls are picked up or timeout.
parkedcallhangup	Custom	no	false	Who to apply the DTMF hangup feature to when parked calls are picked up or timeout.
parkedcallrecording	Custom	no	false	Who to apply the DTMF MixMonitor recording feature to when parked calls are picked up or timeout.
findslot	Custom	first	false	Rule to use when trying to figure out which parking space a call should be parked with.
courtesytone				If set, the sound set will be played to whomever is set by parkedplay

Configuration Option Descriptions

context

This option is only used if parkext is set.

parkext

If this option is used, this extension will automatically be created to place calls into parking lots. In addition, if parkext_exclusive is set for this parking lot, the name of the parking lot will be included in the application's arguments so that it only parks to this parking lot. The extension will be created in `context`. Using this option also creates extensions for retrieving parked calls from the parking spaces in the same context.

parkpos

If parkext is set, these extensions will automatically be mapped in `context` in order to pick up calls parked to these parking spaces.

comebacktoorigin

Valid Options:

- `yes` - Automatically have the parked channel dial the device that parked the call with dial timeout set by the `parkingtime` option. When the call times out an extension to dial the PARKER will automatically be created in the `park-dial` context with an extension of the flattened parker device name. If the call is not answered, the parked channel that is timing out will continue in the dial plan at that point if there are more priorities in the extension (which won't be the case unless the dialplan deliberately includes such priorities in the `park-dial` context through pattern matching or deliberately written flattened peer extensions).
- `no` - Place the call into the PBX at `comebackcontext` instead. The extension will still be set as the flattened peer name. If an extension the flattened peer name isn't available then it will fall back to the `s` extension. If that also is unavailable it will attempt to fall back to `s@default`. The normal dial extension will still be created in the `park-dial` context with the extension also being the flattened peer name.

> **Note**
> Flattened Peer Names - Extensions can not include slash characters since those are used for pattern matching. When a peer name is flattened, slashes become underscores. For example if the parker of a call is called `SIP/0004F2040001` then flattened peer name and therefor the extensions created and used on timeouts will be `SIP_0004F204001`.

> **Note**
> When parking times out and the channel returns to the dial plan, the following variables are set:

- `PARKING_SPACE` - extension that the call was parked in prior to timing out.
- `PARKINGSLOT` - Deprecated. Use `PARKING_SPACE` instead.
- `PARKEDLOT` - name of the lot that the call was parked in prior to timing out.
- `PARKER` - The device that parked the call
- `PARKER_FLAT` - The flat version of `PARKER`

comebackcontext

The extension the call enters will prioritize the flattened peer name in this context. If the flattened peer name extension is unavailable, then the 's' extension in this context will be used. If that also is unavailable, the 's' extension in the 'default' context will be used.

courtesytone

By default, this tone is only played to the caller of a parked call. Who receives the tone can be changed using the `parkedplay` option.

parkedplay

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

> **Note**
> If courtesy tone is not specified then this option will be ignored.

parkedcalltransfers

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

parkedcallreparking

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

parkedcallhangup

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

parkedcallrecording

- `no` - Apply to neither side.
- `caller` - Apply only to the call connecting with the call coming out of the parking lot.
- `callee` - Apply only to the call coming out of the parking lot.
- `both` - Apply to both sides.

findslot

- `first` - Always try to place in the lowest available space in the parking lot
- `next` - Track the last parking space used and always attempt to use the one immediately after.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_res_pjsip

SIP Resource using PJProject

This configuration documentation is for functionality provided by `res_pjsip`.

pjsip.conf

endpoint

Endpoint

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
100rel	Custom	yes	false	Allow support for RFC3262 provisional ACK tags
aggregate_mwi	Boolean	yes	false	Condense MWI notifications into a single NOTIFY.
allow	Codec		false	Media Codec(s) to allow
aors	String		false	AoR(s) to be used with the endpoint
auth	Custom		false	Authentication Object(s) associated with the endpoint
callerid	Custom		false	CallerID information for the endpoint
callerid_privacy	Custom		false	Default privacy level
callerid_tag	Custom		false	Internal id_tag for the endpoint
context	String	default	false	Dialplan context for inbound sessions
direct_media_glare_mitigation	Custom	none	false	Mitigation of direct media (re)INVITE glare
direct_media_method	Custom	invite	false	Direct Media method type
connected_line_method	Custom	invite	false	Connected line method type
direct_media	Boolean	yes	false	Determines whether media may flow directly between endpoints.
disable_direct_media_on_nat	Boolean	no	false	Disable direct media session refreshes when NAT obstructs the media session
disallow				Media Codec(s) to disallow
dtmf_mode	Custom	rfc4733	false	DTMF mode
media_address	String		false	IP address used in SDP for media handling
force_rport	Boolean	yes	false	Force use of return port
ice_support	Boolean	no	false	Enable the ICE mechanism to help traverse NAT
identify_by	Custom	username	false	Way(s) for Endpoint to be identified
redirect_method	Custom	user	false	How redirects received from an endpoint are handled

mailboxes	String		false	NOTIFY the endpoint when state changes for any of the specified mailboxes
moh_suggest	String	default	false	Default Music On Hold class
outbound_auth	Custom		false	Authentication object used for outbound requests
outbound_proxy	String		false	Proxy through which to send requests, a full SIP URI must be provided
rewrite_contact	Boolean	no	false	Allow Contact header to be rewritten with the source IP address-port
rtp_ipv6	Boolean	no	false	Allow use of IPv6 for RTP traffic
rtp_symmetric	Boolean	no	false	Enforce that RTP must be symmetric
send_diversion	Boolean	yes	false	Send the Diversion header, conveying the diversion information to the called user agent
send_pai	Boolean	no	false	Send the P-Asserted-Identity header
send_rpid	Boolean	no	false	Send the Remote-Party-ID header
timers_min_se	Unsigned Integer	90	false	Minimum session timers expiration period
timers	Custom	yes	false	Session timers for SIP packets
timers_sess_expires	Unsigned Integer	1800	false	Maximum session timer expiration period
transport	String		false	Desired transport configuration
trust_id_inbound	Boolean	no	false	Accept identification information received from this endpoint
trust_id_outbound	Boolean	no	false	Send private identification details to the endpoint.
type	None		false	Must be of type 'endpoint'.
use_ptime	Boolean	no	false	Use Endpoint's requested packetisation interval
use_avpf	Boolean	no	false	Determines whether res_pjsip will use and enforce usage of AVPF for this endpoint.
force_avp	Boolean	no	false	Determines whether res_pjsip will use and enforce usage of AVP, regardless of the RTP profile in use for this endpoint.
media_use_received_transport	Boolean	no	false	Determines whether res_pjsip will use the media transport received in the offer SDP in the corresponding answer SDP.

media_encryption	Custom	no	false	Determines whether res_pjsip will use and enforce usage of media encryption for this endpoint.
inband_progress	Boolean	no	false	Determines whether chan_pjsip will indicate ringing using inband progress.
call_group	Custom		false	The numeric pickup groups for a channel.
pickup_group	Custom		false	The numeric pickup groups that a channel can pickup.
named_call_group	Custom		false	The named pickup groups for a channel.
named_pickup_group	Custom		false	The named pickup groups that a channel can pickup.
device_state_busy_at	Unsigned Integer	0	false	The number of in-use channels which will cause busy to be returned as device state
t38_udptl	Boolean	no	false	Whether T.38 UDPTL support is enabled or not
t38_udptl_ec	Custom	none	false	T.38 UDPTL error correction method
t38_udptl_maxdatagram	Unsigned Integer	0	false	T.38 UDPTL maximum datagram size
fax_detect	Boolean	no	false	Whether CNG tone detection is enabled
t38_udptl_nat	Boolean	no	false	Whether NAT support is enabled on UDPTL sessions
t38_udptl_ipv6	Boolean	no	false	Whether IPv6 is used for UDPTL Sessions
tone_zone	String		false	Set which country's indications to use for channels created for this endpoint.
language	String		false	Set the default language to use for channels created for this endpoint.
one_touch_recording	Boolean	no	false	Determines whether one-touch recording is allowed for this endpoint.
record_on_feature	String	automixmon	false	The feature to enact when one-touch recording is turned on.
record_off_feature	String	automixmon	false	The feature to enact when one-touch recording is turned off.
rtp_engine	String	asterisk	false	Name of the RTP engine to use for channels created for this endpoint
allow_transfer	Boolean	yes	false	Determines whether SIP REFER transfers are allowed for this endpoint
sdp_owner	String	-	false	String placed as the username portion of an SDP origin (o=) line.

sdp_session	String	Asterisk	false	String used for the SDP session (s=) line.
tos_audio	Custom	0	false	DSCP TOS bits for audio streams
tos_video	Custom	0	false	DSCP TOS bits for video streams
cos_audio	Unsigned Integer	0	false	Priority for audio streams
cos_video	Unsigned Integer	0	false	Priority for video streams
allow_subscribe	Boolean	yes	false	Determines if endpoint is allowed to initiate subscriptions with Asterisk.
sub_min_expiry	Unsigned Integer	0	false	The minimum allowed expiry time for subscriptions initiated by the endpoint.
from_user	String		false	Username to use in From header for requests to this endpoint.
mwi_from_user	String		false	Username to use in From header for unsolicited MWI NOTIFYs to this endpoint.
from_domain	String		false	Domain to user in From header for requests to this endpoint.
dtls_verify	Custom		false	Verify that the provided peer certificate is valid
dtls_rekey	Custom		false	Interval at which to renegotiate the TLS session and rekey the SRTP session
dtls_cert_file	Custom		false	Path to certificate file to present to peer
dtls_private_key	Custom		false	Path to private key for certificate file
dtls_cipher	Custom		false	Cipher to use for DTLS negotiation
dtls_ca_file	Custom		false	Path to certificate authority certificate
dtls_ca_path	Custom		false	Path to a directory containing certificate authority certificates
dtls_setup	Custom		false	Whether we are willing to accept connections, connect to the other party, or both.
srtp_tag_32	Boolean	no	false	Determines whether 32 byte tags should be used instead of 80 byte tags.
set_var	Custom		false	Variable set on a channel involving the endpoint.
message_context	String		false	Context to route incoming MESSAGE requests to.
accountcode	String		false	An accountcode to set automatically on any channels created for this endpoint.

Configuration Option Descriptions

100rel

- `no`
- `required`
- `yes`

aggregate_mwi

When enabled, *aggregate_mwi* condenses message waiting notifications from multiple mailboxes into a single NOTIFY. If it is disabled, individual NOTIFYs are sent for each mailbox.

aors

List of comma separated AoRs that the endpoint should be associated with.

auth

This is a comma-delimited list of *auth* sections defined in `pjsip.conf` to be used to verify inbound connection attempts.

Endpoints without an `authentication` object configured will allow connections without vertification.

callerid

Must be in the format `Name <Number>`, or only `<Number>`.

callerid_privacy

- `allowed_not_screened`
- `allowed_passed_screened`
- `allowed_failed_screened`
- `allowed`
- `prohib_not_screened`
- `prohib_passed_screened`
- `prohib_failed_screened`
- `prohib`
- `unavailable`

direct_media_glare_mitigation

This setting attempts to avoid creating INVITE glare scenarios by disabling direct media reINVITEs in one direction thereby allowing designated servers (according to this option) to initiate direct media reINVITEs without contention and significantly reducing call setup time.

A more detailed description of how this option functions can be found on the Asterisk wiki https://wiki.asterisk.org/wiki/display/AST/SIP+Direct+Media+Reinvite+Glare+Avoidance

- `none`
- `outgoing`
- `incoming`

direct_media_method

Method for setting up Direct Media between endpoints.

- `invite`
- `reinvite` - Alias for the `invite` value.
- `update`

connected_line_method

Method used when updating connected line information.

- `invite`
- `reinvite` - Alias for the `invite` value.
- `update`

dtmf_mode

This setting allows to choose the DTMF mode for endpoint communication.

- `rfc4733` - DTMF is sent out of band of the main audio stream.This supercedes the older **RFC-2833** used within the older `chan_sip`.
- `inband` - DTMF is sent as part of audio stream.
- `info` - DTMF is sent as SIP INFO packets.

media_address

At the time of SDP creation, the IP address defined here will be used as the media address for individual streams in the SDP.

> **Note**
> Be aware that the `external_media_address` option, set in Transport configuration, can also affect the final media address used in the SDP.

identify_by

An endpoint can be identified in multiple ways. Currently, the only supported option is `username`, which matches the endpoint based on the username in the From header.

> **Note**
> Endpoints can also be identified by IP address; however, that method of identification is not handled by this configuration option. See the documentation for the `identify` configuration section for more details on that method of endpoint identification. If this option is set to `username` and an `identify` configuration section exists for the endpoint, then the endpoint can be identified in multiple ways.

- `username`

redirect_method

When a redirect is received from an endpoint there are multiple ways it can be handled. If this option is set to `user` the user portion of the redirect target is treated as an extension within the dialplan and dialed using a Local channel. If this option is set to `uri_core` the target URI is returned to the dialing application which dials it using the PJSIP channel driver and endpoint originally used. If this option is set to `uri_pjsip` the redirect occurs within chan_pjsip itself and is not exposed to the core at all. The `uri_pjsip` option has the benefit of being more efficient and also supporting multiple potential redirect targets. The con is that since redirection occurs within chan_pjsip redirecting information is not forwarded and redirection can not be prevented.

- `user`
- `uri_core`
- `uri_pjsip`

mailboxes

Asterisk will send unsolicited MWI NOTIFY messages to the endpoint when state changes happen for any of the specified mailboxes. More than one mailbox can be specified with a comma-delimited string. app_voicemail mailboxes must be specified as mailbox@context; for example: mailboxes=6001@default. For mailboxes provided by external sources, such as through the res_external_mwi module, you must specify strings supported by the external system.

For endpoints that SUBSCRIBE for MWI, use the `mailboxes` option in your AOR configuration.

rewrite_contact

On inbound SIP messages from this endpoint, the Contact header will be changed to have the source IP address and port. This option does not affect outbound messages send to this endpoint.

timers_min_se

Minimium session timer expiration period. Time in seconds.

timers

- `forced`
- `no`
- `required`
- `yes`

timers_sess_expires

Maximium session timer expiration period. Time in seconds.

transport

This will set the desired transport configuration to send SIP data through.

> **Warning**
> Not specifying a transport will **DEFAULT** to the first configured transport in `pjsip.conf` which is valid for the URI we are trying to contact.

> **Warning**
> Transport configuration is not affected by reloads. In order to change transports, a full Asterisk restart is required

trust_id_inbound

This option determines whether Asterisk will accept identification from the endpoint from headers such as P-Asserted-Identity or Remote-Party-ID header. This option applies both to calls originating from the endpoint and calls originating from Asterisk. If `no`, the configured Caller-ID from pjsip.conf will always be used as the identity for the endpoint.

trust_id_outbound

This option determines whether res_pjsip will send private identification information to the endpoint. If `no`, private Caller-ID information will not be forwarded to the endpoint. "Private" in this case refers to any method of restricting identification. Example: setting *callerid_privacy* to any `prohib` variation. Example: If *trust_id_inbound* is set to `yes`, the presence of a `Privacy: id` header in a SIP request or response would indicate the identification provided in the request is private.

use_avpf

If set to `yes`, res_pjsip will use the AVPF or SAVPF RTP profile for all media offers on outbound calls and media updates and will decline media offers not using the AVPF or SAVPF profile.

If set to `no`, res_pjsip will use the AVP or SAVP RTP profile for all media offers on outbound calls and media updates, but will accept either the AVP/AVPF or SAVP/SAVPF RTP profile for all inbound media offers.

force_avp

If set to `yes`, res_pjsip will use the AVP, AVPF, SAVP, or SAVPF RTP profile for all media offers on outbound calls and media updates including those for DTLS-SRTP streams.

If set to `no`, res_pjsip will use the respective RTP profile depending on configuration.

media_use_received_transport

If set to `yes`, res_pjsip will use the received media transport.

If set to `no`, res_pjsip will use the respective RTP profile depending on configuration.

media_encryption

- `no` - res_pjsip will offer no encryption and allow no encryption to be setup.
- `sdes` - res_pjsip will offer standard SRTP setup via in-SDP keys. Encrypted SIP transport should be used in conjunction with this option to prevent exposure of media encryption keys.
- `dtls` - res_pjsip will offer DTLS-SRTP setup.

inband_progress

If set to `yes`, chan_pjsip will send a 183 Session Progress when told to indicate ringing and will immediately start sending ringing as audio.

If set to `no`, chan_pjsip will send a 180 Ringing when told to indicate ringing and will NOT send it as audio.

call_group

Can be set to a comma separated list of numbers or ranges between the values of 0-63 (maximum of 64 groups).

pickup_group

Can be set to a comma separated list of numbers or ranges between the values of 0-63 (maximum of 64 groups).

named_call_group

Can be set to a comma separated list of case sensitive strings limited by supported line length.

named_pickup_group

Can be set to a comma separated list of case sensitive strings limited by supported line length.

device_state_busy_at

When the number of in-use channels for the endpoint matches the devicestate_busy_at setting the PJSIP channel driver will return busy as the device state instead of in use.

t38_udptl

If set to yes T.38 UDPTL support will be enabled, and T.38 negotiation requests will be accepted and relayed.

t38_udptl_ec

- `none` - No error correction should be used.
- `fec` - Forward error correction should be used.
- `redundancy` - Redundacy error correction should be used.

t38_udptl_maxdatagram

This option can be set to override the maximum datagram of a remote endpoint for broken endpoints.

fax_detect

This option can be set to send the session to the fax extension when a CNG tone is detected.

t38_udptl_nat

When enabled the UDPTL stack will send UDPTL packets to the source address of received packets.

t38_udptl_ipv6

When enabled the UDPTL stack will use IPv6.

record_on_feature

When an INFO request for one-touch recording arrives with a Record header set to "on", this feature will be enabled for the channel. The feature designated here can be any built-in or dynamic feature defined in features.conf.

> **Note**
> This setting has no effect if the endpoint's one_touch_recording option is disabled

record_off_feature

When an INFO request for one-touch recording arrives with a Record header set to "off", this feature will be enabled for the channel. The feature designated here can be any built-in or dynamic feature defined in features.conf.

> **Note**

This setting has no effect if the endpoint's one_touch_recording option is disabled

tos_audio

See https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service for more information about QoS settings

tos_video

See https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service for more information about QoS settings

cos_audio

See https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service for more information about QoS settings

cos_video

See https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service for more information about QoS settings

dtls_verify

This option only applies if *media_encryption* is set to `dtls`.

dtls_rekey

This option only applies if *media_encryption* is set to `dtls`.

If this is not set or the value provided is 0 rekeying will be disabled.

dtls_cert_file

This option only applies if *media_encryption* is set to `dtls`.

dtls_private_key

This option only applies if *media_encryption* is set to `dtls`.

dtls_cipher

This option only applies if *media_encryption* is set to `dtls`.

Many options for acceptable ciphers. See link for more: http://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS

dtls_ca_file

This option only applies if *media_encryption* is set to `dtls`.

dtls_ca_path

This option only applies if *media_encryption* is set to `dtls`.

dtls_setup

This option only applies if *media_encryption* is set to `dtls`.

- `active` - res_pjsip will make a connection to the peer.
- `passive` - res_pjsip will accept connections from the peer.
- `actpass` - res_pjsip will offer and accept connections from the peer.

srtp_tag_32

This option only applies if *media_encryption* is set to `sdes` or `dtls`.

set_var

When a new channel is created using the endpoint set the specified variable(s) on that channel. For multiple channel variables specify multiple 'set_var'(s).

message_context

If specified, incoming MESSAGE requests will be routed to the indicated dialplan context. If no *message_context* is specified, then the *context* setting is used.

accountcode

If specified, any channel created for this endpoint will automatically have this accountcode set on it.

auth

Authentication type

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`auth_type`	`Custom`	`userpass`	`false`	Authentication type
`nonce_lifetime`	`Unsigned Integer`	`32`	`false`	Lifetime of a nonce associated with this authentication config.
`md5_cred`	`String`		`false`	MD5 Hash used for authentication.
`password`	`String`		`false`	PlainText password used for authentication.
`realm`	`String`		`false`	SIP realm for endpoint
`type`	`None`		`false`	Must be 'auth'
`username`	`String`		`false`	Username to use for account

Configuration Option Descriptions

auth_type

This option specifies which of the password style config options should be read when trying to authenticate an endpoint inbound request. If set to `userpass` then we'll read from the 'password' option. For `md5` we'll read from 'md5_cred'.

- `md5`
- `userpass`

md5_cred

Only used when auth_type is `md5`.

password

Only used when auth_type is `userpass`.

domain_alias

Domain Alias

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
type	None		false	Must be of type 'domain_alias'.
domain	String		false	Domain to be aliased

transport

SIP Transport

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
async_operations	Unsigned Integer	1	false	Number of simultaneous Asynchronous Operations
bind	Custom		false	IP Address and optional port to bind to for this transport
ca_list_file	String		false	File containing a list of certificates to read (TLS ONLY)
cert_file	String		false	Certificate file for endpoint (TLS ONLY)
cipher	Custom		false	Preferred Cryptography Cipher (TLS ONLY)
domain	String		false	Domain the transport comes from
external_media_address	String		false	External IP address to use in RTP handling
external_signaling_address	String		false	External address for SIP signalling
external_signaling_port	Unsigned Integer	0	false	External port for SIP signalling
method	Custom		false	Method of SSL transport (TLS ONLY)
local_net	Custom		false	Network to consider local (used for NAT purposes).
password	String		false	Password required for transport
priv_key_file	String		false	Private key file (TLS ONLY)
protocol	Custom	udp	false	Protocol to use for SIP traffic
require_client_cert	Custom		false	Require client certificate (TLS ONLY)
type	None		false	Must be of type 'transport'.
verify_client	Custom		false	Require verification of client certificate (TLS ONLY)
verify_server	Custom		false	Require verification of server certificate (TLS ONLY)
tos	Custom	0	false	Enable TOS for the signalling sent over this transport

`cos`	`Unsigned Integer`	`0`	`false`	Enable COS for the signalling sent over this transport
`websocket_write_timeout`	`Integer`	`100`	`false`	The timeout (in milliseconds) to set on WebSocket connections.

Configuration Option Descriptions

cipher

Many options for acceptable ciphers see link for more: http://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS

external_media_address

When a request or response is sent out, if the destination of the message is outside the IP network defined in the option `localnet`, and the media address in the SDP is within the localnet network, then the media address in the SDP will be rewritten to the value defined for `external_media_address`.

method

- `default`
- `unspecified`
- `tlsv1`
- `sslv2`
- `sslv3`
- `sslv23`

local_net

This must be in CIDR or dotted decimal format with the IP and mask separated with a slash ('/').

protocol

- `udp`
- `tcp`
- `tls`
- `ws`
- `wss`

tos

See `https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service` for more information on this parameter.

> **Note**
> This option does not apply to the *ws* or the *wss* protocols.

cos

See `https://wiki.asterisk.org/wiki/display/AST/IP+Quality+of+Service` for more information on this parameter.

> **Note**
> This option does not apply to the *ws* or the *wss* protocols.

websocket_write_timeout

If a websocket connection accepts input slowly, the timeout for writes to it can be increased to keep it from being disconnected. Value is in milliseconds; default is 100 ms.

contact

A way of creating an aliased name to a SIP URI

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
type	None		false	Must be of type 'contact'.
uri	String		false	SIP URI to contact peer
expiration_time	Custom		false	Time to keep alive a contact
qualify_frequency	Unsigned Integer	0	false	Interval at which to qualify a contact
outbound_proxy	String		false	Outbound proxy used when sending OPTIONS request
path	String		false	Stored Path vector for use in Route headers on outgoing requests.
user_agent	String		false	User-Agent header from registration.

Configuration Option Descriptions

expiration_time

Time to keep alive a contact. String style specification.

qualify_frequency

Interval between attempts to qualify the contact for reachability. If `0` never qualify. Time in seconds.

outbound_proxy

If set the provided URI will be used as the outbound proxy when an OPTIONS request is sent to a contact for qualify purposes.

user_agent

The User-Agent is automatically stored based on data present in incoming SIP REGISTER requests and is not intended to be configured manually.

aor

The configuration for a location of an endpoint

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
contact	Custom		false	Permanent contacts assigned to AoR
default_expiration	Unsigned Integer	3600	false	Default expiration time in seconds for contacts that are dynamically bound to an AoR.
mailboxes	String		false	Allow subscriptions for the specified mailbox(es)
maximum_expiration	Unsigned Integer	7200	false	Maximum time to keep an AoR
max_contacts	Unsigned Integer	0	false	Maximum number of contacts that can bind to an AoR

minimum_expiration	Unsigned Integer	60	false	Minimum keep alive time for an AoR
remove_existing	Boolean	no	false	Determines whether new contacts replace existing ones.
type	None		false	Must be of type 'aor'.
qualify_frequency	Unsigned Integer	0	false	Interval at which to qualify an AoR
authenticate_qualify	Boolean	no	false	Authenticates a qualify request if needed
outbound_proxy	String		false	Outbound proxy used when sending OPTIONS request
support_path	Boolean	no	false	Enables Path support for REGISTER requests and Route support for other requests.

Configuration Option Descriptions

contact

Contacts specified will be called whenever referenced by `chan_pjsip`.

Use a separate "contact=" entry for each contact required. Contacts are specified using a SIP URI.

mailboxes

This option applies when an external entity subscribes to an AoR for Message Waiting Indications. The mailboxes specified will be subscribed to. More than one mailbox can be specified with a comma-delimited string. app_voicemail mailboxes must be specified as mailbox@context; for example: mailboxes=6001@default. For mailboxes provided by external sources, such as through the res_external_mwi module, you must specify strings supported by the external system.

For endpoints that cannot SUBSCRIBE for MWI, you can set the `mailboxes` option in your endpoint configuration section to enable unsolicited MWI NOTIFYs to the endpoint.

maximum_expiration

Maximium time to keep a peer with explicit expiration. Time in seconds.

max_contacts

Maximum number of contacts that can associate with this AoR. This value does not affect the number of contacts that can be added with the "contact" option. It only limits contacts added through external interaction, such as registration.

> **Note**
> This should be set to `1` and *remove_existing* set to `yes` if you wish to stick with the older `chan_sip` behaviour.

minimum_expiration

Minimum time to keep a peer with an explict expiration. Time in seconds.

remove_existing

On receiving a new registration to the AoR should it remove the existing contact that was registered against it?

> **Note**
> This should be set to `yes` and *max_contacts* set to `1` if you wish to stick with the older `chan_sip` behaviour.

qualify_frequency

Interval between attempts to qualify the AoR for reachability. If `0` never qualify. Time in seconds.

authenticate_qualify

If true and a qualify request receives a challenge or authenticate response authentication is attempted before declaring the contact available.

outbound_proxy

If set the provided URI will be used as the outbound proxy when an OPTIONS request is sent to a contact for qualify purposes.

support_path

When this option is enabled, the Path headers in register requests will be saved and its contents will be used in Route headers for outbound out-of-dialog requests and in Path headers for outbound 200 responses. Path support will also be indicated in the Supported header.

system

Options that apply to the SIP stack as well as other system-wide settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`timer_t1`	`Unsigned Integer`	`500`	`false`	Set transaction timer T1 value (milliseconds).
`timer_b`	`Unsigned Integer`	`32000`	`false`	Set transaction timer B value (milliseconds).
`compact_headers`	`Boolean`	`no`	`false`	Use the short forms of common SIP header names.
`threadpool_initial_size`	`Unsigned Integer`	`0`	`false`	Initial number of threads in the res_pjsip threadpool.
`threadpool_auto_increment`	`Unsigned Integer`	`5`	`false`	The amount by which the number of threads is incremented when necessary.
`threadpool_idle_timeout`	`Unsigned Integer`	`60`	`false`	Number of seconds before an idle thread should be disposed of.
`threadpool_max_size`	`Unsigned Integer`	`0`	`false`	Maximum number of threads in the res_pjsip threadpool. A value of 0 indicates no maximum.
`type`	`None`		`false`	Must be of type 'system'.

Configuration Option Descriptions

timer_t1

Timer T1 is the base for determining how long to wait before retransmitting requests that receive no response when using an unreliable transport (e.g. UDP). For more information on this timer, see RFC 3261, Section 17.1.1.1.

timer_b

Timer B determines the maximum amount of time to wait after sending an INVITE request before terminating the transaction. It is recommended that this be set to 64 * Timer T1, but it may be set higher if desired. For more information on this timer, see RFC 3261, Section 17.1.1.1.

global

Options that apply globally to all SIP communications

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description	
`max_forwards`	`Unsigned Integer`	`70`	`false`	Value used in Max-Forwards header for SIP requests.	
`type`	`None`		`false`	Must be of type 'global'.	
`user_agent`	`String`	`Asterisk PBX SVN-branch-13-r420717`	`false`	Value used in User-Agent header for SIP requests and Server header for SIP responses.	
`default_outbound_endpoint`	`String`	`default_outbound_endpoint`	`false`	Endpoint to use when sending an outbound request to a URI without a specified endpoint.	
`debug`	`String`	`no`	`false`	Enable/Disable SIP debug logging. Valid options include yes	no or a host address

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_res_pjsip_acl

SIP ACL module

This configuration documentation is for functionality provided by `res_pjsip_acl`.

Overview

ACL

The ACL module used by `res_pjsip`. This module is independent of `endpoints` and operates on all inbound SIP communication using res_pjsip.

There are two main ways of defining your ACL with the options provided. You can use the `permit` and `deny` options which act on **IP** addresses, or the `contactpermit` and `contactdeny` options which act on **Contact header** addresses in incoming REGISTER requests. You can combine the various options to create a mixed ACL.

Additionally, instead of defining an ACL with options, you can reference IP or Contact header ACLs from the file `acl.conf` by using the `acl` or `contactacl` options.

pjsip.conf

acl

Access Control List

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`acl`	`Custom`		`false`	List of IP ACL section names in acl.conf
`contact_acl`	`Custom`		`false`	List of Contact ACL section names in acl.conf
`contact_deny`	`Custom`		`false`	List of Contact header addresses to deny
`contact_permit`	`Custom`		`false`	List of Contact header addresses to permit
`deny`	`Custom`		`false`	List of IP addresses to deny access from
`permit`	`Custom`		`false`	List of IP addresses to permit access from
`type`	`None`		`false`	Must be of type 'acl'.

Configuration Option Descriptions

acl

This matches sections configured in `acl.conf`. The value is defined as a list of comma-delimited section names.

contact_acl

This matches sections configured in `acl.conf`. The value is defined as a list of comma-delimited section names.

contact_deny

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

contact_permit

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

deny

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

permit

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dotted-decimal notation. Separate the IP address and subnet mask with a slash ('/')

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_pjsip_endpoint_identifier_ip

Module that identifies endpoints via source IP address

This configuration documentation is for functionality provided by `res_pjsip_endpoint_identifier_ip`.

pjsip.conf

identify

Identifies endpoints via source IP address

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`endpoint`	`String`		`false`	Name of Endpoint
`match`	`Custom`		`false`	IP addresses or networks to match against
`type`	`None`		`false`	Must be of type 'identify'.

Configuration Option Descriptions

match

The value is a comma-delimited list of IP addresses. IP addresses may have a subnet mask appended. The subnet mask may be written in either CIDR or dot-decimal notation. Separate the IP address and subnet mask with a slash ('/')

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_pjsip_notify

Module that supports sending NOTIFY requests to endpoints from external sources

This configuration documentation is for functionality provided by `res_pjsip_notify`.

pjsip_notify.conf

general

Unused, but reserved.

notify

Configuration of a NOTIFY request.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`.*`	`Custom`		`true`	A key/value pair to add to a NOTIFY request.

Configuration Option Descriptions

.*

If the key is `Content`, it will be treated as part of the message body. Otherwise, it will be added as a header in the NOTIFY request.

The following headers are reserved and cannot be specified:

- `Call-ID`
- `Contact`
- `CSeq`
- `To`
- `From`
- `Record-Route`
- `Route`
- `Via`

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_pjsip_outbound_publish

SIP resource for outbound publish

This configuration documentation is for functionality provided by `res_pjsip_outbound_publish`.

Overview

Outbound Publish

This module allows `res_pjsip` to publish to other SIP servers.

pjsip.conf

outbound-publish

The configuration for outbound publish

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`expiration`	`Unsigned Integer`	`3600`	`false`	Expiration time for publications in seconds
`outbound_auth`	`Custom`		`false`	Authentication object to be used for outbound publishes.
`outbound_proxy`	`String`		`false`	SIP URI of the outbound proxy used to send publishes
`server_uri`	`String`		`false`	SIP URI of the server and entity to publish to
`from_uri`	`String`		`false`	SIP URI to use in the From header
`to_uri`	`String`		`false`	SIP URI to use in the To header
`event`	`String`		`false`	Event type of the PUBLISH.
`max_auth_attempts`	`Unsigned Integer`	`5`	`false`	Maximum number of authentication attempts before stopping the publication.
`type`	`None`		`false`	Must be of type 'outbound-publish'.

Configuration Option Descriptions

server_uri

This is the URI at which to find the entity and server to send the outbound PUBLISH to. This URI is used as the request URI of the outbound PUBLISH request from Asterisk.

from_uri

This is the URI that will be placed into the From header of outgoing PUBLISH messages. If no URI is specified then the URI provided in `server_uri` will be used.

to_uri

This is the URI that will be placed into the To header of outgoing PUBLISH messages. If no URI is specified then the URI provided in `server_uri` will be used.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_pjsip_outbound_registration

SIP resource for outbound registrations

This configuration documentation is for functionality provided by `res_pjsip_outbound_registration`.

Overview

Outbound Registration

This module allows `res_pjsip` to register to other SIP servers.

pjsip.conf

registration

The configuration for outbound registration

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
auth_rejection_permanent	Boolean	yes	false	Determines whether failed authentication challenges are treated as permanent failures.
client_uri	String		false	Client SIP URI used when attemping outbound registration
contact_user	String		false	Contact User to use in request
expiration	Unsigned Integer	3600	false	Expiration time for registrations in seconds
max_retries	Unsigned Integer	10	false	Maximum number of registration attempts.
outbound_auth	Custom		false	Authentication object to be used for outbound registrations.
outbound_proxy	String		false	Outbound Proxy used to send registrations
retry_interval	Unsigned Integer	60	false	Interval in seconds between retries if outbound registration is unsuccessful
forbidden_retry_interval	Unsigned Integer	0	false	Interval used when receiving a 403 Forbidden response.
server_uri	String		false	SIP URI of the server to register against
transport	String		false	Transport used for outbound authentication
type	None		false	Must be of type 'registration'.
support_path	Boolean	no	false	Enables Path support for outbound REGISTER requests.

Configuration Option Descriptions

auth_rejection_permanent

If this option is enabled and an authentication challenge fails, registration will not be attempted again until the configuration is reloaded.

client_uri

This is the address-of-record for the outbound registration (i.e. the URI in the To header of the REGISTER).

For registration with an ITSP, the client SIP URI may need to consist of an account name or number and the provider's hostname for their registrar, e.g. client_uri=1234567890@example.com. This may differ between providers.

For registration to generic registrars, the client SIP URI will depend on networking specifics and configuration of the registrar.

forbidden_retry_interval

If a 403 Forbidden is received, chan_pjsip will wait *forbidden_retry_interval* seconds before attempting registration again. If 0 is specified, chan_pjsip will not retry after receiving a 403 Forbidden response. Setting this to a non-zero value goes against a "SHOULD NOT" in RFC3261, but can be used to work around buggy registrars.

server_uri

This is the URI at which to find the registrar to send the outbound REGISTER. This URI is used as the request URI of the outbound REGISTER request from Asterisk.

For registration with an ITSP, the setting may often be just the domain of the registrar, e.g. sip:sip.example.com.

transport

> **Note**
> A *transport* configured in `pjsip.conf`. As with other `res_pjsip` modules, this will use the first available transport of the appropriate type if unconfigured.

support_path

When this option is enabled, outbound REGISTER requests will advertise support for Path headers so that intervening proxies can add to the Path header as necessary.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_pjsip_publish_asterisk

SIP resource for inbound and outbound Asterisk event publications

This configuration documentation is for functionality provided by `res_pjsip_publish_asterisk`.

Overview

Inbound and outbound Asterisk event publication

This module allows `res_pjsip` to send and receive Asterisk event publications.

pjsip.conf

asterisk-publication

The configuration for inbound Asterisk event publication

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`devicestate_publish`	`String`		`false`	Optional name of a publish item that can be used to publish a request for full device state information.
`mailboxstate_publish`	`String`		`false`	Optional name of a publish item that can be used to publish a request for full mailbox state information.
`device_state`	`Boolean`	`no`	`false`	Whether we should permit incoming device state events.
`device_state_filter`	`Custom`		`false`	Optional regular expression used to filter what devices we accept events for.
`mailbox_state`	`Boolean`	`no`	`false`	Whether we should permit incoming mailbox state events.
`mailbox_state_filter`	`Custom`		`false`	Optional regular expression used to filter what mailboxes we accept events for.
`type`	`None`		`false`	Must be of type 'asterisk-publication'.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_pjsip_pubsub

Module that implements publish and subscribe support.

This configuration documentation is for functionality provided by `res_pjsip_pubsub`.

pjsip.conf

subscription_persistence

Persists SIP subscriptions so they survive restarts.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`packet`	`String`		`false`	Entire SIP SUBSCRIBE packet that created the subscription
`src_name`	`String`		`false`	The source address of the subscription
`src_port`	`Unsigned Integer`	`0`	`false`	The source port of the subscription
`transport_key`	`String`	`0`	`false`	The type of transport the subscription was received on
`local_name`	`String`		`false`	The local address the subscription was received on
`local_port`	`Unsigned Integer`	`0`	`false`	The local port the subscription was received on
`cseq`	`Unsigned Integer`	`0`	`false`	The sequence number of the next NOTIFY to be sent
`tag`	`Custom`		`false`	The local tag of the dialog for the subscription
`endpoint`	`Custom`		`false`	The name of the endpoint that subscribed
`expires`	`Custom`		`false`	The time at which the subscription expires

resource_list

Resource list configuration parameters.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`type`	`None`		`false`	Must be of type 'resource_list'
`event`	`String`		`false`	The SIP event package that the list resource belong to.
`list_item`	`Custom`		`false`	The name of a resource to report state on
`full_state`	`Boolean`	`no`	`false`	Indicates if the entire list's state should be sent out.

`notification_batch_interval`	`Unsigned Integer`	`0`	`false`	Time Asterisk should wait, in milliseconds, before sending notifications.

Configuration Option Descriptions

event

The SIP event package describes the types of resources that Asterisk reports the state of.

- `presence` - Device state and presence reporting.
- `message-summary` - Message-waiting indication (MWI) reporting.

list_item

In general Asterisk looks up list items in the following way:

1. Check if the list item refers to another configured resource list.

2. Pass the name of the resource off to event-package-specific handlers to find the specified resource.

The second part means that the way the list item is specified depends on what type of list this is. For instance, if you have the *event* set to `presence`, then list items should be in the form of dialplan_extension@dialplan_context. For `message-summary` mailbox names should be listed.

full_state

If this option is enabled, and a resource changes state, then Asterisk will construct a notification that contains the state of all resources in the list. If the option is disabled, Asterisk will construct a notification that only contains the states of resources that have changed.

> **Note**
> Even with this option disabled, there are certain situations where Asterisk is forced to send a notification with the states of all resources in the list. When a subscriber renews or terminates its subscription to the list, Asterisk MUST send a full state notification.

notification_batch_interval

When a resource's state changes, it may be desired to wait a certain amount before Asterisk sends a notification to subscribers. This allows for other state changes to accumulate, so that Asterisk can communicate multiple state changes in a single notification instead of rapidly sending many notifications.

inbound-publication

The configuration for inbound publications

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`endpoint`	`Custom`		`false`	Optional name of an endpoint that is only allowed to publish to this resource
`type`	`None`		`false`	Must be of type 'inbound-publication'.

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_statsd

Statsd client.

This configuration documentation is for functionality provided by `res_statsd`.

statsd.conf

global

Global configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`enabled`	`Boolean`	`no`	`false`	Enable/disable the statsd module
`server`	`IP Address`	`127.0.0.1`	`false`	Address of the statsd server
`prefix`	`String`		`false`	Prefix to prepend to every metric
`add_newline`	`Boolean`	`no`	`false`	Append a newline to every event. This is useful if you want to fake out a server using netcat (nc -lu 8125)

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_res_xmpp

XMPP Messaging

This configuration documentation is for functionality provided by `res_xmpp`.

xmpp.conf

global

Global configuration settings

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`debug`	`Custom`	`no`	`false`	Enable/disable XMPP message debugging
`autoprune`	`Custom`	`no`	`false`	Auto-remove users from buddy list.
`autoregister`	`Custom`	`yes`	`false`	Auto-register users from buddy list
`collection_nodes`	`Custom`	`no`	`false`	Enable support for XEP-0248 for use with distributed device state
`pubsub_autocreate`	`Custom`	`no`	`false`	Whether or not the PubSub server supports/is using auto-create for nodes
`auth_policy`	`Custom`	`accept`	`false`	Whether to automatically accept or deny users' subscription requests

Configuration Option Descriptions

autoprune

Auto-remove users from buddy list. Depending on the setup (e.g., using your personal Gtalk account for a test) this could cause loss of the contact list.

client

Configuration options for an XMPP client

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`username`	`String`		`false`	XMPP username with optional resource
`secret`	`String`		`false`	XMPP password
`serverhost`	`String`		`false`	Route to server, e.g. talk.google.com
`statusmessage`	`String`	`Online and Available`	`false`	Custom status message
`pubsub_node`	`String`		`false`	Node for publishing events via PubSub
`context`	`String`	`default`	`false`	Dialplan context to send incoming messages to
`priority`	`Unsigned Integer`	`1`	`false`	XMPP resource priority
`port`	`Unsigned Integer`	`5222`	`false`	XMPP server port

timeout	Unsigned Integer	5	false	Timeout in seconds to hold incoming messages
debug	Custom	no	false	Enable debugging
type	Custom	client	false	Connection is either a client or a component
distribute_events	Custom	no	false	Whether or not to distribute events using this connection
usetls	Custom	yes	false	Whether to use TLS for the connection or not
usesasl	Custom	yes	false	Whether to use SASL for the connection or not
forceoldssl	Custom	no	false	Force the use of old-style SSL for the connection
keepalive	Custom	yes	false	If enabled, periodically send an XMPP message from this client with an empty message
autoprune	Custom	no	false	Auto-remove users from buddy list.
autoregister	Custom	yes	false	Auto-register users bfrom buddy list
auth_policy	Custom	accept	false	Whether to automatically accept or deny users' subscription requests
sendtodialplan	Custom	no	false	Send incoming messages into the dialplan
status	Custom	available	false	Default XMPP status for the client
buddy	Custom		false	Manual addition of buddy to list

Configuration Option Descriptions

timeout

Timeout (in seconds) on the message stack. Messages stored longer than this value will be deleted by Asterisk. This option applies to incoming messages only which are intended to be processed by the `JABBER_RECEIVE` dialplan function.

autoprune

Auto-remove users from buddy list. Depending on the setup (e.g., using your personal Gtalk account for a test) this could cause loss of the contact list.

status

Can be one of the following XMPP statuses:

- `chat`
- `available`
- `away`
- `xaway`
- `dnd`

buddy

Manual addition of buddy to the buddy list. For distributed events, these budies are automatically added in the whitelist as 'owners' of the node(s).

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420538

Asterisk 13 Configuration_stasis

This configuration documentation is for functionality provided by `stasis`.

stasis.conf

declined_message_types

Stasis message types for which to decline creation.

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`decline`	`Custom`		`false`	The message type to decline.

Configuration Option Descriptions

decline

This configuration option defines the name of the Stasis message type that Asterisk is forbidden from creating and can be specified as many times as necessary to achieve the desired result.

- `stasis_app_recording_snapshot_type`
- `stasis_app_playback_snapshot_type`
- `stasis_test_message_type`
- `confbridge_start_type`
- `confbridge_end_type`
- `confbridge_join_type`
- `confbridge_leave_type`
- `confbridge_start_record_type`
- `confbridge_stop_record_type`
- `confbridge_mute_type`
- `confbridge_unmute_type`
- `confbridge_talking_type`
- `cel_generic_type`
- `ast_bridge_snapshot_type`
- `ast_bridge_merge_message_type`
- `ast_channel_entered_bridge_type`
- `ast_channel_left_bridge_type`
- `ast_blind_transfer_type`
- `ast_attended_transfer_type`
- `ast_endpoint_snapshot_type`
- `ast_endpoint_state_type`
- `ast_device_state_message_type`
- `ast_test_suite_message_type`
- `ast_mwi_state_type`
- `ast_mwi_vm_app_type`
- `ast_format_register_type`
- `ast_format_unregister_type`
- `ast_manager_get_generic_type`
- `ast_parked_call_type`
- `ast_channel_snapshot_type`
- `ast_channel_dial_type`
- `ast_channel_varset_type`
- `ast_channel_hangup_request_type`
- `ast_channel_dtmf_begin_type`
- `ast_channel_dtmf_end_type`
- `ast_channel_hold_type`
- `ast_channel_unhold_type`
- `ast_channel_chanspy_start_type`
- `ast_channel_chanspy_stop_type`
- `ast_channel_fax_type`
- `ast_channel_hangup_handler_type`
- `ast_channel_moh_start_type`
- `ast_channel_moh_stop_type`
- `ast_channel_monitor_start_type`
- `ast_channel_monitor_stop_type`

- ast_channel_agent_login_type
- ast_channel_agent_logoff_type
- ast_channel_talking_start
- ast_channel_talking_stop
- ast_security_event_type
- ast_named_acl_change_type
- ast_local_bridge_type
- ast_local_optimization_begin_type
- ast_local_optimization_end_type
- stasis_subscription_change_type
- ast_multi_user_event_type
- stasis_cache_clear_type
- stasis_cache_update_type
- ast_network_change_type
- ast_system_registry_type
- ast_cc_available_type
- ast_cc_offertimerstart_type
- ast_cc_requested_type
- ast_cc_requestacknowledged_type
- ast_cc_callerstopmonitoring_type
- ast_cc_callerstartmonitoring_type
- ast_cc_callerrecalling_type
- ast_cc_recallcomplete_type
- ast_cc_failure_type
- ast_cc_monitorfailed_type
- ast_presence_state_message_type
- ast_rtp_rtcp_sent_type
- ast_rtp_rtcp_received_type
- ast_call_pickup_type
- aoc_s_type
- aoc_d_type
- aoc_e_type
- dahdichannel_type
- mcid_type
- session_timeout_type
- cdr_read_message_type
- cdr_write_message_type
- cdr_prop_write_message_type
- corosync_ping_message_type
- agi_exec_start_type
- agi_exec_end_type
- agi_async_start_type
- agi_async_exec_type
- agi_async_end_type
- queue_caller_join_type
- queue_caller_leave_type
- queue_caller_abandon_type
- queue_member_status_type
- queue_member_added_type
- queue_member_removed_type
- queue_member_pause_type
- queue_member_penalty_type
- queue_member_ringinuse_type
- queue_agent_called_type
- queue_agent_connect_type
- queue_agent_complete_type
- queue_agent_dump_type
- queue_agent_ringnoanswer_type
- meetme_join_type
- meetme_leave_type
- meetme_end_type
- meetme_mute_type
- meetme_talking_type
- meetme_talk_request_type
- appcdr_message_type
- forkcdr_message_type
- cdr_sync_message_type

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717

Asterisk 13 Configuration_udptl

This configuration documentation is for functionality provided by `udptl`.

udptl.conf

global

Global options for configuring UDPTL

Configuration Option Reference

Option Name	Type	Default Value	Regular Expression	Description
`udptlstart`	`Unsigned Integer`	`4000`	`false`	The start of the UDPTL port range
`udptlend`	`Unsigned Integer`	`4999`	`false`	The end of the UDPTL port range
`udptlchecksums`	`Boolean`	`yes`	`false`	Whether to enable or disable UDP checksums on UDPTL traffic
`udptlfecentries`	`Unsigned Integer`		`false`	The number of error correction entries in a UDPTL packet
`udptlfecspan`	`Unsigned Integer`		`false`	The span over which parity is calculated for FEC in a UDPTL packet
`use_even_ports`	`Boolean`	`no`	`false`	Whether to only use even-numbered UDPTL ports
`t38faxudpec`	`Custom`		`false`	Removed
`t38faxmaxdatagram`	`Custom`		`false`	Removed

Import Version

This documentation was imported from Asterisk Version SVN-branch-13-r420717