Asterisk Documentation

Asterisk Development Team <asteriskteam@digium.com>

1. New in 10 .................................................................................................................. 8
2. Asterisk 10 Command Reference ............................................................................ 11
  2.1 Asterisk 10 AGI Commands .................................................................................. 11
     2.1.1 Asterisk 10 AGICommand_ANSWER ......................................................... 11
     2.1.2 Asterisk 10 AGICommand_ASYNCAGI BREAK ........................................... 12
     2.1.3 Asterisk 10 AGICommand_CHANNEL STATUS ........................................... 12
     2.1.4 Asterisk 10 AGICommand_CONTROL STREAM FILE ............................... 13
     2.1.5 Asterisk 10 AGICommand_DATABASE DEL ................................................ 14
     2.1.6 Asterisk 10 AGICommand_DATABASE GET ............................................... 15
     2.1.7 Asterisk 10 AGICommand_DATABASE PUT ............................................... 16
     2.1.8 Asterisk 10 AGICommand_EXEC ................................................................ 16
     2.1.9 Asterisk 10 AGICommand_GET ................................................................ 17
     2.1.10 Asterisk 10 AGICommand_GET DATA ...................................................... 17
     2.1.11 Asterisk 10 AGICommand_GET FULL VARIABLE ..................................... 17
     2.1.12 Asterisk 10 AGICommand_GET OPTION ................................................ 18
     2.1.13 Asterisk 10 AGICommand_GET VARIABLE ............................................. 18
     2.1.14 Asterisk 10 AGICommand_HANGUP ......................................................... 19
     2.1.15 Asterisk 10 AGICommand_NOOP .............................................................. 20
     2.1.16 Asterisk 10 AGICommand_RECEIVE CHAR ............................................. 21
     2.1.17 Asterisk 10 AGICommand_RECEIVE TEXT .............................................. 21
     2.1.18 Asterisk 10 AGICommand_RECORD FILE ................................................. 22
     2.1.19 Asterisk 10 AGICommand_SET DATE ..................................................... 23
     2.1.20 Asterisk 10 AGICommand_SET EXTENSION .......................................... 23
     2.1.21 Asterisk 10 AGICommand_SET AUTOHANGUP ..................................... 24
     2.1.22 Asterisk 10 AGICommand_SET DATETIME ............................................. 24
     2.1.23 Asterisk 10 AGICommand_SET DIGITS .................................................. 25
     2.1.24 Asterisk 10 AGICommand_SET OPTION ............................................... 25
     2.1.25 Asterisk 10 AGICommand_SET PHONE .................................................. 26
     2.1.26 Asterisk 10 AGICommand_SET SAY ....................................................... 27
     2.1.27 Asterisk 10 AGICommand_SET SOURCE .............................................. 27
     2.1.28 Asterisk 10 AGICommand_SET SPEECH ................................................ 28
     2.1.29 Asterisk 10 AGICommand_SET TDD MODE .......................................... 28
     2.1.30 Asterisk 10 AGICommand_SET CallerID ............................................... 29
     2.1.31 Asterisk 10 AGICommand_SET CONTEXT ............................................ 29
     2.1.32 Asterisk 10 AGICommand_SET EXTENSION ........................................ 29
     2.1.33 Asterisk 10 AGICommand_SET MUSIC ................................................ 30
     2.1.34 Asterisk 10 AGICommand_SET PRIORITY .............................................. 31
     2.1.35 Asterisk 10 AGICommand_SET VOLUME ................................................ 31
     2.1.36 Asterisk 10 AGICommand_SPEECH Activate Grammar ............................. 32
     2.1.37 Asterisk 10 AGICommand_SPEECH CREATE .......................................... 32
     2.1.38 Asterisk 10 AGICommand_SPEECH DEACTIVATE Grammar .................... 33
     2.1.39 Asterisk 10 AGICommand_SPEECH DESTROY ........................................ 33
     2.1.40 Asterisk 10 AGICommand_SPEECH LOAD GRAMMAR ............................ 34
     2.1.41 Asterisk 10 AGICommand_SPEECH RECOGNIZE ..................................... 34
     2.1.42 Asterisk 10 AGICommand_SPEECH SET ................................................ 35
     2.1.43 Asterisk 10 AGICommand_SPEECH UNLOAD GRAMMAR ....................... 35
     2.1.44 Asterisk 10 AGICommand_STREAM FILE .............................................. 36
     2.1.45 Asterisk 10 AGICommand_TDD MODE .................................................. 36
     2.1.46 Asterisk 10 AGICommand_VERBOSE ...................................................... 37
     2.1.47 Asterisk 10 AGICommand_WAIT FOR DIGIT ......................................... 38
  2.2 Asterisk 10 AMI Actions ....................................................................................... 38
     2.2.1 Asterisk 10 ManagerAction_AbsoluteTimeout .......................................... 38
     2.2.2 Asterisk 10 ManagerAction_Agents ......................................................... 39
     2.2.3 Asterisk 10 ManagerAction_AgentLogoff ............................................... 39
     2.2.4 Asterisk 10 ManagerAction_AGI ............................................................... 40
     2.2.5 Asterisk 10 ManagerAction_AOCMessage .............................................. 41
     2.2.6 Asterisk 10 ManagerAction_Atxfer .......................................................... 42
     2.2.7 Asterisk 10 ManagerAction_Bridge .......................................................... 43
     2.2.8 Asterisk 10 ManagerAction_Challenge ..................................................... 44
     2.2.9 Asterisk 10 ManagerAction_ConfbridgeMute .......................................... 44
     2.2.10 Asterisk 10 ManagerAction_ConfbridgeStartRecord ............................. 45
     2.2.11 Asterisk 10 ManagerAction_ConfbridgeKick ......................................... 45
     2.2.12 Asterisk 10 ManagerAction_ConfbridgeList ........................................... 46
     2.2.13 Asterisk 10 ManagerAction_ConfbridgeListRooms .................................. 47
     2.2.14 Asterisk 10 ManagerAction_ConfbridgeLock ........................................... 47
     2.2.15 Asterisk 10 ManagerAction_ConfbridgeMute ......................................... 48
     2.2.16 Asterisk 10 ManagerAction_ConfbridgeSetSingleVideoSrc ...................... 48
     2.2.17 Asterisk 10 ManagerAction_ConfbridgeStartRecord ............................. 49
| 2.2.18 Asterisk 10 ManagerAction_ConfbridgeStopRecord | 49 |
| 2.2.19 Asterisk 10 ManagerAction_ConfbridgeUnlock | 50 |
| 2.2.20 Asterisk 10 ManagerAction_ConfbridgeUnmute | 51 |
| 2.2.21 Asterisk 10 ManagerAction_CoreSettings | 51 |
| 2.2.22 Asterisk 10 ManagerAction_CoreShowChannels | 52 |
| 2.2.23 Asterisk 10 ManagerAction_CoreStatus | 52 |
| 2.2.24 Asterisk 10 ManagerAction_CreateConfig | 53 |
| 2.2.25 Asterisk 10 ManagerAction_DAHDIDialOffhook | 53 |
| 2.2.26 Asterisk 10 ManagerAction_DAHDIDNDoff | 54 |
| 2.2.27 Asterisk 10 ManagerAction_DAHDIDNDon | 55 |
| 2.2.28 Asterisk 10 ManagerAction_DAHDIHangup | 55 |
| 2.2.29 Asterisk 10 ManagerAction_DAHDIRestart | 56 |
| 2.2.30 Asterisk 10 ManagerAction_DAHDIShowChannels | 56 |
| 2.2.31 Asterisk 10 ManagerAction_DAHDITransfer | 57 |
| 2.2.32 Asterisk 10 ManagerAction_DataGet | 58 |
| 2.2.33 Asterisk 10 ManagerAction_DBDelete | 58 Del |
| 2.2.34 Asterisk 10 ManagerAction_DBDeleteTree | 59 |
| 2.2.35 Asterisk 10 ManagerAction_DBGet | 59 |
| 2.2.36 Asterisk 10 ManagerAction_DBPut | 60 |
| 2.2.37 Asterisk 10 ManagerAction_Events | 61 |
| 2.2.38 Asterisk 10 ManagerAction_ExtensionState | 61 |
| 2.2.39 Asterisk 10 ManagerAction_Filter | 62 |
| 2.2.40 Asterisk 10 ManagerAction_FilterList | 63 filterList |
| 2.2.41 Asterisk 10 ManagerAction_GetConfig | 63 |
| 2.2.42 Asterisk 10 ManagerAction_GetConfigJSON | 64 |
| 2.2.43 Asterisk 10 ManagerAction_Getvar | 64 |
| 2.2.44 Asterisk 10 ManagerAction_Hangup | 65 |
| 2.2.45 Asterisk 10 ManagerAction_IAXnetstats | 66 |
| 2.2.46 Asterisk 10 ManagerAction_IAXpeerlist | 66 |
| 2.2.47 Asterisk 10 ManagerAction_IAXpeers | 67 |
| 2.2.48 Asterisk 10 ManagerAction_IAXregistry | 67 |
| 2.2.49 Asterisk 10 ManagerAction_JabberSend | 68 |
| 2.2.50 Asterisk 10 ManagerAction_ListCategories | 68 |
| 2.2.51 Asterisk 10 ManagerAction_ListCommands | 69 |
| 2.2.52 Asterisk 10 ManagerAction_LocalOptimizeAway | 70 |
| 2.2.53 Asterisk 10 ManagerAction_Login | 70 |
| 2.2.54 Asterisk 10 ManagerAction_Logoff | 71 |
| 2.2.55 Asterisk 10 ManagerAction_MailboxCount | 71 |
| 2.2.56 Asterisk 10 ManagerAction_MailboxStatus | 72 |
| 2.2.57 Asterisk 10 ManagerAction_MeetmeList | 73 |
| 2.2.58 Asterisk 10 ManagerAction_MeetmeListRooms | 74 |
| 2.2.59 Asterisk 10 ManagerAction_MeetmeMute | 74 |
| 2.2.60 Asterisk 10 ManagerAction_MeetmeUnmute | 75 |
| 2.2.61 Asterisk 10 ManagerAction_MonitorMute | 75 |
| 2.2.62 Asterisk 10 ManagerAction_ModuleCheck | 76 |
| 2.2.63 Asterisk 10 ManagerAction_ModuleLoad | 76 |
| 2.2.64 Asterisk 10 ManagerAction_Monitor | 77 |
| 2.2.65 Asterisk 10 ManagerAction_Origin | 78 |
| 2.2.66 Asterisk 10 ManagerAction_Park | 79 |
| 2.2.67 Asterisk 10 ManagerAction_ParkedCalls | 80 |
| 2.2.68 Asterisk 10 ManagerAction_PauseMonitor | 80 |
| 2.2.69 Asterisk 10 ManagerAction_PlayDTMF | 81 |
| 2.2.70 Asterisk 10 ManagerAction_PlayFiles | 82 |
| 2.2.71 Asterisk 10 ManagerAction_PRIShowSpans | 82 |
| 2.2.72 Asterisk 10 ManagerAction_QueueAdd | 83 |
| 2.2.73 Asterisk 10 ManagerAction_QueueLog | 84 |
| 2.2.74 Asterisk 10 ManagerAction_QueuePause | 84 |
| 2.2.75 Asterisk 10 ManagerAction_QueuePenalty | 85 |
| 2.2.76 Asterisk 10 ManagerAction_QueueReload | 86 |
| 2.2.77 Asterisk 10 ManagerAction_QueueRemove | 86 |
| 2.2.78 Asterisk 10 ManagerAction_QueueReset | 87 |
| 2.2.79 Asterisk 10 ManagerAction_QueueRule | 87 |
| 2.2.80 Asterisk 10 ManagerAction_QueueStats | 88 |
| 2.2.81 Asterisk 10 ManagerAction_QueueStatus | 88 |
| 2.2.82 Asterisk 10 ManagerAction_QueueSummary | 89 |
| 2.2.83 Asterisk 10 ManagerAction_Redirect | 89 |
| 2.2.84 Asterisk 10 ManagerAction_Reload | 90 |
| 2.2.85 Asterisk 10 ManagerAction_SendText | 91 |
| 2.2.86 Asterisk 10 ManagerAction_Setvar | 91 |
| 2.2.87 Asterisk 10 ManagerAction_ShowDialPlan | 92 |
| 2.2.88 Asterisk 10 ManagerAction_SIPnotify | 93 |
| 2.2.89 Asterisk 10 ManagerAction_SIPpeers | 94 |
| 2.2.90 Asterisk 10 ManagerAction_SIPqualifypeer | 94 |
| 2.2.91 Asterisk 10 ManagerAction_SIPshowpeer | 95 |
| 2.2.92 Asterisk 10 ManagerAction_SIPshowregistry | 95 |
| 2.2.93 Asterisk 10 ManagerAction_SKINNYdevices | 96 |
| 2.2.94 Asterisk 10 ManagerAction_SKINNYlines | 96 |
| 2.2.95 Asterisk 10 ManagerAction_SKINNYshowdevice | 97 |
| 2.2.96 Asterisk 10 ManagerAction_SKINNYshowline | 98 |
| 2.2.97 Asterisk 10 ManagerAction_Status | 98 |
| 2.2.98 Asterisk 10 ManagerAction_StopMonitor | 99 |
| 2.2.99 Asterisk 10 ManagerAction_UnpauseMonitor | 99 |
| 2.2.100 Asterisk 10 ManagerAction_UpdateConfig | 100 |
| 2.2.101 Asterisk 10 ManagerAction_UserEvent | 101 |
| 2.2.102 Asterisk 10 ManagerAction_VoicemailUsersList | 101 |
| 2.2.103 Asterisk 10 ManagerAction_WaitEvent | 102 |
| 2.3 Asterisk 10 Dialplan Applications | 103 |
| 2.3.1 Asterisk 10 Application_AddQueueMember | 103 |
| 2.3.2 Asterisk 10 Application_ADSIProg | 104 |
| 2.3.3 Asterisk 10 Application_AELSub | 104 |
| 2.3.4 Asterisk 10 Application_AgentLogin | 105 |
| 2.3.5 Asterisk 10 Application_AgentMonitorOutgoing | 106 |
| 2.3.6 Asterisk 10 Application_AGI | 106 |
| 2.3.7 Asterisk 10 Application_AlarmReceiver | 107 |
| 2.3.8 Asterisk 10 Application_AMD | 108 |
| 2.3.9 Asterisk 10 Application_Answer | 109 |
| 2.3.10 Asterisk 10 Application_Authenticate | 110 |
| 2.3.11 Asterisk 10 Application_Background | 111 |
| 2.3.12 Asterisk 10 Application_BackgroundDetect | 112 |
| 2.3.13 Asterisk 10 Application_Bridge | 112 |
| 2.3.14 Asterisk 10 Application_Busy | 113 |
| 2.3.15 Asterisk 10 Application_CallCompletionCancel | 114 |
| 2.3.16 Asterisk 10 Application_CallCompletionRequest | 114 |
| 2.3.17 Asterisk 10 Application_CEILogUserEvent | 115 |
| 2.3.18 Asterisk 10 Application_CEILogMonitor | 116 |
| 2.3.19 Asterisk 10 Application_ChannelBusy | 116 |
| 2.3.20 Asterisk 10 Application_ChannelRedirect | 117 |
| 2.3.21 Asterisk 10 Application_ChantSpy | 118 |
| 2.3.22 Asterisk 10 Application_ClearHash | 119 |
| 2.3.23 Asterisk 10 Application_ConfBridge | 119 |
| 2.3.24 Asterisk 10 Application_Congestion | 120 |
| 2.3.25 Asterisk 10 Application_ContinueWhile | 121 |
| 2.3.26 Asterisk 10 Application_ContinueWhen | 121 |
| 2.3.27 Asterisk 10 Application_DAHDIAcceptR2Call | 122 |
| 2.3.28 Asterisk 10 Application_DAHDIBarge | 123 |
| 2.3.29 Asterisk 10 Application_DAHDIRAS | 123 |
| 2.3.30 Asterisk 10 Application_DAHDIScan | 124 |
| 2.3.31 Asterisk 10 Application_DAHDISendCallroutingFacility | 124 |
| 2.3.32 Asterisk 10 Application_DAHDISendKeypadFacility | 125 |
| 2.3.33 Asterisk 10 Application_DDAgile | 126 |
| 2.3.34 Asterisk 10 Application_DBdel | 126 |
| 2.3.35 Asterisk 10 Application_DBdeltree | 127 |
| 2.3.36 Asterisk 10 Application_DeadAGI | 127 |
| 2.3.37 Asterisk 10 Application_Dial | 128 |
| 2.3.38 Asterisk 10 Application_Dictate | 131 |
| 2.3.39 Asterisk 10 Application_Directory | 132 |
| 2.3.40 Asterisk 10 Application_DiskMount | 133 |
| 2.3.41 Asterisk 10 Application_DumpChan | 134 |
| 2.3.42 Asterisk 10 Application_EAGI | 135 |
| 2.3.43 Asterisk 10 Application_Echo | 136 |
| 2.3.44 Asterisk 10 Application_EndWhile | 136 |
| 2.3.45 Asterisk 10 Application_Exec | 137 |
| 2.3.46 Asterisk 10 Application_ExecIf | 137 |
| 2.3.47 Asterisk 10 Application_ExitWhile | 138 |
| 2.3.48 Asterisk 10 Application_ExitWhen | 139 |
| 2.3.49 Asterisk 10 Application_ExtenSpy | 139 |
| 2.3.50 Asterisk 10 Application_ExternalIVR | 141 |
| 2.3.51 Asterisk 10 Application_Festival | 141 |
| 2.3.52 Asterisk 10 Application_Flash | 142 |
| 2.3.53 Asterisk 10 Application_FollowMe | 143 |
| 2.3.54 Asterisk 10 Application_ForkCDR | 143 |
| 2.3.55 Asterisk 10 Application_GetCEID | 145 |
| 2.3.56 Asterisk 10 Application_Gosub | 146 |
| 2.3.57 Asterisk 10 Application_GosubIf | 146 |
| 2.3.58 Asterisk 10 Application_Goto | 147 |
| 2.3.59 Asterisk 10 Application_Gotol | 148 |
| 2.3.60 Asterisk 10 Application_GotolIf | 149 |
| 2.3.61 Asterisk 10 Application_Hangup | 150 |
| 2.3.62 Asterisk 10 Application_IAX2Provision | 150 |
| 2.3.63 Asterisk 10 ApplicationICES | 151 |
2.3.139 Asterisk 10 Application_SayDigits .................................................. 209
2.3.140 Asterisk 10 Application_SayNumber .................................................. 209
2.3.141 Asterisk 10 Application_Say Phonetic .............................................. 210
2.3.142 Asterisk 10 Application_SayUnixTime ............................................. 211
2.3.143 Asterisk 10 Application_SendDTMF .................................................. 211
2.3.144 Asterisk 10 Application_SendFAX (app_fax) ..................................... 212
2.3.145 Asterisk 10 Application_SendFAX (res_fax) ..................................... 213
2.3.149 Asterisk 10 Application_SendImage .................................................. 214
2.3.147 Asterisk 10 Application_SendText .................................................... 214
2.3.148 Asterisk 10 Application_SendURL .................................................... 215
2.3.149 Asterisk 10 Application_Set ............................................................. 216
2.3.150 Asterisk 10 Application_SetMAFFlags ............................................. 217
2.3.151 Asterisk 10 Application_SetCallerPres ........................................... 217
2.3.152 Asterisk 10 Application_SetMusicOnHold ....................................... 218
2.3.153 Asterisk 10 Application_SIPAddHeader .......................................... 219
2.3.154 Asterisk 10 Application_SIPDMFMode ............................................. 219
2.3.155 Asterisk 10 Application_SIPRemoveHeader ...................................... 220
2.3.156 Asterisk 10 Application_Skel .......................................................... 221
2.3.157 Asterisk 10 Application_SLASStation ............................................. 221
2.3.158 Asterisk 10 Application_SLATrunk .................................................. 222
2.3.159 Asterisk 10 Application_SMS .......................................................... 223
2.3.160 Asterisk 10 Application_SoftHangup .............................................. 224
2.3.161 Asterisk 10 Application_SpeechActivateGrammar ................................ 224
2.3.162 Asterisk 10 Application_SpeechBackground ..................................... 225
2.3.163 Asterisk 10 Application_SpeechCreate ........................................... 226
2.3.164 Asterisk 10 Application_SpeechDeactivateGrammar ........................... 226
2.3.165 Asterisk 10 Application_SpeechDestroy .......................................... 227
2.3.166 Asterisk 10 Application_SpeechLoadGrammar ................................... 227
2.3.167 Asterisk 10 Application_SpeechProcessingSound .............................. 228
2.3.168 Asterisk 10 Application_SpeechReset ............................................. 228
2.3.169 Asterisk 10 Application_SpeechUnloadGrammar ................................ 229
2.3.170 Asterisk 10 Application_StackPop ................................................... 229
2.3.171 Asterisk 10 Application_StartMusicOnHold ...................................... 230
2.3.172 Asterisk 10 Application_StopMixMonitor ........................................ 230
2.3.173 Asterisk 10 Application_StopMonitor ............................................. 231
2.3.174 Asterisk 10 Application_StopMusicOnHold ...................................... 231
2.3.175 Asterisk 10 Application_StopPlayTones .......................................... 232
2.3.176 Asterisk 10 Application_System ..................................................... 232
2.3.177 Asterisk 10 Application_TestClient ................................................ 233
2.3.178 Asterisk 10 Application_TestServer .............................................. 234
2.3.179 Asterisk 10 Application_Transfer ................................................... 234
2.3.180 Asterisk 10 Application_TryExec .................................................... 235
2.3.181 Asterisk 10 Application_TrySystem ................................................ 236
2.3.182 Asterisk 10 Application_UnpauseMonitor ....................................... 236
2.3.183 Asterisk 10 Application_VerbResponse .......................................... 237
2.3.184 Asterisk 10 Application_UserEvent ................................................ 238
2.3.185 Asterisk 10 Application_Verbose .................................................... 239
2.3.186 Asterisk 10 Application_VMAuthenticate ......................................... 239
2.3.187 Asterisk 10 Application_VMSayName .............................................. 240
2.3.188 Asterisk 10 Application_VoiceMail ................................................ 240
2.3.189 Asterisk 10 Application_VoiceMailMain ........................................ 242
2.3.190 Asterisk 10 Application_VoiceMailQueue ...................................... 243
2.3.191 Asterisk 10 Application_WaitExten ............................................... 243
2.3.192 Asterisk 10 Application_WaitForNoise ......................................... 244
2.3.193 Asterisk 10 Application_WaitForRing ............................................ 244
2.3.194 Asterisk 10 Application_WaitForSilence ....................................... 245
2.3.195 Asterisk 10 Application_WaitMusicOnHold .................................... 245
2.3.196 Asterisk 10 Application_WaitUntil ................................................ 247
2.3.197 Asterisk 10 Application_WaitUnint .............................................................................. 247
2.3.198 Asterisk 10 Application_Zapatriello .............................................. 248

2.4 Asterisk 10 Dialplan Functions ............................................................... 249
2.4.1 Asterisk 10 Function_AES_DECYPRT .................................................. 249
2.4.2 Asterisk 10 Function_AES_ENCRYPT .................................................. 249
2.4.3 Asterisk 10 Function_AGCC ................................................................. 250
2.4.4 Asterisk 10 Function_AGENT ............................................................... 251
2.4.5 Asterisk 10 Function.ARRAY ............................................................... 251
2.4.6 Asterisk 10 Function_AST_CONFIG .................................................... 252
2.4.7 Asterisk 10 Function_AudioHOOK_INHERIT ....................................... 252
2.4.8 Asterisk 10 Function_BASE64_DECRYPT .......................................... 252
2.4.9 Asterisk 10 Function_BASE64_ENCODE ............................................. 254
2.4.10 Asterisk 10 Function_BLACKLIST ................................................... 254
2.4.11 Asterisk 10 Function_CALENDAR_BUSY ......................................... 255
2.4.12 Asterisk 10 Function_CALENDAR_EVENT ....................................... 256
2.4.13 Asterisk 10 Function_CALENDAR_QUERY ....................................... 257
2.4.14 Asterisk 10 Function_CALENDAR_QUERY_RESULT ............................. 258
New in 10

- Overview
  - In Brief
  - Detailed Listing
    - Text Messaging
    - Parking
    - Asterisk Manager Interface
    - Asterisk HTTP Server
    - chan_dahdi
    - CLI Changes
    - CDR
    - CODECS
    - ConfBridge
    - Dialplan Variables
    - Dialplan Functions
    - libpri channel driver (chan_dahdi) DAHDI changes
    - Calendaring
    - MixMonitor
    - FollowMe
    - CEL
    - pbx_lua
    - res_fax
    - SIP Changes
    - Queue changes
    - Applications
    - Asterisk Database
    - Asterisk Modules

Overview

A listing of new capabilities in Asterisk 10

In Brief

Asterisk 10 introduces a number of new features since the previous 1.8 release. Highlights include:

- Advanced, high-performance wide and ultra-wideband conferencing application for 8-192kHz clients
- Re-architected media negotiation framework featuring support for an array of common sampling rates
- Support for SKYPE’s SILK codec, offering narrow, wide and ultra-wideband audio
- Pass-ThroughSupport for the CELT low-latency audio codec at 32 and 48kHz
- Support for the SPEEX codec at 32kHz
- New receive-side jitter buffer capabilities
- CCSS Device State Information

Detailed Listing

Text Messaging

- Asterisk now has protocol independent support for processing text messages outside of a call. Messages are routed through the Asterisk dialplan. SIP MESSAGE and XMPP are currently supported. There are options in jabber.conf and sip.conf to allow enabling these features.
  - `jabber.conf`: see the "sendtodialplan" and "context" options.
  - `sip.conf`: see the "accept_outofcall_message", "auth_message_requests" and "outofcall_message_context" options.
- The `MESSAGE()` dialplan function and `MessageSend()` application have been added to go along with this functionality. More detailed usage information can be found on the Asterisk wiki ([http://wiki.asterisk.org/](http://wiki.asterisk.org/)).
Parking

- parkedmusicclass can now be set for non-default parking lots.
- ParkedCall application can now specify a specific parkinglot.

Asterisk Manager Interface

- PeerStatus now includes Address and Port.
- Added Hold events for when the remote party puts the call on and off hold for chan_dahdi ISDN channels.
- Added new action MeetmeListRooms to list active conferences (shows same data as "meetme list" at the CLI).
- DAHDIShowChannels, SIPshowpeer, SIPpeers, and IAXpeers now contains a Description field that is set by 'description' in the channel configuration file.
- Added Uniqueid header to UserEvent.
- Added new action FilterAdd to control event filters for the current session. This requires the system permission and uses the same filter syntax as filters that can be defined in manager.conf

Asterisk HTTP Server

- The HTTP Server can bind to IPv6 addresses.

chan_dahdi

- Busy tone patterns featuring 2 silence and 2 tone lengths can now be used with busydetect. usage example:
  
  busypattern=200,200,200,600

CLI Changes

- New ‘gtalk show settings' command showing the current settings loaded from gtalk.conf.
- The ‘logger reload' command now supports an optional argument, specifying an alternate configuration file to use.
- 'dialplan add extension' command will now automatically create a context if the specified context does not exist with a message indicated it did so.
- 'sip show peers', 'iax show peers', and 'dahdi show peers' now contains a Description field which can be populated with 'description' in the channel configuration files (sip.conf, iax2.conf, and chan_dahdi.conf).

CDR

- The filter option in cdr_adaptive_odbc now supports negating the argument, thus allowing records which do NOT match the specified filter.

CODECS

- Ability to define custom SILK formats in codecs.conf.
- Addition of speex32 audio format with translation.
- CELT codec pass-through support and ability to define custom CELT formats in codecs.conf.
- Ability to read raw signed linear files with sample rates ranging from 8khz - 192khz. The new file extensions introduced are .sln12, .sln24, .sln32, .sln44, .sln48, .sln96, .sln192.

ConfBridge

- New highly optimized and customizable ConfBridge application capable of mixing audio at sample rates ranging from 8khz-96khz.
- CONFBRIDGE dialplan function capable of creating dynamic ConfBridge user and bridge profiles on a channel.
- CONFBRIDGE_INFO dialplan function capable of retrieving information about a conference such as locked status and number of parties, admins, and marked users.
- Addition of video_mode option in confbridge.conf for adding video support into a bridge profile.
- Addition of the follow_talker video_mode in confbridge.conf. This video mode dynamically switches the video feed to always display the loudest talker supplying video in the conference.

Dialplan Variables
• Added ASTETCDIR, ASTMODDIR, ASTVARLIBDIR, ASTDBDIR, ASTKEYDIR, ASTDATADIR, ASTAGIDIR, ASTSPOOLDIR, ASTUNDIR, ASTLOGDIR which hold the equivalent variables from asterisk.conf.

**Dialplan Functions**

• Addition of the JITTERBUFFER dialplan function. This function allows for jitterbuffering to occur on the read side of a channel. By using this function conference applications such as ConfBridge and MeetMe can have the rx streams jitterbuffered before conference mixing occurs.
• Added DB_KEYS, which lists the next set of keys in the Asterisk database hierarchy.
• Added STRREPLACE function. This function let's the user search a variable for a given string to replace with another string as many times as the user specifies or just throughout the whole string.
• Added option to CHANNEL(pickupgroup) allow reading and setting the pickupgroup of channel.

**libpri channel driver (chan_dahdi) DAHDI changes**

• Added moh_signaling option to specify what to do when the channel's bridged peer puts the ISDN channel on hold.
• Added display_send and display_receive options to control how the display ie is handled. To send display text from the dialplan use the SendText() application when the option is enabled.
• Added mcid_send option to allow sending a MCID request on a span.

**Calendaring**

• Added setvar option to calendar.conf to allow setting channel variables on notification channels.
• Added "calendar show types" CLI command to list registered calendar connectors.

**MixMonitor**

• Added two new options, r and t with file name arguments to record single direction (unmixed) audio recording separate from the bidirectional (mixed) recording. The mixed file name argument is optional now as long as at least one recording option is used.

**FollowMe**

• Added a new option, l, which will disable local call optimization for channels involved with the FollowMe thread. Use this option to improve compatibility for a FollowMe call with certain dialplan apps, options, and functions.

**CEL**

• cel_pgsqI now supports the 'extra' column for data added using the CELGenUserEvent() application.

**pbx_lua**

• Support for defining hints has been added to pbx_lua. See the 'hints' table in the sample extensions.lua file for syntax details.
• Applications that perform jumps in the dialplan such as Goto will now execute properly. When pbx_lua detects that the context, extension, or priority we are executing on has changed it will immediately return control to the asterisk PBX engine. Currently the engine cannot detect a Goto to the priority after the currently executing priority.
• An autoservice is now started by default for pbx_lua channels. It can be stopped and restarted using the autoservice_stop() and autoservice_start() functions.

**res_fax**

• The ReceiveFAXStatus and SendFAXStatus manager events have been consolidated into a FAXStatus event with an 'Operation' header that will be either 'send', 'receive', or 'gateway'.
• T.38 gateway functionality has been added to res_fax (and res_fax_spandsp). Set FAXOPT(gateway)=yes to enable this functionality on a channel. This feature will handle converting a fax call between an audio T.30 fax terminal and an IFP T.38 fax terminal.

**SIP Changes**

• Add T38 support for REJECTED state where T.38 Negotiation is explicitly rejected.
Queue changes

- Added general option negative_penalty_invalid default off. When set members are seen as invalid/logged out when there penalty is negative. For realtime members when set remove from queue will set penalty to -1.
- Added queue option autopausedelay when autopause is enabled it will be delayed for this number of seconds since last successful call if there was no prior call the agent will be autopaused immediately.
- Added member option ignorebusy this when set and ringinuse is not will allow per member control of multiple calls as ringinuse does for the Queue.

Applications

- Added 'v' option to MeetMe to play voicemail greetings when a user joins/leaves a MeetMe conference
- Added ability to include '@parkinglot' to ParkedCall extension in order to specify a specific parkinglot on which to search the extension.

Asterisk Database

- The internal Asterisk database has been switched from Berkeley DB 1.86 to SQLite 3. An existing Berkeley astdb file can be converted with the astdb2sqlite3 utility in the UTILS section of menuselect. If an existing astdb is found and no astdb.sqlite3 exists, astdb2sqlite3 will be compiled automatically. Asterisk will convert an existing astdb to the SQLite3 version automatically at runtime.

Asterisk Modules

- Modules marked as deprecated are no longer marked as building by default. Enabling these modules is still available via menuselect.

Asterisk 10 Command Reference

This page is the top level page for all of the Asterisk 10 applications, functions, manager actions, and AGI commands that are kept in the XML based documentation that is included with Asterisk 10.

Asterisk 10 AGI Commands

Asterisk 10 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**

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
See Also

Asterisk 10 AGICommand_HANGUP

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 AGICommand_HANGUP

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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:
Channel is down and available.

Channel is down, but reserved.

Channel is off hook.

Digits (or equivalent) have been dialed.

Line is ringing.

Remote end is ringing.

Line is up.

Line is busy.

Syntax

```
CHANNEL STATUS [CHANNELNAME]
```

Arguments

- `CHANNELNAME`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 AGICloneCommand**

**CONTROL STREAM FILE**

Synopsis

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

Description

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

Syntax

```
CONTROL STREAM FILE FILENAME ESCAPE_DIGITS [SKIPMS] [FFCHAR] [REWCHR] [PAUSECHR]
```
Arguments

- **FILENAME** - The file extension must not be included in the filename.
- **ESCAPE_DIGITS**
- **SKIPMS**
- **FFCHAR** - Defaults to +
- **REWCHR** - Defaults to #
- **PAUSECHR**

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 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-10-r340810.

### Asterisk 10 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-10-r340810.

**Asterisk 10 AGICall Command**

**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
Asterisk 10 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-10-r340810.

Asterisk 10 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
```
Asterisk 10 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-10-r340810.

Asterisk 10 AGICommand_GET FULL VARIABLE

GET FULL VARIABLE

Synopsis

Evaluates a channel expression

Description
Returns 0 if \textit{variblename} is not set or channel does not exist. Returns 1 if \textit{variblename} is set and returns the variable in parenthesis. Understands complex variable names and builtin variables, unlike \texttt{GET VARIABLE}.

Example return code: 200 result=1 (testvariable)

\textbf{Syntax}

\begin{verbatim}
GET FULL VARIABLE \texttt{VARIBLENAME} [\texttt{CHANNEL\_NAME}]
\end{verbatim}

\textbf{Arguments}

- \texttt{VARIBLENAME}
- \texttt{CHANNEL\_NAME}

\textbf{See Also}

\texttt{Import Version}

This documentation was imported from Asterisk version SVN-branch-10-r340810.

\textbf{Asterisk 10 AGICommand\_GET OPTION}

\textbf{GET OPTION}

\textbf{Synopsis}

Stream file, prompt for DTMF, with timeout.

\textbf{Description}

Behaves similar to \texttt{STREAM FILE} but used with a timeout option.

\textbf{Syntax}

\begin{verbatim}
GET OPTION FILENAME ESCAPE\_DIGITS [TIMEOUT]
\end{verbatim}

\textbf{Arguments}

- \texttt{FILENAME}
- \texttt{ESCAPE\_DIGITS}
- \texttt{TIMEOUT}

\textbf{See Also}

\texttt{Asterisk 10 AGICommand\_STREAM FILE}

\texttt{Import Version}

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 AGICommand_GET_VARIABLE

GET VARIABLE

Synopsis

Gets a channel variable.

Description

Returns 0 if \textit{variablename} is not set. Returns 1 if \textit{variablename} is set and returns the variable in parentheses.

Example return code: 200 result=1 (testvariable)

Syntax

\begin{verbatim}
GET VARIABLE VARIABlename
\end{verbatim}

Arguments

- \textbf{VARIABlename}

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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

\begin{verbatim}
GOSUB CONTEXT EXTENSION PRIORITY [OPTIONAL-ARGUMENT]
\end{verbatim}

Arguments

- \textbf{CONTEXT}
- \textbf{EXTENSION}
Asterisk 10 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-10-r340810.

Asterisk 10 AGICommand_NOOP

NOOP

Synopsis

Does nothing.

Description

Does nothing.

Syntax


Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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**
Asterisk 10 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-10-r340810.

Asterisk 10 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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
SET AUTOHANGUP

Synopsis

Autohangup channel in some time.

Description

Cause the channel to automatically hangup at \textit{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


\texttt{SET \textbf{AUTOHANGUP} \textit{TIME}}

Arguments

- \texttt{TIME}

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

\textbf{Asterisk 10 AGICommand\_SET CALLERID}

\textbf{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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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.

Always returns 0.

Syntax

```
SET MUSIC UNNAMED_PARAMETER CLASS
```

Arguments

- `UNNAMED_PARAMETER`
  - Unnamed Option
  - Unnamed Option
- `CLASS`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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-10-r340810.

Asterisk 10 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
Asterisk 10 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-10-r340810.

Asterisk 10 AGICommand_SPEECH CREATE

**SPEECH CREATE**

**Synopsis**

Creates a speech object.

**Description**

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

**Syntax**

```
SPEECH CREATE ENGINE
```

**Arguments**

- `'ENGINE'`

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 AGICommand_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-10-r340810.

Asterisk 10 AGICommand_SPEECH DESTROY

**Synopsis**
Destroys a speech object.

**Description**
Destroy the speech object created by `SPEECH CREATE`.

**Syntax**
```
SPEECH DESTROY
```

**Arguments**
Asterisk 10 AGICommand_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-10-r340810.

Asterisk 10 AGICommand_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-10-r340810.

**Asterisk 10 AGICommand_SPEECH SET**

**SPEECH SET**

**Synopsis**

Sets a speech engine setting.

**Description**

Set an engine-specific setting.

**Syntax**

```
SPEECH SET NAME VALUE
```

Arguments

- SPEECH SET
- VALUE

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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.

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 10 AGICommand_CONTROL STREAM FILE**

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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**
Asterisk 10 AGICommand_WAIT FOR DIGIT

WAIT FOR DIGIT

Synopsis

Waits for a digit to be pressed.

Description

Waits up to \textit{timeout} milliseconds for channel to receive a DTMF digit. Returns \texttt{-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 \texttt{-1} for the \textit{timeout} value if you desire the call to block indefinitely.

Syntax

\begin{verbatim}
WAIT FOR DIGIT TIMEOUT
\end{verbatim}

Arguments

\begin{itemize}
\item \textbf{TIMEOUT}
\end{itemize}

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 AMI Actions

Asterisk 10 ManagerAction.AbsoluteTimeout

AbsoluteTimeout

Synopsis

Set absolute timeout.

Description

Hangup a channel after a certain time. Acknowledges set time with \texttt{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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 ManagerAction_Agents
Agents

Synopsis

Lists agents and their status.

Description

Will list info about all possible agents.

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.
Asterisk 10 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 entries were required this could be achieved by setting the parameter 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-10-r340810.

Asterisk 10 ManagerAction_Atxfer

Atxfer

Synopsis

Attended transfer.

Description

Attended transfer.

Syntax
Action: Atxfer
[ActionID:] <value>
Channel: <value>
Exten: <value>
Context: <value>
Priority: <value>

Arguments

- **ActionID**: ActionID for this transaction. Will be returned.
- **Channel**: Transferee’s channel.
- **Exten**: Extension to transfer to.
- **Context**: Context to transfer to.
- **Priority**: Priority to transfer to.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Manager**

**Action_Bridge**

**Bridge**

**Synopsis**

Bridge two channels already in the PBX.

**Description**

Bridge together two channels already in the PBX.

**Syntax**

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

**Arguments**

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

See Also
**Asterisk 10 Manager** Action_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-10-r340810.

**Asterisk 10 Manager** Action_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-10-r340810.

Asterisk 10 ManagerAction_Command

Command

Synopsis

Execute Asterisk CLI Command.

Description

Run a CLI command.

Syntax

```plaintext
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-10-r340810.

Asterisk 10 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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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
Asterisk 10 Manager\texttt{Action\_ConfbridgeListRooms}

\textbf{ConfbridgeListRooms}

**Synopsis**

List active conferences.

**Description**

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

**Syntax**

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

**Arguments**

- \texttt{ActionID} - ActionID for this transaction. Will be returned.

**See Also**

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Manager\texttt{Action\_ConfbridgeLock}

\textbf{ConfbridgeLock}

**Synopsis**

Lock a Confbridge conference.

**Description**

**Syntax**

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

**Arguments**
Asterisk 10 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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 ManagerAction_ConfbridgeSetSingleVideoSrc

ConfbridgeSetSingleVideoSrc

Synopsis

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

Description

Syntax
**Asterisk 10 Manager**

**Action** _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**

```plaintext
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-10-r340810.
Asterisk 10 Manager Action 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-10-r340810.

Asterisk 10 Manager Action 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
Asterisk 10 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`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 ManagerAction_CoreSettings

CoreSettings

Synopsis

Show PBX core settings (version etc).

Description

Query for Core PBX settings.

Syntax

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

Arguments

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_DAHDIDNDoff**

**DAHDIDNDoff**

**Synopsis**

Toggle DAHDI channel Do Not Disturb status OFF.

**Description**

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

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.
Asterisk 10 ManagerAction_DAHDIDNDOn

DAHDIDNDOn

Synopsis

Toggle DAHDI channel Do Not Disturb status ON.

Description

Equivalent to the CLI command "dahdi set dnd Equivalent to the CLI command "dahdi set dnd
None - channel on".

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-10-r340810.

Asterisk 10 ManagerAction_DAHDIHangup

DAHDIHangup

Synopsis

Hangup DAHDI Channel.

Description

Simulate an on-hook event by the user connected to the channel.
Valid only for analog channels. 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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 ManagerAction_DAHDITransfer**

**DAHDITransfer**

Synopsis

Transfer DAHDI Channel.

Description

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

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.
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Manager**

**Action_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**
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Manager**

**Action 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 command 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-10-r340810.

Asterisk 10 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

Action: FilterList

Arguments

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_Getvar**
Getvar

Synopsis

Gets a channel variable.

Description

Get the value of a global or local channel variable.

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.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 ManagerAction_Hangup

Hangup

Synopsis

Hangup channel.

Description

Hangup a channel.

Syntax

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

Arguments
Asterisk 10 Manager Action IAXnetstats

IAXnetstats

Synopsis

Show IAX Netstats.

Description

Show IAX channels network statistics.

Syntax

```
Action: IAXnetstats
```

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Manager Action 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-10-r340810.

**Asterisk 10 Manager**

**Action_IAXpeers**

**IAXpeers**

**Synopsis**

List IAX peers.

**Description**

**Syntax**

```plaintext
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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

Asterisk 10 ManagerAction_JabberSend

JabberSend

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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Manager**

**Action/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
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 ManagerAction_MailboxStatus

MailboxStatus

Synopsis

Check mailbox.

Description
Checks a voicemail account for status.
Returns number of messages.
Message: Mailbox Status.

Mailbox: mailboxid.

Waiting: count.

Syntax

```plaintext
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-10-r340810.

**Asterisk 10 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**

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

Arguments

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

See Also

Import Version
**Asterisk 10 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-10-r340810.

**Asterisk 10 ManagerAction_MeetmeMute**

**MeetmeMute**

**Synopsis**

Mute a Meetme user.

**Description**

**Syntax**

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

**Arguments**
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 ManagerAction_ModuleCheck**

**ModuleCheck**

**Synopsis**

Check if module is loaded.

**Description**

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

**Syntax**

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

**Arguments**

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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
  - enum
  - dnsmgr
  - extconfig
  - manager
  - rtp
  - http
- **LoadType** - The operation to be done on module. If no module is specified for a loadtype, all modules are reloaded.
  - load
  - unload
  - reload

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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>
[Async:] <value>
[Codecs:] <value>

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 ManagerAction_Park

Park

Synopsis

Park a channel.

Description

Park a channel.

Syntax
Action: Park
[ActionID:] <value>
Channel: <value>
Channel2: <value>
[Timeout:] <value>
[Parkinglot:] <value>

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 ManagerAction_ParkedCalls**

**ParkedCalls**

**Synopsis**

List parked calls.

**Description**

List parked calls.

**Syntax**

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

Arguments

- **ActionID**: ActionID for this transaction. Will be returned.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 ManagerAction_PauseMonitor**
**PauseMonitor**

**Synopsis**

Pause monitoring of a channel.

**Description**

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

**Syntax**

```plaintext
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-10-r340810.

**Asterisk 10 Manager**

**Action_Ping**

**Ping**

**Synopsis**

Keepalive command.

**Description**

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

**Syntax**

```plaintext
Action: Ping
[ActionID:] <value>
```

**Arguments**

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

**See Also**
**Asterisk 10 Manager**

**Action_PlayDTMF**

**PlayDTMF**

**Synopsis**

Play DTMF signal on a specific channel.

**Description**

Plays a dtmf digit on the specified channel.

**Syntax**

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

**Arguments**

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

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_QueueAdd**

**QueueAdd**

**Synopsis**

Add interface to queue.

**Description**

**Syntax**

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 Manager

**Action.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-10-r340810.

Asterisk 10 Manager

**Action.QueuePause**

**QueuePause**

*Synopsis*

Makes a queue member temporarily unavailable.

*Description*

*Syntax*
Action: QueuePause
[ActionID:] <value>
Interface: <value>
Paused: <value>
[Queue:] <value>
[Reason:] <value>

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 ManagerAction_QueuePenalty

QueuePenalty

Synopsis

Set the penalty for a queue member.

Description

Syntax

Action: QueuePenalty
[ActionID:] <value>
Interface: <value>
Penalty: <value>
[Queue:] <value>

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 ManagerAction_QueueReload

QueueReload

Synopsis

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

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 ManagerAction_QueueRemove

QueueRemove

Synopsis

Remove interface from queue.

Description

Syntax
Action: QueueRemove
[ActionID:] <value>
Queue: <value>
Interface: <value>

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Manager Action_QueueReset

QueueReset

Synopsis

Reset queue statistics.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Manager Action_QueueRule

QueueRule

Synopsis
Queue Rules.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 ManagerAction_Queues**

Queues

Synopsis

Queues.

Description

Syntax

```
Action: Queues
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 ManagerAction_QueueStatus**

QueueStatus

Synopsis

Show queue status.
Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 ManagerAction_QueueSummary**

**QueueSummary**

Synopsis

Show queue summary.

Description

Syntax

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

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 ManagerAction_Setvar**
Setvar

Synopsis

Set a channel variable.

Description

Set a global or local channel variable.

Syntax

Action: Setvar
[ActionID:] <value>
[Channel:] <value>
Variable: <value>
Value: <value>

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Manager

Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.
Asterisk 10 Manager

Action_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-10-r340810.

Asterisk 10 Manager

Action_SIPqualifypeer

SIPqualifypeer

Synopsis

Qualify SIP peers.

Description

Qualify a SIP peer.

Syntax

Action: SIPqualifypeer
[ActionID:] <value>
Peer: <value>

Arguments
Asterisk 10 ManagerAction_SIPshowpeer

SIPshowpeer

Synopsis

Show one SIP peer with details on current status.

Syntax

```plaintext
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-10-r340810.

Asterisk 10 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.
Asterisk 10 Manager Action_SIPshowregistry

**Action:** SIPshowregistry

**Arguments**

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

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

---

Asterisk 10 Manager Action_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-10-r340810.

---

Asterisk 10 Manager Action_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-10-r340810.

**Asterisk 10 Manager**

**Action_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-10-r340810.
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

### Asterisk 10 Manager `Action_UserEvent`

**UserEvent**

**Synopsis**

Send an arbitrary event.

**Description**

Send an event to manager sessions.

**Syntax**

```plaintext
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-10-r340810.
Asterisk 10 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-10-r340810.

Asterisk 10 ManagerAction_WaitEvent

WaitEvent

Synopsis

Wait for an event to occur.

Description

This action will elicit 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
Asterisk 10 Dialplan Applications

Asterisk 10 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 10 Application_Queue
- Asterisk 10 Application_QueueLog
- Asterisk 10 Application_AddQueueMember
- Asterisk 10 Application_RemoveQueueMember
- Asterisk 10 Application_PauseQueueMember
- Asterisk 10 Application_UnpauseQueueMember
- Asterisk 10 Function_QUEUE_VARIABLES
- Asterisk 10 Function_QUEUE_MEMBER
- Asterisk 10 Function_QUEUE_MEMBER_COUNT
- Asterisk 10 Function_QUEUE_EXISTS
**Asterisk 10 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 10 Application_GetCPEID
adsi.conf

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 Application_AgentLogin**

**AgentLogin()**

Synopsis

Call agent login.

Description

Asks the agent to login to the system. Always returns -1. While logged in, the agent can receive calls and will hear a beep when a new call comes in. The agent can dump the call by pressing the star key.

Syntax

```
AgentLogin([AgentNo[,options]])
```

Arguments

- `AgentNo`
- `options`
  - `s` - silent login - do not announce the login ok segment after agent logged on/off

See Also

**Asterisk 10 Application_Queue**
**Asterisk 10 Application_AddQueueMember**
**Asterisk 10 Application_RemoveQueueMember**
**Asterisk 10 Application_PauseQueueMember**
**Asterisk 10 Application_UnpauseQueueMember**
**Asterisk 10 Function_AGENT**
Asterisk 10 Application_AgentMonitorOutgoing

AgentMonitorOutgoing()

Synopsis

Record agent’s outgoing call.

Description

Tries to figure out the id of the agent who is placing outgoing call based on comparison of the callerid of the current interface and the global variable placed by the AgentCallbackLogin application. That's why it should be used only with the AgentCallbackLogin app. Uses the monitoring functions in chan_agent instead of Monitor application. That has to be configured in the agents.conf file.

Normally the app returns 0 unless the options are passed.

Syntax

AgentMonitorOutgoing([options])

Arguments

- **options**
  - d: make the app return -1 if there is an error condition.
  - c: change the CDR so that the source of the call is Agent/agent_id
  - n: don't generate the warnings when there is no callerid or the agentid is not known. It's handy if you want to have one context for agent and non-agent calls.

See Also

agents.conf

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
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 these signals may be disabled by setting the `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 these signals may be disabled by setting the` channel variable to `None` - AGISIGHUP` channel variable to `no` before executing the AGI application.

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 run the AGI script text string, one of:
  - SUCCESS
  - FAILURE
  - NOTFOUND
  - HANGUP

**Syntax**

```plaintext
AGI(command[,arg1[,arg2]])
```

**Arguments**

- `command`
- `arg1`
- `arg2`

**See Also**

Asterisk 10 Application_EAGI
Asterisk 10 Application_DeadAGI

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.

Only 1 signalling format is supported at this time: Ademco Contact ID. Only 1 signalling format is supported at this time: Ademco Contact ID.

Syntax

```plaintext
AlarmReceiver()
```

Arguments

See Also

alarmreceiver.conf

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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:

- **AMDDSTATUS** - This is the status of the answering machine detection
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

```python
AMD([initialSilence[,greeting[,afterGreetingSilence[,totalAnalysis
Time[,minimumWordLength[,betweenWordSilence[,maximumNumberOfWords[,sile
```
See Also

Asterisk 10 Application_Hangup

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application_VMAuthenticate
Asterisk 10 Application_DISA
Asterisk 10 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

```python
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 10 Application_ControlPlayback
- Asterisk 10 Application_WaitExten
- Asterisk 10 Application_BackgroundDetect
- Asterisk 10 Function_TIMEOUT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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

```plaintext
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-10-r340810.

Asterisk 10 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`
Syntax

```plaintext
Bridge(channel[,options])
```

Arguments

- `channel` - The current channel is bridged to the specified `channel`.
- `options` -
  - `p` - Play a courtesy tone to `channel`.
  - `h` - Allow the called party to hang up by sending the '#' DTMF digit.
  - `R` - 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(x:y:z)` - 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: `yes`|`no` (default `yes`) Play sounds to the caller. `yes`|`no` File to play when time is up. `File to play when call begins. 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`.
  - `W` - 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`.
  - `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-10-r340810.

Asterisk 10 Application_Busy

Busy()

Synopsis

Indicate the Busy condition.

Description

This application will indicate the busy condition to the calling channel.

Syntax

```plaintext
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.
Asterisk 10 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()

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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
Asterisk 10 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-10-r340810.

Asterisk 10 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 - 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.
  - Technology2/Resource2 - Optional extra devices to check. If you need more than one, enter them as Technology2/Resource2&Technology3/Resource3&....
- 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-10-r340810.

**Asterisk 10 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
  - NOCHANNEL
  - SUCCESS Are set to the result of the redirection

Syntax

ChannelRedirect (channel[,context[,extension,priority]])

Arguments

- channel
- context
- extension
- priority

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 Application_ChanSpy

ChanSpy()

Synopsis

Listen to a channel, and optionally whisper into it.

Description

This application is used to listen to the audio from an Asterisk channel. This includes the audio coming in and out of the channel being spied on. If the `chanprefix` parameter is specified, only channels beginning with this string will be spied upon.

While spying, the following actions may be performed:

- Dialing `#` cycles the volume level.
- Dialing `*` will stop spying and look for another channel to spy on.
- Dialing a series of digits followed by `#` builds a channel name to append to `chanprefix`. For example, executing `ChanSpy(Agent)` and then dialing the digits `1234#` while spying will begin spying on the channel `Agent/1234`. Note that this feature will be overridden if the `d` option is used.

The `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` - 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` - both `grp` and `SPYGROUP` can contain either a single group or a colon-delimited list of groups, such as `sales:support:accounting`.
    - `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 will be used when searching for the name (i.e. if SIP/1000 is the channel being spied on and no mailbox is specified, then `1000` will be used when searching for the name).
    - `mailbox`
    - `context`
  - `o` - Only listen to audio coming from this channel.
- q - Don't play a beep when beginning to spy on a channel, or speak the selected channel name.
- r - Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is `chanspy`.
  - basename
- s - Skip the playback of the channel type (i.e. SIP, IAX, etc) when speaking the selected channel name.
- S - Stop when no more channels are left to spy on.
- v - Adjust the initial volume in the range from -4 to 4. A negative value refers to a quieter setting.
  - value
- w - Enable whisper mode, so the spying channel can talk to the spied-on channel.
- W - Enable private whisper mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.
- x - Specify a DTMF digit that can be used to exit the application.
- X - Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the `SPY_EXIT_CONTEXT` channel variable. The name of the last channel that was spied on will be stored in the `SPY_CHANNEL` variable.
- 4 - spy mode
- 5 - whisper mode
- 6 - barge mode

See Also

Asterisk 10 Application_ExtenSpy

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 Application_ConfBridge
ConfBridge()

Synopsis

Conference bridge application.

Description

Enters the user into a specified conference bridge. The user can exit the conference by hangup or DTMF menu option.

Syntax

ConfBridge([confno[,bridge_profile[,user_profile[,menu]]]])

Arguments

- confno - The conference number
- bridge_profile - The bridge profile name from confbridge.conf. When left blank, a dynamically built bridge profile created by the CONFBRIDGE dialplan function is searched for on the channel and used. If no dynamic profile is present, the 'default_bridge' profile found in confbridge.conf is used. It is important to note that while user profiles may be unique for each participant, mixing bridge profiles on a single conference is NOT recommended and will produce undefined results.
- user_profile - The user profile name from confbridge.conf. When left blank, a dynamically built user profile created by the CONFBRIDGE dialplan function is searched for on the channel and used. If no dynamic profile is present, the 'default_user' profile found in confbridge.conf is used.
- menu - The name of the DTMF menu in confbridge.conf to be applied to this channel. No menu is applied by default if this option is left blank.

See Also

Asterisk 10 Application_ConfBridge
Asterisk 10 Function_CONFBRIDGE
Asterisk 10 Function_CONFBRIDGE_INFO

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application_Busy
Asterisk 10 Application_Progress
Asterisk 10 Application_PlayTones
Asterisk 10 Application_Hangup

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Application_While
Asterisk 10 Application_EndWhile
Asterisk 10 Application_ExitWhile

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_ControlPlayback**
ControlPlayback()

Synopsis

Play a file with fast forward and rewind.

Description

This application will play back the given filename.

It sets the following channel variables upon completion:

- CPLAYBACKSTATUS - Contains the status of the attempt as a text string
  - SUCCESS
  - USERSTOPPED
  - ERROR
- CPLAYBACKOFFSET - Contains the offset in ms into the file where playback was at when it stopped. -1 is end of file.
- CPLAYBACKSTOPKEY - If the playback is stopped by the user this variable contains the key that was pressed.

Syntax

ControlPlayback(filename[,skipms[,ff[,rew[,stop[,pause[,restart[,options]]]]]]])

Arguments

- filename
- skipms - This is number of milliseconds to skip when rewinding or fast-forwarding.
- ff - Fast-forward when this DTMF digit is received. (defaults to #)
- rew - Rewind when this DTMF digit is received. (defaults to *)
- stop - Stop playback when this DTMF digit is received.
- pause - Pause playback when this DTMF digit is received.
- restart - Restart playback when this DTMF digit is received.
- options
  - time - Start at time ms from the beginning of the file.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Application_DAHDIBarge**

**DAHDIBarge()**

**Synopsis**

Barge in (monitor) DAHDI channel.

**Description**

Barges in on a specified DAHDI `channel` or prompts if one is not specified. Returns `-1` when caller user hangs up and is independent of the state of the channel being monitored.

**Syntax**

```plaintext
DAHDIBarge([channel])
```

**Arguments**

- `channel` - Channel to barge.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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

```
DAHRDIRAS(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-10-r340810.

**Asterisk 10 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

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**

```plaintext
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-10-r340810.

**Asterisk 10 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**

```plaintext
DAHDISendKeypadFacility(digits)
```

**Arguments**

- `digits`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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

Asterisk 10 Application_DBdel

DBdel()

Synopsis

Delete a key from the asterisk database.

Description

This application will delete a key from the Asterisk database.

This application has been DEPRECATED in favor of the DB_DELETE function. This application has been DEPRECATED in favor of the DB_DELETE function.

Syntax

```
DBdel(family,key)
```
**Arguments**

- family
- key

**See Also**

- Asterisk 10 Function_DB_DELETE
- Asterisk 10 Application_DBdeltree
- Asterisk 10 Function_DB

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 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 10 Function_DB_DELETE
- Asterisk 10 Application_DBdel
- Asterisk 10 Function_DB

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 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 these signals may be disabled by setting the `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 -channel variable to None - AGISIGHUP channel variable to no before executing the AGI application.`

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 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 10 Application_AGI
Asterisk 10 Application_EAGI

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 If the None - OUTBOUND_GROUP variable is set, all peer channels created by this application will be put into that group (as in Set(GROUP())=...). If the If the None - 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 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/Resource3&....
- **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.

- c - Reset the call detail record (CDR) for this call.
- c - If the Dial() application cancels this call, always set the flag to tell the channel driver that the call is 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. Many SIP and ISDN phones cannot send DTMF digits until the call is connected. If you wish to use this option with these phones, you can use the Many SIP and ISDN phones cannot send DTMF digits until the call is connected. If you wish to use this option with these phones, you can use the Answer application before dialing.
- 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 progress is specified, its DTMF is sent immediately after receiving a PROGRESS message.
- e - Execute the h extension for peer after the call ends
- e - 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.
- f - When the caller hangs up, transfer the called party to the specified destination and start execution at that location. Any channel variables you want the called channel to inherit from the caller channel must be prefixed with one or two underbars ('_'). Any channel variables you want the called channel to inherit from the caller channel must be prefixed with one or two underbars ('_').
- g - Proceed with dialplan execution at the next priority in the current context 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. You cannot use any additional action post answer options in conjunction with this option. You cannot use any additional action post answer options in conjunction with this option.
- 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. Many SIP and ISDN phones cannot send DTMF digits until the call is connected. If you wish to allow DTMF disconnect before the dialed party answers with these phones, you can use the Many SIP and ISDN phones cannot send DTMF digits until the call is connected. If you wish to allow DTMF disconnect before the dialed party answers with these phones, you can use the Answer application before dialing.
- i - Asterisk will ignore any forwarding requests it may receive on this dial attempt.
- i - Asterisk will ignore any connected line update requests or 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.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 - 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: If set, this variable causes Asterisk to play the prompts to the callee. If specified, filename specifies the sound prompt to play when the timeout is reached. If not set, the time remaining will be announced. If specified, filename specifies the sound prompt to play when the timeout is reached. If not set, the time remaining will be announced. 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.
- m - Provide hold music to the calling party until a requested channel answers. A specific music on hold class (as defined in musiconhold.conf) can be specified.
- 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: If set, this action will be taken after the macro finished executing. You cannot use any additional action post answer options in conjunction with this option. Also, pbx services are not run on the peer (called) channel, so you will not be able to set timeouts via the TIMEOUT() function in this macro. You cannot use any additional action post answer options in conjunction with this option. Also, pbx services are not run on the peer (called) channel, so you will not be able to set timeouts via the TIMEOUT() function in this macro. Be aware of the limitations that macros have, specifically with regards to use of the WaitExten application. For more information, see the documentation for Macro().
- macro - Name of the macro that should be executed.
• `arg` - Macro arguments
• `n` - This option is a modifier for the call screening/privacy mode. (See the `p` and `P` options.) It specifies that no introductions are to be saved in the `priv-callerintros` directory.
  • `delete` - With `delete` either not specified or set to 0, the recorded introduction will not be deleted if the caller hangs up while the remote party has not yet answered. With `delete` set to 1, the introduction will always be deleted.
• `N` - This option is a modifier for the call screening/privacy mode. It specifies that if Caller*ID is present, do not screen the call.
  • `o` - If `x` is not provided, specify that the CallerID that was present on the calling channel be stored as the CallerID on the called channel. This was the behavior of Asterisk 1.0 and earlier. If `x` is provided, specify the CallerID stored on the called channel. Note that `o(CALLERID(all))` is similar to option `o` without the parameter.
• `p` - 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`
• `e` - Default: Indicate ringing to the calling party, even if the called party isn't actually ringing. Pass no audio to the calling party until the called channel has answered.
  • `tone` - Indicate progress to calling party. Send audio `tone` from indications.conf
• `S` - Hang up the call `x` seconds after the called party has answered.
  • `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. You cannot use any additional action post answer options in conjunction with this option. Also, pbx services are not run on the peer (called) channel, so you will not be able to set timeouts via the TIMEOUT() function in this routine. You cannot use any additional action post answer options in conjunction with this option. Also, pbx services are not run on the peer (called) channel, so you will not be able to set timeouts via the TIMEOUT() function in this routine.
  • `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-10-r340810.

Asterisk 10 Application_Dictate

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
**Dictate()**

**Synopsis**

Virtual Dictation Machine.

**Description**

Start dictation machine using optional `base_dir` for files.

**Syntax**

```python
Dictate([base_dir[,filename]])
```

**Arguments**

- `base_dir`
- `filename`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_Directory**

**Directory()**

**Synopsis**

Provide directory of voicemail extensions.

**Description**

This application will present the calling channel with a directory of extensions from which they can search by name. The list of names and corresponding extensions is retrieved from the voicemail configuration file, `voicemail.conf`.

This application will immediately exit if one of the following DTMF digits are received and the extension to jump to exists:
- 0 - Jump to the 'o' extension, if it exists.
- * - Jump to the 'a' extension, if it exists.

**Syntax**

```python
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** - Only one of the 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.
  - 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.
  - 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 - 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.
  - 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.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 specify 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 10 Application_Authenticate
Asterisk 10 Application_VMAuthenticate

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Application_NoOp
Asterisk 10 Application_Verbose
Asterisk 10 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 these signals may be disabled by setting the 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 None - AGISIGHUP channel variable to no before executing the AGI application.

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 run the AGI script text string, one of:
  - SUCCESS
  - FAILURE
  - NOTFOUND
  - HANGUP

Syntax

```
EAGI(command[,arg1[,,arg2]])
```

Arguments

- **command**
- **args**
  - **arg1**
  - **arg2**
**Asterisk 10 Application_Echo**

**Echo()**

**Synopsis**

Echo audio, video, DTMF back to the calling party

**Description**

Echoes back any audio, video or DTMF frames read from the calling channel back to itself. Note: If '#' detected application exits

This application does not automatically answer and should be preceded by an application such as Answer() or Progress().

**Syntax**

```plaintext
Echo()
```

**Arguments**

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_EndWhile**

**EndWhile()**

**Synopsis**

End a while loop.

**Description**

Return to the previous called `While()`. 
Syntax

EndWhile()

Arguments

See Also

Asterisk 10 Application_While
Asterisk 10 Application_ExitWhile
Asterisk 10 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_Exec

Exec()

Synopsis

Executes dialplan application.

Description

Allows an arbitrary application to be invoked even when not hard coded into the dialplan. If the underlying application terminates the dialplan, or if the application cannot be found, Exec will terminate the dialplan.

To invoke external applications, see the application System. If you would like to catch any error instead, see TryExec.

Syntax

Exec(arguments)

Arguments

- appname - Application name and arguments of the dialplan application to execute.
  * arguments

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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`
  - `appiffalse`

See Also

Import Version
This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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]?appargs)
```

Arguments
- `day_condition`
  - `times`
  - `weekdays`
  - `mdays`
Asterisk 10 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 10 Application_While
Asterisk 10 Application_EndWhile
Asterisk 10 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.

The 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 - both both grp and SPYGROUP can contain either a single group or a colon-delimited list of groups, such as sales: support: accounting.
    - grp - Only spy on channels in which one or more of the groups listed in grp matches one or more groups from the SPYGROUP variable set on the channel to be spied upon.
  - n - Say the name of the person being spied on if that person has recorded his/her name. If a context is specified, then that voicemail context will be searched when retrieving the name, otherwise the default context be used when searching for the name (i.e. if SIP/1000 is the channel being spied on and no mailbox is specified, then 1000 will be used when searching for the name).
    - mailbox
    - context
  - o - Only listen to audio coming from this channel.
  - q - Don’t play a beep when beginning to spy on a channel, or speak the selected channel name.
  - r - Record the session to the monitor spool directory. An optional base for the filename may be specified. The default is chanspy.
    - basename
  - s - Skip the playback of the channel type (i.e. SIP, IAX, etc) when speaking the selected channel name.
  - S - Stop when there are no more extensions left to spy on.
  - v - Adjust the initial volume in the range from -4 to 4. A negative value refers to a quieter setting.
    - value
  - w - Enable whisper mode, so the spying channel can talk to the spied-on channel.
  - W - Enable private whisper mode, so the spying channel can talk to the spied-on channel but cannot listen to that channel.
  - x
    - digit - Specify a DTMF digit that can be used to exit the application.
  - X - Allow the user to exit ChanSpy to a valid single digit numeric extension in the current context or the context specified by the SPY_EXIT_CONTEXT channel variable. The name of the last channel that was spied on will be stored in the SPY_CHANNEL.
variable.
• 4 - spy mode
• 5 - whisper mode
• 6 - barge mode

See Also

Asterisk 10 Application_ChanSpy

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_ExternallIVR

ExternallIVR()

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

```
ExternallIVR([arg1][,arg2][,options])
```

Arguments

• command|ivr://host
  • arg1
  • arg2
• options
  • n - Tells ExternallIVR() not to answer the channel.
  • i - Tells ExternallIVR() 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.
  • H - Tells ExternallIVR() to run on a channel that has been hung up and will not look for hangups. The external application must exit with an H command.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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-10-r340810.

Asterisk 10 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 10 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
  - s - Playback the incoming status message prior to starting the follow-me step(s)
  - a - Record the caller's name so it can be announced to the callee on each step.
  - n - Playback the unreachable status message if we've run out of steps to reach the or the callee has elected not to be reachable.
  - N - Don't answer the incoming call until we're ready to connect the caller or give up. This will disable all the other options while implicitly turning on the 'd' option.
  - d - Disable the 'Please hold while we try to connect your call' announcement.
  - 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.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Forks the Call Data Record.

Description

Causes the Call Data Record to fork an additional cdr record starting from the time of the fork call. This new cdr record will be linked to end of the list of cdr records attached to the channel. The original CDR has a LOCKED flag set, which forces most cdr operations to skip it, except for the functions that set the answer and end times, which ignore the LOCKED flag. This allows all the cdr records in the channel to be 'ended' together when the channel is closed.

The CDR() func (when setting CDR values) normally ignores the LOCKED flag also, but has options to vary its behavior. The 'T' option (described below), can override this behavior, but beware the risks.

First, this app finds the last cdr record in the list, and makes a copy of it. This new copy will be the newly forked cdr record. Next, this new record is linked to the end of the cdr record list. Next, the new cdr record is RESET (unless you use an option to prevent this)

This means that:

1. All flags are unset on the cdr record
2. the start, end, and answer times are all set to zero.
3. the billsec and duration fields are set to zero.
4. the start time is set to the current time.
5. the disposition is set to NULL.

Next, unless you specified the v option, all variables will be removed from the original cdr record. Thus, the v option allows any CDR variables to be replicated to all new forked cdr records. Without the v option, the variables on the original are effectively moved to the new forked cdr record.

Next, if the s option is set, the provided variable and value are set on the original cdr record.

Next, if the a option is given, and the original cdr record has an answer time set, then the new forked cdr record will have its answer time set to its start time. If the old answer time were carried forward, the answer time would be earlier than the start time, giving strange duration and billsec times.

If the d option was specified, the disposition is copied from the original cdr record to the new forked cdr. If the D option was specified, the destination channel field in the new forked CDR is erased. If the e option was specified, the 'end' time for the original cdr record is set to the current time. Future hang-up or ending events will not override this time stamp. If the A option is specified, the original cdr record will have it ANS_LOCKED flag set, which prevent future answer events from updating the original cdr record's disposition. Normally, an ANSWERED event would mark all cdr records in the chain as ANSWERED. If the T option is specified, the original cdr record
will have its `DONT TOUCH` flag set, which will force the `cdr_answer`, `cdr_end`, and `cdr_setvar` functions to leave that cdr record alone.

And, last but not least, the original cdr record has its `LOCKED` flag set. Almost all internal CDR functions (except for the funcs that set the end, and answer times, and set a variable) will honor this flag and leave a `LOCKED` cdr record alone. This means that the newly created forked cdr record will be affected by events transpiring within Asterisk, with the previously noted exceptions.

**Syntax**

```
ForkCDR([options])
```

**Arguments**

- **options**
  - **a** - Update the answer time on the NEW CDR just after it's been inited. The new CDR may have been answered already. The reset that forkcdr does will erase the answer time. This will bring it back, but the answer time will be a copy of the fork/start time. It will only do this if the initial cdr was indeed already answered.
  - **A** - Lock the original CDR against the answer time being updated. This will allow the disposition on the original CDR to remain the same.
  - **d** - Copy the disposition forward from the old cdr, after the init.
  - **D** - Clear the `dstchannel` on the new CDR after reset.
  - **e** - End the original CDR. Do this after all the necessary data is copied from the original CDR to the new forked CDR.
  - **r** - Do NOT reset the new cdr.
  - **s(name=val)** - Set the CDR var `name` in the original CDR, with value `val`.
  - **T** - Mark the original CDR with a `DONT TOUCH` flag. `setvar`, `answer`, and `end` cdr funcs will obey this flag; normally they don't honor the `LOCKED` flag set on the original CDR record. Using this flag may cause CDR's not to have their end times updated! It is suggested that if you specify this flag, you might wish to use the `Using this flag may cause CDR's not to have their end times updated! It is suggested that if you specify this flag, you might wish to use the` flag as well.
  - **v** - When the new CDR is forked, it gets a copy of the vars attached to the current CDR. The vars attached to the original CDR are removed unless this option is specified.

**See Also**

- Asterisk 10 Function_CDR
- Asterisk 10 Application_NoCDR
- Asterisk 10 Application_ResetCDR

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**

```plaintext
GetCPEID()
```

**Arguments**

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_Gosub**

**Gosub()**

**Synopsis**

Jump to label, saving return address.

**Description**

Jumps to the label specified, saving the return address.

**Syntax**

```plaintext
Gosub([context[,exten,arg1[,...][,argN]]])
```

**Arguments**

- `context`
- `exten`
- `priority`
- `arg1`
- `argN`

**See Also**

- Asterisk 10 Application_GosubIf
- Asterisk 10 Application_Macro
- Asterisk 10 Application_Goto
- Asterisk 10 Application_Return
- Asterisk 10 Application_StackPop

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**
  - **labeliffalse**

**See Also**

Asterisk 10 Application_Gosub  
Asterisk 10 Application_Return  
Asterisk 10 Application_MacroIf  
Asterisk 10 Function_IF  
Asterisk 10 Application_GotoIf  

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 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. 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 10 Application_GotoIf
Asterisk 10 Application_GotoIfTime
Asterisk 10 Application_Gosub
Asterisk 10 Application_Macro

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 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.
  - labeliffalse - Continue at `labeliffalse` if the condition is false.

See Also

- Asterisk 10 Application_Goto
- Asterisk 10 Application_GotoIfTime
- Asterisk 10 Application_GosubIf
- Asterisk 10 Application_MacroIf

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_GotoIfTime**

**GotoIfTime()**

**Synopsis**

Conditional Goto based on the current time.

**Description**

This application will set the context, extension, and priority in the channel structure based on the evaluation of the given time specification. After this application completes, the pbx engine will continue dialplan execution at the specified location in the dialplan. If the current time is within the given time specification, the channel will continue at `labeliftrue`. Otherwise the channel will continue at `labeliffalse`. If the label chosen by the condition is omitted, no jump is performed, and execution passes to the next instruction. If the target jump location is bogus, the same actions would be taken as for `Goto`. Further information on the time specification can be found in examples illustrating how to do time-based context includes in the dialplan.

**Syntax**

```
GotoIfTime(timesweekdaysmdaysmonths[,timezone]?[labeliftrue][:labeliffalse])
```

Arguments
See Also

Asterisk 10 Application_Gotolf
Asterisk 10 Function_IFTIME
Asterisk 10 Function_TESTTIME

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Application_Answer
Asterisk 10 Application_Busy
Asterisk 10 Application_Congestion

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

Asterisk 10 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).

ICES version 2 client and server required.

Syntax

ICES(config)

Arguments

- config - ICES configuration file.

See Also
Asterisk 10 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 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=channelnamevariable)

Arguments

- newvar
- vardata
  - channelname
  - variable

See Also

Asterisk 10 Application_Set

Asterisk 10 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

```plaintext
Incomplete([n])
```

Arguments

- `n` - If specified, then Incomplete will not attempt to answer the channel first. Most channel types need to be in Answer state in order to receive DTMF. 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-10-r340810.

**Asterisk 10 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

```plaintext
IVRDemo(filename)
```

Arguments

- `filename`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application JabberJoin**

**JabberJoin()**

**Synopsis**
Join a chat room

Description

Allows Asterisk to join a chat room.

Syntax

```plaintext
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. 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-10-r340810.

Asterisk 10 Application_JabberLeave

**JabberLeave()**

Synopsis

Leave a chat room

Description

Allows Asterisk to leave a chat room.

Syntax

```plaintext
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-10-r340810.
Asterisk 10 Application_JabberSend

JabberSend()

Synopsis

Sends an XMPP message to a buddy.

Description

Sends the content of message as text message from the given account to the buddy identified by jid.

Example: JabberSend(asterisk,bob@domain.com,Hello world) sends "Hello world" to bob@domain.com as an XMPP message from the account asterisk, configured in jabber.conf.

Syntax

JabberSend(account,jid,message)

Arguments

- account - The local named account to listen on (specified in jabber.conf)
- jid - Jabber ID of the buddy to send the message to. It can be a bare JID (username@domain) or a full JID (username@domain/resource).
- message - The message to send.

See Also

Asterisk 10 Function_JABBER_STATUS
Asterisk 10 Function_JABBER_RECEIVE

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_JabberSendGroup

JabberSendGroup()

Synopsis

Send a Jabber Message to a specified chat room

Description

Allows user to send a message to a chat room via XMPP.

To be able to send messages to a chat room, a user must have previously joined it. Use the To
**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-10-r340810.

**Asterisk 10 Application_JabberStatus**

**JabberStatus()**

**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.

Online.

Chatty.

Away.

Extended Away.

Do Not Disturb.

Offline.

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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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 `None - MACRO_EXTEN`, The calling extension, context, and priority are stored in `None - MACRO_CONTEXT` and The calling extension, context, and priority are stored in `None - MACRO_PRIORITY` respectively. Arguments become The calling extension, context, and priority are stored in `None - ARG1`, The calling extension, context, and priority are stored in `None - 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 `None - 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.

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 explicit Return() calls instead.

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 10 Application_MacroExit
Asterisk 10 Application_Goto
Asterisk 10 Application_Gosub

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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()

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

```plaintext
MacroExclusive(name[,arg1[,arg2[,...]]])
```

Arguments

- `name` - The name of the macro
- `arg1`
- `arg2`

See Also

Asterisk 10 Application_Macro

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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

```plaintext
MacroExit()
```

Arguments

See Also

Asterisk 10 Application_Macro

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_MacroIf

Macrolf()
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()

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`
- `macroiffalse`

See Also

Asterisk 10 Application_GotoIf
Asterisk 10 Application_GosubIf
Asterisk 10 Function_IF

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_MailboxExists

MailboxExists()

Synopsis

Check to see if Voicemail mailbox exists.

Description

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:

- `VMBOXEXISTSSSTATUS` - This will contain the status of the execution of the MailboxExists application. Possible values include:
  - SUCCESS
  - FAILED
Syntax

```plaintext
MailboxExists(mailbox[@context][,options])
```

Arguments

- `mailbox`
  - `mailbox`
  - `context`
- `options` - None options.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 `#`.

The DAHDI kernel modules and at least one hardware driver (or dahdi_dummy) must be present for conferencing to operate properly. In addition, the chan_dahdi channel driver must be loaded for the DAHDI kernel modules and at least one hardware driver (or dahdi_dummy) 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**

```plaintext
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`. This does not work with non-DAHDI channels in the same conference). This does not work with non-DAHDI channels in the same conference).
  - `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.
- 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
- 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. The key used is set to channel variable MEETME_EXIT_KEY. Option Option s has priority for * since it cannot change its activation code.
- keys
- P - Always prompt for the pin even if it is specified.
- g - Quiet mode (don’t play enter/leave sounds).
- c - 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 - Close the conference when last marked user exits
- 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. Option Option s has priority for * since it cannot change its activation code.
- i - Do not play message when first person enters
- S - Kick the user x seconds after he entered into the conference.
- 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: File to play when time is up. File to play as warning if y is defined. The default is to say the time remaining.
  - x
  - y
  - z
- pin

See Also

Asterisk 10 Application_MeetMeCount
Asterisk 10 Application_MeetMeAdmin
Asterisk 10 Application_MeetMeChannelAdmin

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_MeetMeAdmin

MeetMeAdmin()

Synopsis

MeetMe conference administration.

Description
Run admin command for conference confno.

Will additionally set the variable Will additionally set the variable None - 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 10 Application_MeetMe

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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 10 Application_MeetMe**

**Import Version**
Asterisk 10 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).

This application sets the following channel variables:

- MESSAGE_SEND_STATUS - This is the time from dialing a channel until when it is disconnected.
- 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 unable to deliver the message for some reason.

Syntax

```
MessageSend(to[, from])
```

Arguments

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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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(usernamedomain[,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
Asterisk 10 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-10-r340810.

Asterisk 10 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

```plaintext
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-10-r340810.

**Asterisk 10 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

```plaintext
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.
**Asterisk 10 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_NOTIFY_STATUS` 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-10-r340810.

**Asterisk 10 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 `None - MVM_FILENAME` and the duration of the message will be stored in `None - MVM_DURATION`.

If the caller hangs up after the recording, the only way to send the message and clean up is to execute in the `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` - Voicemail username
  - `username` - Voicemail username
  - `domain` - Voicemail domain
- `options` -
  - `0` - Jump to the `a` 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-10-r340810.

**Asterisk 10 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().

- MIXMONITOR_FILENAME - Will contain the filename used to record.

Syntax

MixMonitor(filenameextension[,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. Does not include conferences or sounds played to each bridged party. Does not include conferences or sounds played to each bridged party. If you utilize this option inside a Local channel, you must make sure the Local channel is not optimized away. To do this, be sure to call your Local channel with the If you utilize this option inside a Local channel, you must make sure the Local channel is not optimized away. To do this, be sure to call your Local channel with the /n option. For example: Dial(Local/start@mycontext/n)
  - 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
- 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 10 Application_Monitor
Asterisk 10 Application_StopMixMonitor
Asterisk 10 Application_PauseMonitor
Asterisk 10 Application_UnpauseMonitor

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_Monitor
Monitor()

Synopsis

Monitor a channel.

Description

Used to start monitoring a channel. The channel's input and output voice packets are logged to files until the channel hangs up or monitoring is stopped by the StopMonitor application.

By default, files are stored to /var/spool/asterisk/monitor/. Returns -1 if monitor files can't be opened or if the channel is already monitored, otherwise 0.

Syntax

Monitor([file_format[:urlbase][,fname_base[,options]]])

Arguments

- file_format
  - file_format - optional, if not set, defaults to wav
- urlbase
- fname_base - if set, changes the filename used to the one specified.
- options
  - m - when the recording ends mix the two leg files into one and delete the two leg files. If the variable MONITOR_EXEC is set, the application referenced in it will be executed instead of soxmix/sox and the raw leg files will NOT be deleted automatically. soxmix/sox or MONITOR_EXEC is handed 3 arguments, the two leg files and a target mixed file name which is the same as the leg file names only without the in/out designator. If MONITOR_EXEC_ARGS is set, the contents will be passed on as additional arguments to MONITOR_EXEC. Both MONITOR_EXEC and the Mix flag can be set from the administrator interface.
  - b - Don't begin recording unless a call is bridged to another channel.
  - i - Skip recording of input stream (disables m option).
  - o - Skip recording of output stream (disables m option).

See Also

Asterisk 10 Application_StopMonitor

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 preceded 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 10 Application_SayAlpha
Asterisk 10 Application_SayPhonetic

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 preceded by an application such as Answer() or Progress().

**Syntax**

MP3Player(Location)
Asterisk 10 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(name1value1[,name2value2[,...]])

Arguments

- set1
  - name1
  - value1
- set2
  - name2
  - value2

See Also

Asterisk 10 Application_Set

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 omitted, music plays indefinitely. Returns 0 when done, −1 on hangup.

This application does not automatically answer and should be preceded 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-10-r340810.

**Asterisk 10 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**
**Asterisk 10 Application_NoCDR**

**NoCDR()**

**Synopsis**

Tell Asterisk to not maintain a CDR for the current call

**Description**

This application will tell Asterisk not to maintain a CDR for the current call.

**Syntax**

```
NoCDR()
```

**Arguments**

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

---

**Asterisk 10 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**

---

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

#### Asterisk 10 Application_ODBCFinish

**ODBCFinish()**

**Synopsis**

Clear the resultset of a successful 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-10-r340810.

#### Asterisk 10 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]])
```

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.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_OSPAuth**

OSPAuth()

Synopsis

OSP Authentication.

Description

Authenticate a call by OSP.

Input variables:

- `OSPINPEERIP` - The last hop IP address.
- `OSPI NTOKEN` - The inbound OSP token.

Output variables:
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

```python
OSPAuth([provider[,options]])
```

Arguments

- **provider** - The name of the provider that authenticates the call.
- **options** - Reserved.

See Also

Asterisk 10 Application_OSPLookup
Asterisk 10 Application_OSPNext
Asterisk 10 Application_OSPFinish

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**
OSPFinish([[cause[,options]]])

Arguments

- cause - Hangup cause.
- options - Reserved.

See Also

Asterisk 10 Application_OSPA
Asterisk 10 Application_OSPLookup
Asterisk 10 Application_OSPNext

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.
- OSPIPHERIP - The last hop IP address.
- OSPINHANDLE - The inbound call OSP transaction handle.
- OSPINTIMELIMIT - The inbound call duration limit in seconds.
- OSPINNETWORKID - The inbound source network ID.
- OSPINNFRN - The inbound routing number.
- OSPINPCIC - The inbound carrier identification code.
- OSPINPDID - The inbound number portability database dip indicator.
- OSPINSPI - The inbound service provider identity.
- OSPINOCN - The inbound operator company number.
- OSPINSPI - The inbound service provider name.
- OSPINALTSPN - The inbound alternate service provider name.
- OSPINMCC - The inbound mobile country code.
- OSPINMNC - The inbound mobile network code.
- OSPINTOHST - The inbound To header host part.
- OSPINDIVUSER - The inbound Diversion header user part.
- OSPINDIVHOST - The inbound Diversion header host part.
- OSPINCUSTOMINFO - The inbound custom information, where n is the index beginning with 1 upto 8.

Output variables:
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

```plaintext
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 10 Application_OSPAuth
- Asterisk 10 Application_OSPNext
- Asterisk 10 Application_OSPFinish

### Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
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:

- OSPOUTFTECH - 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.
- OSPOUTNPN - 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

```python
OSPNext()
```

Arguments

See Also

Asterisk 10 Application_OSPAuth
Asterisk 10 Application_OSPLookup
Asterisk 10 Application_OSPFinish

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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 callers 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 then one enter them as Technology2/Resource2& Technology3/Resource3&.....
* options
  * d - Full duplex audio
  * i - Ignore attempts to forward the call
  * q - Quiet, do not play beep to caller
  * r - Record the page into a file (meetme option r)
  * s - Only dial a channel if its device state says that it is NOT_INUSE
  * A - Play an announcement simultaneously to all paged participants
    * x - The announcement to playback in all devices
  * n - Do not play simultaneous announcement to caller (implies A)
* timeout - Specify the length of time that the system will attempt to connect a call. After this duration, any intercom calls that have not been answered will be hung up by the system.

See Also

Asterisk 10 Application_MeetMe

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_Park

Park()

Synopsis

Park yourself.
Description

Used to park yourself (typically in combination with a supervised transfer to know the parking space).

If you set the If you set the `None - 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.

If the parkeddynamic option is enabled in `features.conf` the following variables can be used to dynamically create new parking lots.

If you set the If you set the `None - PARKINGDYNAMIC` variable and this parking lot exists then it will be used as a template for the newly created dynamic lot. Otherwise, the default parking lot will be used.

If you set the If you set the `None - PARKINGDYNCONTEXT` variable then the newly created dynamic parking lot will use this context.

If you set the If you set the `None - PARKINGDYNEX TEN` variable then the newly created dynamic parking lot will use this extension to access the parking lot.

If you set the If you set the `None - PARKINGDYNPOS` variable then the newly created dynamic parking lot will use those parking positions.

This application must be used as the first extension priority to be recognized as a parking access extension. DTMF transfers and some channel drivers need this distinction to operate properly. The parking access extension in this case is treated like a dialplan hint. This application must be used as the first extension priority to be recognized as a parking access extension. DTMF transfers and some channel drivers need this distinction to operate properly. The parking access extension in this case is treated like a dialplan hint.

Parking lots automatically create and manage dialplan extensions in the parking lot context. You do not need to explicitly use this application in your dialplan. Instead, all you should do is include the parking lot context in your dialplan.

Syntax

```
Park([timeout[,return_context[,return_exten[,return_priority[,options]]]]])
```

Arguments

- `timeout` - A custom parking timeout for this parked call. Value in milliseconds.
- `return_context` - The context to return the call to after it times out.
- `return_exten` - The extension to return the call to after it times out.
- `return_priority` - The priority to return the call to after it times out.
- `options` - A list of options for this parked call.
  - `r` - Send ringing instead of MOH to the parked call.
  - `R` - Randomize the selection of a parking space.
  - `s` - Silence announcement of the parking space number.
parking_lot_name - Specify in which parking lot to park a call. The parking lot used is selected in the following order: 1) parking_lot_name option 2) PARKINGLOT variable 3) CHANNEL(parkinglot) function (Possibly preset by the channel driver.) 4) Default parking lot.

See Also

Asterisk 10 Application_ParkAndAnnounce
Asterisk 10 Application_ParkedCall

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_ParkAndAnnounce

ParkAndAnnounce()

Synopsis

Park and Announce.

Description

Park a call into the parkinglot and announce the call to another channel.

The variable None - PARKEDAT will contain the parking extension into which the call was placed. Use with the Local channel to allow the dialplan to make use of this information.

Syntax

```
ParkAndAnnounce(announce[:announce1[:...]],timeout,dial[,return_context])
```

Arguments

- `announce_template`
- `announce` - Colon-separated list of files to announce. The word PARKED will be replaced by a say_digits of the extension in which the call is parked.
- `timeout` - Time in seconds before the call returns into the return context.
- `dial` - The app_dial style resource to call to make the announcement. Console/dsp calls the console.
- `return_context` - The goto-style label to jump the call back into after timeout. Default priority+1.

See Also

Asterisk 10 Application_Park
Asterisk 10 Application_ParkedCall

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_ParkedCall
ParkedCall()

Synopsis

Retrieve a parked call.

Description

Used to retrieve a parked call from a parking lot.

Parking lots automatically create and manage dialplan extensions in the parking lot context. You do not need to explicitly use this application in your dialplan. Instead, all you should do is include the parking lot context in your dialplan.

Syntax

ParkedCall([exten[,parking_lot_name]])

Arguments

- exten - Parking space extension to retrieve a parked call. If not provided then the first available parked call in the parking lot will be retrieved.
- parking_lot_name - Specify from which parking lot to retrieve a parked call. The parking lot used is selected in the following order: 1) parking_lot_name option 2) PARKINGLOT variable 3) CHANNEL(parkinglot) function (Possibly preset by the channel driver.) 4) Default parking lot.

See Also

Asterisk 10 Application_Park
Asterisk 10 Application_ParkAndAnnounce

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application_UnpauseMonitor

Import Version
This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 unpause 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
- **queue name**
- **interface**
- **options**
- **reason** - Is used to add extra information to the appropriate queue_log entries and manager events.

See Also

Asterisk 10 Application_Queue
Asterisk 10 Application_Pickup

Pickup()

Synopsis

Directed extension call pickup.

Description

This application can pickup any ringing channel that is calling the specified extension. If no context is specified, the current context will be used. If you use the special string PICKUPMARK for the context parameter, for example 10@PICKUPMARK, this application tries to find a channel which has defined a PICKUPMARK channel variable with the same value as extension (in this example, 10).

When no parameter is specified, the application will pickup a channel matching the pickup group of the active channel.

Syntax

```
Pickup(extension[@context][&extension2[@context2][&...]])
```

Arguments

- ext
  - extension
  - context
- ext2
  - extension2
  - context2

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 Application_PickupChan

PickupChan()

Synopsis

Pickup a ringing channel.

Description

This will pickup a specified channel if ringing.

Syntax

```
PickupChan(Technology/Resource[&Technology2/Resource2[&...]][,options])
```

Arguments

- Technology/Resource
  - Technology/Resource
  - Technology2/Resource2
- options
  - p - Channel name specified partial name. Used when find channel by callid.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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. Not all channel types support playing messages while still on hook. Not all channel types support playing messages while still on hook.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Application_StopPlayTones**
Asterisk 10 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 10 Application_Zapateller

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_Proceeding

Proceeding()

Synopsis

Indicate proceeding.
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-10-r340810.

**Asterisk 10 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 10 Application_Busy
Asterisk 10 Application_Congestion
Asterisk 10 Application_Ringing
Asterisk 10 Application_PlayTones

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 preceded 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` can be:
  - 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).
  - 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 calling party to enable parking of the call by sending the DTMF sequence defined for call parking in features.conf.
  - K - Allow the party to enable parking of the call by sending the DTMF sequence defined for call parking in features.conf.
  - x - 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 user to write the conversation to disk via MixMonitor.
  - URL - Will be sent to the called party if the channel supports it.
  - announceoverride -
  - timeout - Will cause the queue to fail out after a specified number of seconds, checked between each queues.conf timeout and retry cycle.
  - AGI - Will setup an AGI script to be executed on the calling party's channel once they are connected to a queue member.
  - macro - Will run a macro on the calling party's channel once they are connected to a queue member.
  - gosub - Will run a gosub on the calling party's channel once they are connected to a queue member.
  - rule - Will cause the queue's defaultrule to be overridden by the rule specified.
• position: Attempt to enter the caller into the queue at the numerical position specified. 1 would attempt to enter the caller at the head of the queue, and 3 would attempt to place the caller third in the queue.

See Also

Asterisk 10 Application Queue
Asterisk 10 Application QueueLog
Asterisk 10 Application AddQueueMember
Asterisk 10 Application RemoveQueueMember
Asterisk 10 Application PauseQueueMember
Asterisk 10 Application UnpauseQueueMember
Asterisk 10 Function QUEUE_VARIABLES
Asterisk 10 Function QUEUE_MEMBER
Asterisk 10 Function QUEUE_MEMBER_COUNT
Asterisk 10 Function QUEUE_EXISTS
Asterisk 10 Function QUEUE_WAITING_COUNT
Asterisk 10 Function QUEUE_MEMBER_LIST
Asterisk 10 Function QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application Queue
Asterisk 10 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 10 Function_EXCEPTION

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application_SendDTMF

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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:

- **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-10-r340810.

**Asterisk 10 Application_ReadFile**

**ReadFile()**

Synopsis

Read the contents of a text file into a channel variable.

Description

Read the contents of a text file into channel variable `varname`

ReadFile has been deprecated in favor of Set(varname=${FILE(file,0,length)})

Syntax

```
ReadFile(varname=file[,length])
```

Arguments
varname - Result stored here.
fileparams
  • file - The name of the file to read.
  • length - Maximum number of characters to capture. If not specified defaults to max.

See Also

Asterisk 10 Application_System
Asterisk 10 Application_Read

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_ReceiveFAX (app_fax)

ReceiveFAX()

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
• REMOTESECTIONID - 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
Asterisk 10 Application_ReceiveFAX (res_fax)

ReceiveFAX()

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 10 Function_FAXOPT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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(filenameformat[,silence[,maxduration[,options]]])
```

**Arguments**

- **filename**
  - **filename** - Is the format of the file type to be recorded (wav, gsm, etc).
- **silence** - Is the number of seconds of silence to allow before returning.
- **maxduration** - Is the maximum recording duration in seconds. If missing or 0 there is no maximum.
- **options**
  - **a** - Append to existing recording rather than replacing.
  - **n** - Do not answer, but record anyway if line not yet answered.
  - **q** - quiet (do not play a beep tone).
  - **s** - skip recording if the line is not yet answered.
  - **t** - use alternate '*' terminator key (DTMF) instead of default '#'
  - **x** - Ignore all terminator keys (DTMF) and keep recording until hangup.
  - **k** - Keep recorded file upon hangup.
  - **y** - Terminate recording if **any** DTMF digit is received.

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**
Example: RemoveQueueMember(techsupport,SIP/3000)

**Syntax**

```plaintext
RemoveQueueMember(queuename[,interface[,options]])
```

**Arguments**

- queuename
- interface
- options

**See Also**

- Asterisk 10 Application.Queue
- Asterisk 10 Application.QueueLog
- Asterisk 10 Application.AddQueueMember
- Asterisk 10 Application.RemoveQueueMember
- Asterisk 10 Application.PauseQueueMember
- Asterisk 10 Application.UnpauseQueueMember
- Asterisk 10 Function.QUEUE_VARIABLES
- Asterisk 10 Function.QUEUE_MEMBER
- Asterisk 10 Function.QUEUE_MEMBER_COUNT
- Asterisk 10 Function.QUEUE_EXISTS
- Asterisk 10 Function.QUEUE_WAITING_COUNT
- Asterisk 10 Function.QUEUE_MEMBER_LIST
- Asterisk 10 Function.QUEUE.Member.PENALTY

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_ResetCDR**

**ResetCDR()**

**Synopsis**

Resets the Call Data Record.

**Description**

This application causes the Call Data Record to be reset.

**Syntax**

```plaintext
ResetCDR([options])
```

**Arguments**
• options
  • w - Store the current CDR record before resetting it.
  • a - Store any stacked records.
  • v - Save CDR variables.
  • e - Enable CDR only (negate effects of NoCDR).

See Also

Asterisk 10 Application_ForkCDR
Asterisk 10 Application_NoCDR

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 This application will attempt to place a call using the normal Dial application. If no channel can be reached, the None - 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-10-r340810.
Asterisk 10 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 Jumps to the last label on the stack, removing it. The return None - GOSUB_RETVAL.

Syntax

```
Return([value])
```

Arguments

- value - Return value.

See Also

Asterisk 10 Application_Gosub
Asterisk 10 Application_StackPop

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application_SayAlpha

SayAlpha()

Synopsis

Say Alpha.

Description

This application will play the sounds that correspond to the letters of the given string.

Syntax

SayAlpha(string)

Arguments

* string

See Also

Asterisk 10 Application_SayDigits
Asterisk 10 Application_SayNumber
Asterisk 10 Application_SayPhonetic
Asterisk 10 Function_CHANNEL

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 preceded 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 10 Application_SayCountedNoun
Asterisk 10 Application_SayNumber

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 proceeded 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 10 Application_SayCountedAdj
Asterisk 10 Application_SayNumber

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_SayCountPL**

**SayCountPL()**

Synopsis

Say Polish counting words.

Description

Polish grammar has some funny rules for counting words. for example 1 zloty, 2 zlote, 5 zlotych. This application will take the words for 1, 2-4 and 5 and decide based on grammar rules which one to use with the number you pass to it.

Example: SayCountPL(zloty,zlote,zlotych,122) will give: zlote

Syntax
SayCountPL(word1, word2, word5, number)

Arguments
- word1
- word2
- word5
- number

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_SayDigits**

SayDigits()

Synopsis

Say Digits.

Description

This application will play the sounds that correspond to the digits of the given number. This will use the language that is currently set for the channel.

Syntax

```
SayDigits(digits)
```

Arguments
- digits

See Also

**Asterisk 10 Application_SayAlpha**
**Asterisk 10 Application_SayNumber**
**Asterisk 10 Application_SayPhonetic**
**Asterisk 10 Function_CHANNEL**

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 LANGUAGE() function for more information on setting the language for the channel.

Syntax

SayNumber(digits[,gender])

Arguments

• digits
• gender

See Also

Asterisk 10 Application_SayAlpha
Asterisk 10 Application_SayDigits
Asterisk 10 Application_SayPhonetic
Asterisk 10 Function_CHANNEL

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_SayPhonetic

SayPhonetic()

Synopsis

Say Phonetic.

Description

This application will play the sounds from the phonetic alphabet that correspond to the letters in the given string.

Syntax

SayPhonetic(string)

Arguments
See Also

Asterisk 10 Application_SayAlpha
Asterisk 10 Application_SayDigits
Asterisk 10 Application_SayNumber

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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]]])

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

Asterisk 10 Function_STRFTIME
Asterisk 10 Function_STRPTIME
Asterisk 10 Function_IFTIME

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_SendDTMF

SendDTMF()
Synopsis

Sends arbitrary DTMF digits

Description

DTMF digits sent to a channel with half second pause

It will pass all digits or terminate if it encounters an error.

Syntax

```
SendDTMF(digits[,timeout_ms[,duration_ms[,channel]]])
```

Arguments

- `digits`: List of digits 0-9,*#,abcd
- `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 10 Application_Read

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_SendFAX (app_fax)

SendFAX()

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
- `REMTSTATIONID`: 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-10-r340810.

**Asterisk 10 Application_SendFAX (res_fax)**

**SendFAX()**

**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` - TIFF file to send as a FAX.
- `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 statevents setting in res_fax.conf).
  - `z` - Initiate a T.38 reinvite on the channel if the remote end does not.

See Also

**Asterisk 10 Function_FAXOPT**

Import Version
Asterisk 10 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 Result of transmission will be stored in None - 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 10 Application_SendText
Asterisk 10 Application_SendURL

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_SendText

SendText()

Synopsis

Send a Text Message.

Description

Sends text to current channel (callee).
Result of transmission will be stored in the Result of transmission will be stored in the
None - SENDTEXTSTATUS

- SENDTEXTSTATUS
  - SUCCESS - Transmission succeeded.
  - FAILURE - Transmission failed.
  - UNSUPPORTED - Text transmission not supported by channel.

At this moment, text is supposed to be 7 bit ASCII in most channels. At this moment, text is supposed to be 7 bit ASCII in most channels.

Syntax

```
SendText (text)
```

Arguments

- text

See Also

Asterisk 10 Application_SendImage
Asterisk 10 Application_SendURL

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 Result is returned in the None - 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 10 Application_SendImage
Asterisk 10 Application_SendText

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.

If (and only if), in 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**
Asterisk 10 Application_SetAMAFlags

SetAMAFlags()

Synopsis

Set the AMA Flags.

Description

This application will set the channel's AMA Flags for billing purposes.

Syntax

```
SetAMAFlags([flag])
```

Arguments

- `flag`

See Also

Asterisk 10 Function_CDR

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 Application_SetMusicOnHold**

**SetMusicOnHold()**

Synopsis

Set default Music On Hold class.

Description

!!! DEPRECATED. USe Set(CHANNEL(musicclass)=...) instead !!!

Sets the default class for music on hold for a given channel. When music on hold is activated, this class will be used to select which music is played.

!!! DEPRECATED. USe Set(CHANNEL(musicclass)=...) instead !!!

Syntax

```
SetMusicOnHold(class)
```

Arguments

- **class**

See Also

Import Version
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 Application_Skel**

**Skel()**

**Synopsis**

Simple one line explanation.

**Description**

This application is a template to build other applications from. It shows you the basic structure to create your own Asterisk applications.

**Syntax**

```
Skel(dummy[,options])
```

Arguments

- dummy
- options
  - a - Option A.
  - b - Option B.
  - c - Option C.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_SLASation**

**SLASation()**

**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:

- `FAILURE`
- `CONGESTION`
- `SUCCESS`

**Syntax**

```
SLAStation(station)
```

**Arguments**

- `station` - Station name

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

---

**Asterisk 10 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:

- `FAILURE`
SUCCESS
UNANSWERED
RINGTIMEOUT

Syntax

```latex
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-10-r340810.

Asterisk 10 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.

The protocol has tight delay bounds. Please use short frames and disable/keep short the jitter buffer on the ATA to make sure that responses (ACK etc.) are received in time. The protocol has tight delay bounds. Please use short frames and disable/keep short the jitter buffer on the ATA to make sure that responses (ACK etc.) are received in time.

Syntax

```latex
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.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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.

Syntax

SpeechActivateGrammar(gram\arm_name)

Arguments

- grammar_name

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.

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.
Asterisk 10 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.

Syntax

```plaintext
SpeechCreate(engine_name)
```

Arguments
- `engine_name`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_SpeechDeactivateGrammar

SpeechDeactivateGrammar()

Synopsis
Deactivate a grammar.

Description
This deactivates the specified grammar so that it is no longer recognized.

Syntax

```plaintext
SpeechDeactivateGrammar(grammar_name)
```
Arguments

- `grammar_name` - The grammar name to deactivate

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.

**Syntax**

```
SpeechDestroy()
```

**Arguments**

**See Also**

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Application_SpeechLoadGrammar**

**SpeechLoadGrammar()**

**Synopsis**

Load a grammar.

**Description**

Load a grammar only on the channel, not globally.

**Syntax**
SpeechLoadGrammar(gramar_name,path)

Arguments

- grammar_name
- path

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.

Syntax

SpeechProcessingSound(sound_file)

Arguments

- sound_file

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.

**Syntax**

```plaintext
SpeechStart()
```

**Arguments**

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 Application_SpeechUnloadGrammar

#### SpeechUnloadGrammar()

**Synopsis**

Unload a grammar.

**Description**

Unload a grammar.

**Syntax**

```plaintext
SpeechUnloadGrammar(grammar_name)
```

**Arguments**

- `grammar_name`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 Application_StackPop

#### StackPop()

**Synopsis**

Remove one address from gosub stack.
**Description**

Removes last label on the stack, discarding it.

**Syntax**

```plaintext
StackPop()
```

**Arguments**

**See Also**

- Asterisk 10 Application_Return
- Asterisk 10 Application_Gosub

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**

```plaintext
StartMusicOnHold(class)
```

**Arguments**

- `class`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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()
```

Arguments

See Also

Asterisk 10 Application_MixMonitor

Import Version
This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 Application_StopMusicOnHold

StopMusicOnHold()
Synopsis

Stop playing Music On Hold.

Description

Stops playing music on hold.

Syntax

```plaintext
StopMusicOnHold()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_StopPlayTones

StopPlayTones()

Synopsis

Stop playing a tone list.

Description

Stop playing a tone list, initiated by PlayTones().

Syntax

```plaintext
StopPlayTones()
```

Arguments

See Also

Asterisk 10 Application_PlayTones

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 Result of execution is returned in the None - 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-10-r340810.

Asterisk 10 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 10 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**

```plaintext
TestServer()
```

**Arguments**

See Also

**Asterisk 10 Application_TestClient**

**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-10-r340810.

**Asterisk 10 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 `TRSTATUS` will be set to one of:

- `TRSTATUS`:
  - `SUCCESS` - If the application returned zero.
  - `FAILED` - If the application returned non-zero.
  - `NOAPP` - If the application was not found or was not specified.

Syntax

```
TryExec(arguments)
```

Arguments

- `appname`
Asterisk 10 Application_TrySystem

TrySystem()

Synopsis

Try executing a system command.

Description

Executes a command by using system().

Result of execution is returned in the Result of execution is returned in the None - 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-10-r340810.

Asterisk 10 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

```plaintext
UnpauseMonitor()
```

Arguments

See Also

**Asterisk 10 Application_PauseMonitor**

**Asterisk 10 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

```plaintext
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 10 Application_Queue
Asterisk 10 Application_QueueLog
Asterisk 10 Application_AddQueueMember
Asterisk 10 Application_RemoveQueueMember
Asterisk 10 Application_PauseQueueMember
Asterisk 10 Application_UnpauseQueueMember
Asterisk 10 Function_QUEUE_VARIABLES
Asterisk 10 Function_QUEUE_MEMBER
Asterisk 10 Function_QUEUE_MEMBER_COUNT
Asterisk 10 Function_QUEUE_EXISTS
Asterisk 10 Function_QUEUE_WAITING_COUNT
Asterisk 10 FunctionQUEUE_MEMBER_LIST
Asterisk 10 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_UserEvent

UserEvent()

Synopsis

Send an arbitrary event to the manager interface.

Description

Sends an arbitrary event to the manager interface, with an optional body representing additional arguments. The body may be specified as a , delimited list of headers. Each additional argument will be placed on a new line in the event. The format of the event will be:

Event: UserEvent

UserEvent: <specified event name>

body

If no body is specified, only Event and UserEvent headers will be present.

Syntax

UserEvent(eventname[,body])

Arguments

- eventname
- body
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.
Asterisk 10 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:

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

- **mailboxes**
  - `mailbox1`
  - `mailbox2`
- **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 10 Application_VoiceMailMain

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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).
  - # - 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
    - 0 - INBOX
    - 1 - Old
    - 2 - Work
    - 3 - Family
    - 4 - Friends
    - 5 - Cust1
    - 6 - Cust2
    - 7 - Cust3
    - 8 - Cust4
    - 9 - Cust5

See Also
Asterisk 10 Application_VoiceMail

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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.

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.

See Also

Asterisk 10 Application_BackGround
Asterisk 10 Function_TIMEOUT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Application_WaitForSilence

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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-10-r340810.

Asterisk 10 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 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 10 Application_WaitForNoise

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_WaitMusicOnHold

WaitMusicOnHold()

Synopsis

Wait, playing Music On Hold.

Description

!!! DEPRECATED. Use MusicOnHold instead !!!

Plays hold music specified number of seconds. Returns 0 when done, or −1 on hangup. If no hold music is available, the delay will still occur with no sound.

!!! DEPRECATED. Use MusicOnHold instead !!!

Syntax
WaitMusicOnHold(delay)

Arguments
- delay

See Also
Import Version
This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Application_WaitUntil
WaitUntil()

Synopsis
Wait (sleep) until the current time is the given epoch.

Description
Waits until the given epoch.

Sets
Sets None - 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-10-r340810.

Asterisk 10 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 10 Application_EndWhile
Asterisk 10 Application_ExitWhile
Asterisk 10 Application_ContinueWhile

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

Asterisk 10 Dialplan Functions

Asterisk 10 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

```plaintext
AES_DECRYPT(key,string)
```

Arguments

• key - AES Key
• string - Input string.

See Also

Asterisk 10 Function_AES_ENCRYPT
Asterisk 10 Function_BASE64_ENCODE
Asterisk 10 Function_BASE64_DECODE

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 10 Function_AES_DECRYPT
- Asterisk 10 Function_BASE64_ENCODE
- Asterisk 10 Function_BASE64_DECODE

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Function_AGK**

**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

- \text{channeldirection} \ - \text{This can be either rx or tx}

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

\textbf{Asterisk 10 FunctionAGENT}

\textbf{AGENT()}

Synopsis

Gets information about an Agent

Description

Syntax

AGENT(agentid[,item])

Arguments

- \text{agentid}
- \text{item} - The valid items to retrieve are:
  - \text{status} - (default) The status of the agent (LOGGEDIN | LOGGEDOUT)
  - \text{password} - The password of the agent
  - \text{name} - The name of the agent
  - \text{mohclass} - MusicOnHold class
  - \text{channel} - The name of the active channel for the Agent (AgentLogin)
  - \text{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-10-r340810.

\textbf{Asterisk 10 FunctionARRAY}

\textbf{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-10-r340810.

**Asterisk 10 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-10-r340810.
Asterisk 10 Function_AUDIOHOOK_INHERIT

AUDIOHOOK_INHERIT()

Synopsis

Set whether an audiohook may be inherited to another channel

Description

By enabling audiohook inheritance on the channel, you are giving permission for an audiohook to be inherited by a descendent channel. Inheritance may be disabled at any point as well.

Example scenario:

exten => 2000,1,MixMonitor( blah.wav)

exten => 2000,n,Set(AUDIOHOOK_INHERIT(MixMonitor)=yes)

exten => 2000,n,Dial(SIP/2000)

exten => 4000,1,Dial(SIP/4000)

exten => 5000,1,MixMonitor( blah2.wav)

exten => 5000,n,Dial(SIP/5000)

In this basic dialplan scenario, let's consider the following sample calls

Call 1: Caller dials 2000. The person who answers then executes an attended transfer to 4000.

Result: Since extension 2000 set MixMonitor to be inheritable, after the transfer to 4000 has completed, the call will continue to be recorded to blah.wav

Call 2: Caller dials 5000. The person who answers then executes an attended transfer to 4000.

Result: Since extension 5000 did not set MixMonitor to be inheritable, the recording will stop once the call has been transferred to 4000.

Syntax

| AUDIOHOOK_INHERIT(source) |

Arguments
- source - The built-in sources in Asterisk are: Note that the names are not case-sensitive
  - MixMonitor
  - Chanspy
  - Volume
  - Speex
  - JACK_HOOK

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Function_BASE64_ENCODE
Asterisk 10 Function_AES_DECRYPT
Asterisk 10 Function_AES_ENCRYPT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Function_BASE64_DECODE
- Asterisk 10 Function_AES_DECRYPT
- Asterisk 10 Function_AES_ENCRYPT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Function_DB

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Function_CALAENDAR_BUSY**
**CALENDAR_BUSY()**

**Synopsis**

Determine if the calendar is marked busy at this time.

**Description**

Check the specified calendar's current busy status.

**Syntax**

```plaintext
CALENDAR_BUSY(calendar)
```

**Arguments**

- `calendar`

**See Also**

Asterisk 10 Function_CALENDAR_EVENT  
Asterisk 10 Function_CALENDAR_QUERY  
Asterisk 10 Function_CALENDAR_QUERY_RESULT  
Asterisk 10 Function_CALENDAR_WRITE

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**

```plaintext
CALENDAR_EVENT(field)
```

**Arguments**

- `field`
Asterisk 10 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

```plaintext
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 10 Function_CALENDAR_BUSY
Asterisk 10 Function_CALENDAR.Event
Asterisk 10 Function_CALENDAR.Query.Result
Asterisk 10 Function_CALENDAR.Write

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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

```plaintext
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 associated 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 10 Function_CALENDAR_BUSY
Asterisk 10 Function_CALENDAR_EVENT
Asterisk 10 Function_CALENDAR_QUERY
Asterisk 10 Function_CALENDAR_WRITE

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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

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 10 Function_CALENDAR_BUSY
Asterisk 10 Function_CALENDAR_EVENT
Asterisk 10 Function_CALENDAR_QUERY
Asterisk 10 Function_CALENDAR_QUERY_RESULT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 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-10-r340810.

**Asterisk 10 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
- ISO8859-1
- Withdrawn
- ISO8859-2
- ISO8859-3
- ISO8859-4
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
  - 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

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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:

- Presentation Allowed, Not Screened.
- Presentation Allowed, Passed Screen.
- Presentation Allowed, Failed Screen.
- Presentation Allowed, Network Number.
- Presentation Prohibited, Not Screened.
- Presentation Prohibited, Passed Screen.
- Presentation Prohibited, Failed Screen.
- Presentation Prohibited, Network Number.
- Number Unavailable.

Syntax

CALLERPRES()

Arguments

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.
For setting CDR values, the flag does not apply to setting the accountcode, userfield, or amaflags.

Raw values for disposition:

NO ANSWER
NO ANSWER (NULL record)
FAILED
BUSY
ANSWERED

Raw values for amaflags:

OMIT
BILLING
DOCUMENTATION

Example: exten => 1,1,Set(CDR(userfield)=test)

Syntax

```plaintext
CDR(name[,options])
```

Arguments

- **name**: CDR field name:
  - clid: Caller ID.
  - lastdata: Last application arguments.
  - disposition: ANSWERED, NO ANSWER, BUSY, FAILED.
  - src: Source.
  - start: Time the call started.
  - amaflags: DOCUMENTATION, BILL, IGNORE, etc.
  - dst: Destination.
  - answer: Time the call was answered.
  - accountcode: The channel's account code.
  - dcontext: Destination context.
  - end: Time the call ended.
  - uniqueld: The channel's unique id.
  - dstchannel: Destination channel.
  - duration: Duration of the call.
  - userfield: The channel's user specified field.
  - lastapp: Last application.
  - billsec: Duration of the call once it was answered.
  - channel: Channel name.
  - sequence: CDR sequence number.

- **options**
  - f: Returns billsec or duration fields as floating point values.
  - l: Uses the most recent CDR on a channel with multiple records
  - r: Searches the entire stack of CDRs on the channel.
  - s: Skips any CDR's that are marked 'LOCKED' due to forkCDR() calls. (on setting/writing CDR vars only)
  - u: Retrieves the raw, unprocessed value. For example, 'start', 'answer', and 'end' will be retrieved as epoch values, when the u
option is passed, but formatted as YYYY-MM-DD HH:MM:SS otherwise. Similarly, disposition and amaflags will return their raw integral values.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 FunctionCHANNEL

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: chan_sip provides the following additional options: chan_iax2 provides the following additional options: chan_dahdi provides the following additional options: chan_ooh323 provides the following additional options:
  - audioreadformat - R/O format currently being read.
  - audionativeformat - R/O format used natively for audio.
  - audiowriteformat - R/O format currently being written.
  - callgroup - R/W call groups for call pickup.
  - pickupgroup - R/W call groups for call pickup.
  - channeltype - R/O technology used for channel.
  - checkhangup - R/O Whether the channel is hanging up (1/0)
  - 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.
  - txgain - 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.
  - peerip - R/O Get the IP address of the peer.
  - recvip - R/O Get the source IP address of the peer.
• from - R/O Get the URI from the From: header.
• uri - R/O Get the URI from the Contact: header.
• useragent - R/O Get the useragent.
• peername - R/O Get the name of the peer.
• t38passthrough - R/O 1 if T38 is offered or enabled in this channel, otherwise 0
• rtpqos - R/O Get QOS information about the RTP stream This option takes two additional arguments: Argument 1:
  audio Get data about the audio stream video Get data about the video stream text Get data about the text stream Argument 2:
  local_ssrc Local SSRc (stream ID) local_lostpackets Local lost packets local_jitter Local calculated jitter
  local_maxjitter Local calculated jitter (maximum) local_minjitter Local calculated jitter (minimum)
  local_normdevjitter Local calculated jitter (normal deviation) local_stddevjitter 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_stddevjitter 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)
  stddevrtt 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
• 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.
• faxdetect - Fax Detect [R/W] Returns 0 or 1 Write yes or no
• t38support - t38support [R/W] Returns 0 or 1 Write yes or no
• h323id - Returns h323id R

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

Asterisk 10 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 \texttt{domain= configuration} in \texttt{sip.conf}.

Syntax

```
CHECKSIPDOMAIN(domain)
```

Arguments

- \texttt{domain}

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function CONFBRIDGE

CONFBRIDGE()

Synopsis

Set a custom dynamic bridge and user profile on a channel for the ConfBridge application using the same options defined in confbridge.conf.

Description

```--- Example 1 ----```
In this example the custom set user profile on this channel will automatically be used by the ConfBridge app.

```plaintext
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.

```plaintext
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

```plaintext
CONFBRIDGE(type,option)
```

Arguments

- `type` - Type refers to which type of profile the option belongs too. Type can be `bridge` or `user`.
- `option` - Option refers to `confbridge.conf` option that is being set dynamically on this channel.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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

```plaintext
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-10-r340810.

**Asterisk 10 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

ISO8859-1

Withdrawn

ISO8859-2

ISO8859-3

ISO8859-4

ISO8859-5

ISO8859-7
ISO10646 Bmp String

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`
- `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-10-r340810.

Asterisk 10 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
**Asterisk 10 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 10 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)

**httperror** - 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.

**sslverifypeer** - 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. + to the space character, in violation of current RFC standards.
  - yes
  - no
  - legacy - Also translate + to the space character, in violation of current RFC standards.

---

**See Also**

Asterisk 10 Function CURL
Asterisk 10 Function HASH

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**

```plaintext
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 `,`)

---

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
**Asterisk 10 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 10 Application_DBdel
Asterisk 10 Function_DB_DELETE
Asterisk 10 Application_DBdeltree
Asterisk 10 Function_DB_EXISTS

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

---

**Asterisk 10 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. This function will retrieve a value from the Asterisk database and then remove that key from the database. `None - DB_RESULT` will be set to the key's value if it exists.

Syntax

```plaintext
DB_DELETE(family,key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 10 Application_DBdel
- Asterisk 10 Function_DB
- Asterisk 10 Application_DBdeltree

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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

```plaintext
DB_EXISTS(family,key)
```

Arguments

- `family`
- `key`

See Also

- Asterisk 10 Function_DB
**Asterisk 10 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**

```plaintext
DB_KEYS([prefix])
```

**Arguments**

- **prefix**

**See Also**

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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)` - Increments MyVar

**Note:** `DEC(${MyVAR})` - Is wrong, as INC expects the variable name, not its value

**Syntax**

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
Asterisk 10 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-10-r340810.
**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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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

```markdown
DIALPLAN_EXISTS(context[,extension[,priority]])
```

Arguments

- `context`
- `extension`
- `priority`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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

```markdown
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-10-r340810.

**Asterisk 10 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-10-r340810.

Asterisk 10 Function_DUNDIRESULT

DUNDIRESULT()

Synopsis

Retrieve results from a DUNDIQUERY.

Description

This function will retrieve results from a previous use 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-10-r340810.

Asterisk 10 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.
  - 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Function_ENV**

**ENV()**
Synopsis

Gets or sets the environment variable specified.

Description

Variables starting with \texttt{AST$_\_$_} are reserved to the system and may not be set.

Syntax

\begin{verbatim}
ENV(varname)
\end{verbatim}

Arguments

- \texttt{varname} - Environment variable name

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

\textbf{Asterisk 10 Function \texttt{EVAL}}

\textbf{EVAL()}

Synopsis

Evaluate stored variables

Description

Using \texttt{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 \texttt{EVAL} will have it evaluated a second time.

Example: If the Example: If the \texttt{None - MYVAR contains Example: If the None - OTHERVAR, then the result of${EVAL( Example: If the None - MYVAR })$ in the dialplan will be the contents of Example: If the Example: If the \texttt{None - OTHERVAR. Normally just putting Example: If the None - MYVAR in the dialplan the result would be Example: If the None - OTHERVAR.

Syntax

\begin{verbatim}
EVAL(variable)
\end{verbatim}

Arguments

- \texttt{variable}

See Also
Asterisk 10 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 10 Application_RaiseException

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function_EXISTS

EXISTS()

Synopsis

Test the existence of a value.

Description

Returns 1 if exists, 0 otherwise.

Syntax
Asterisk 10 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-10-r340810.
Asterisk 10 Function_FAXOPT

FAXOPT()

Synopsis

Gets/sets various pieces of information about a fax session.

Description

FAXOPT can be used to override the settings for a FAX session listed in `res_fax.conf`, it can also be used to retrieve 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)
  - `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 10 Application_ReceiveFax
Asterisk 10 Application_SendFax

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 occurred, 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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,10,10)})

; reads from the 11th to 20th byte in the file (i.e. skip the first 10, then read 10 bytes).

Set(foo=${FILE(/tmp/test.txt,10,10)})

Read mode (line):

; reads the 3rd line of the file.

Set(foo=${FILE(/tmp/test.txt,3,1,l)})

; reads the 3rd and 4th lines of the file.
Set(foo=${FILE(/tmp/test.txt,3,2,l)})
; reads from the third line to the end of the file.

Set(foo=${FILE(/tmp/test.txt,3,,l)})
; reads the last three lines of the file.

Set(foo=${FILE(/tmp/test.txt,-3,,l)})
; reads the 3rd line of a DOS-formatted file.

Set(foo=${FILE(/tmp/test.txt,3,1,l,d)})

Write mode (byte):
; truncate the file and write "bar"
Set(FILE(/tmp/test.txt)=bar)
; Append "bar"
Set(FILE(/tmp/test.txt,,a)=bar)
; Replace the first byte with "bar" (replaces 1 character with 3)
Set(FILE(/tmp/test.txt,0,1)=bar)
; Replace 10 bytes beginning at the 21st byte of the file with "bar"
Set(FILE(/tmp/test.txt,20,10)=bar)
; Replace all bytes from the 21st with "bar"
Set(FILE(/tmp/test.txt,20)=bar)
; Insert "bar" after the 4th character
Set(FILE(/tmp/test.txt,4,0)=bar)

Write mode (line):
; Replace the first line of the file with "bar"
Set(FILE(/tmp/foo.txt,0,1,l)=bar)
; Replace the last line of the file with "bar"
Set(FILE(/tmp/foo.txt,-1,,l)=bar)
; Append "bar" to the file with a newline

Set(FILE(/tmp/foo.txt,,,al)=bar)

Syntax

```text
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 10 Function_FILE_COUNT_LINE
Asterisk 10 Function_FILE_FORMAT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function_FILE_COUNT_LINE

FILE_COUNT_LINE()

Synopsis

Obtains the number of lines of a text file.

Description

Returns the number of lines, or −1 on error.

Syntax

```text
FILE_COUNT_LINE(filename[,format])
```

Arguments

- **filename**
- **format** - Format may be one of the following: If not specified, an attempt will be made to determine the newline format type. If not specified, an attempt will be made to determine the newline format type.
  - `u` - Unix newline format.
See Also

Asterisk 10 Function_FILE
Asterisk 10 Function_FILE_FORMAT

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function_FILE_FORMAT

FILE_FORMAT()

Synopsis

Return the newline format of a text file.

Description

Return the line terminator type:

'\u' - Unix "\n" format
'd' - DOS "\r\n" format
'm' - Macintosh "\r" format
'x' - Cannot be determined

Syntax

FILE_FORMAT(filename)

Arguments

- filename

See Also

Asterisk 10 Function_FILE
Asterisk 10 Function_FILE_COUNT_LINE

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
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.

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-10-r340810.

Asterisk 10 FunctionFRAME_TRACE

FRAME_TRACE()

Synopsis

View internal ast_frames as they are read and written on a channel.

Description

Examples:

exen => 1,1,Set(FRAME_TRACE(white)=DTMF_BEGIN,DTMF_END); view only DTMF frames.
exen => 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
  - FAX
  - TEXT
  - IMAGE
  - HTML
  - CNG
  - MODEM

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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
Asterisk 10 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-10-r340810.

Asterisk 10 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 specified (and non-empty).

Syntax

GROUP_COUNT([groupname[,category]])

Arguments

- groupname - Group name.
- category - Category name.
Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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[@context][,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-10-r340810.

**Asterisk 10 Function_IAXPEER**
IAXPEER()

Synopsis

Gets IAX peer information.

Description

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 10 Function_SIPPEER

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

IAXVAR()

Synopsis

Sets or retrieves a remote variable.

Description

Syntax

IAXVAR(varname)

Arguments

- varname
Asterisk 10 Function_ICONV

ICONV()

Synopsis

Converts charsets of strings.

Description

Converts string from \textit{in-charset} into \textit{out-charset}. For available charsets, use \texttt{iconv -l} on your shell command line.

Due to limitations within the API, ICONV will not currently work with charsets with embedded NULLs. If found, the string will terminate.

Syntax

\texttt{ICONV(in-charset,out-charset,string)}

Arguments

- in-charset - Input charset
- out-charset - Output charset
- string - String to convert, from \textit{in-charset} to \textit{out-charset}

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Returns the data following ? if true, else the data following :

Syntax

```
IF(expresion?[true][:false])
```

Arguments

- `expresion`:
- `retval`
  - `true`
  - `false`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 Function_IFTIME**

**IFTIME()**
Synopsis
Temporal Conditional.

Description
Returns the data following ? if true, else the data following :.

Syntax

\[
\text{IFTIME}(\text{timespec?true}:false)
\]

Arguments
- timespec
- retvalue
  - true
  - false

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Function IMPORT**

**IMPORT()**

Synopsis
Retrieve the value of a variable from another channel.

Description

Syntax

\[
\text{IMPORT(channel,variable)}
\]

Arguments
- channel
- variable

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

Asterisk 10 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
Asterisk 10 Function JABBER_RECEIVE

JABBER_RECEIVE()

Synopsis

Reads XMPP messages.

Description

Receives a text message on the given account from the buddy identified by jid and returns the contents.

Example: ${JABBER_RECEIVE(asterisk,bob@domain.com)} returns an XMPP message sent from bob@domain.com (or nothing in case of a time out), to the asterisk XMPP account configured in jabber.conf.

Syntax

JABBER_RECEIVE(account, jid[, timeout])

Arguments

- account - The local named account to listen on (specified in jabber.conf)
- jid - Jabber ID of the buddy to receive message from. It can be a bare JID (username@domain) or a full JID (username@domain/resource).
- timeout - In seconds, defaults to 20.

See Also

Asterisk 10 Function JABBER_STATUS
Asterisk 10 Application JabberSend

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function JABBER_STATUS

JABBER_STATUS()

Synopsis

Retrieves a buddy's status.

Description
Retrieves the numeric status associated with the buddy identified by *jid*. If the buddy does not exist in the buddylist, returns 7.

Status will be 1-7.

1=Online, 2=Chatty, 3=Away, 4=XAway, 5=DND, 6=Offline

If not in roster variable will be set to 7.

Example: `${JABBER_STATUS(asterisk,bob@domain.com)}` returns 1 if `bob@domain.com` is online. *asterisk* is the associated XMPP account configured in jabber.conf.

**Syntax**

```
JABBER_STATUS(account,jid)
```

**Arguments**

- **account** - The local named account to listen on (specified in jabber.conf)
- **jid** - Jabber ID of the buddy to receive message from. It can be a bare JID (username@domain) or a full JID (username@domain/resource).

**See Also**

Asterisk 10 Function_JABBER_RECEIVE
Asterisk 10 Application_JabberSend

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Function_JITTERBUFFER**

**JITTERBUFFER()**

**Synopsis**

Add a Jitterbuffer to the Read side of the channel. This dejitters the audio stream before it reaches the Asterisk core. This is a write only function.

**Description**

- **max_size**: Defaults to 200 ms
  
  Length in milliseconds of buffer.

- **resync_threshold**: Defaults to 1000ms
  
  The length in milliseconds over which a timestamp difference will result in resyncing the jitterbuffer.
target_extra: Defaults to 40ms

This option only affects the adaptive jitterbuffer. It represents the amount time in milliseconds by which the new jitter buffer will pad its size.

Examples:

exten => 1,1,Set(JITTERBUFFER(fixed)=default);Fixed with defaults.

exten => 1,1,Set(JITTERBUFFER(fixed)=200);Fixed with max size 200ms, default resync threshold and target extra.

exten => 1,1,Set(JITTERBUFFER(fixed)=200,1500);Fixed with max size 200ms resync threshold 1500.

exten => 1,1,Set(JITTERBUFFER(adaptive)=default);Adaptive with defaults.

exten => 1,1,Set(JITTERBUFFER(adaptive)=200,60);Adaptive with max size 200ms, default resync threshold and 40ms target extra.

Syntax

```plaintext
JITTERBUFFER(jitterbuffer type)
```

Arguments

- `jitterbuffer type` - Jitterbuffer type can be either fixed or adaptive. Used as follows.
  - Set(JITTERBUFFER(type)=max_size[,resync_threshold,target_extra])
  - Set(JITTERBUFFER(type)=default)

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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

```plaintext
LISTFILTER(varname, delim, value)
```

Arguments

- `varname`
- `delim`
- `value`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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

```plaintext
LOCAL(varname)
```

Arguments

- `varname`

See Also

- Asterisk 10 Application_Gosub
- Asterisk 10 Application_GosubIf
- Asterisk 10 Application_Return

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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 Read a variable None - 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 10 Application_Gosub
Asterisk 10 Application_GosubIf
Asterisk 10 Application_Return

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.

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. To avoid the possibility of a deadlock, LOCK will only
attempt to obtain the lock for 3 seconds if the channel already has another lock.

**Syntax**

```
LOCK(lockname)
```

**Arguments**

- `lockname`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 Function_MAILBOX_EXISTS**

**MAILBOX_EXISTS()**

**Synopsis**

Tell if a mailbox is configured.

**Description**

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**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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

```
MASTER_CHANNEL()
```

Arguments

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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: `number1 op number2` 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-10-r340810.
Asterisk 10 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-10-r340810.

Asterisk 10 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 10 Application_MeetMe
Asterisk 10 Application_MeetMeCount
Asterisk 10 Application_MeetMeAdmin
Asterisk 10 Application_MeetMeChannelAdmin

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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.
- **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 10 Application_MessageSend

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function_MESSAGE_DATA

MESSAGE_DATA()

Synopsis

Content is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License.
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 10 Application_MessageSend

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.
### 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 10 Application_MinivmRecord
- Asterisk 10 Application_MinivmGreet
- Asterisk 10 Application_MinivmNotify
- Asterisk 10 Application_MinivmDelete
- Asterisk 10 Application_MinivmAccMess
- Asterisk 10 Function_MINIVMCOUNTER

---

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 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.

Example:

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-10-r340810.

Asterisk 10 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` - 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.
- `transaction` - 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.
- `forcecommit` - 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-10-r340810.

**Asterisk 10 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**

**None** - `ODBC_FETCH_STATUS`.

- `ODBC_FETCH_STATUS`
  - `SUCCESS` - If rows are available.
  - `FAILURE` - If no rows are available.

**Syntax**

```
ODBC_FETCH(result-id)
```

**Arguments**

- `result-id`

**See Also**
Asterisk 10 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.

Syntax

```
PASSTHRU([string])
```

Arguments

- string

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

Asterisk 10 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-10-r340810.
Asterisk 10 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

```plaintext
QUEUE_EXISTS([queueName])
```

Arguments

- `queueName`

See Also

Asterisk 10 Application_Queue
Asterisk 10 Application_QueueLog
Asterisk 10 Application_AddQueueMember
Asterisk 10 Application_RemoveQueueMember
Asterisk 10 Application_PauseQueueMember
Asterisk 10 Application_UnpauseQueueMember
Asterisk 10 Function_QUEUE_VARIABLES
Asterisk 10 Function_QUEUE_MEMBER
Asterisk 10 Function_QUEUE_MEMBER_COUNT
Asterisk 10 Function_QUEUE_EXISTS
Asterisk 10 Function_QUEUE_WAITING_COUNT
Asterisk 10 Function_QUEUE_MEMBER_LIST
Asterisk 10 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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].

*queue name* 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.
  - `ignorebusy` - Gets or sets queue member ignorebusy.
- `interface`

**See Also**

Asterisk 10 Application_Queue
Asterisk 10 Application_QueueLog
Asterisk 10 Application_AddQueueMember
Asterisk 10 Application_RemoveQueueMember
Asterisk 10 Application_PauseQueueMember
Asterisk 10 Application_UnpauseQueueMember
Asterisk 10 Function_QUEUE_VARIABLES
Asterisk 10 Function_QUEUE_MEMBER
Asterisk 10 Function_QUEUE_MEMBER_COUNT
Asterisk 10 Function_QUEUE_EXISTS
Asterisk 10 Function.Queue_WAITING_COUNT
Asterisk 10 Function_QUEUE_MEMBER_LIST
Asterisk 10 Function_QUEUE_MEMBER_PENALTY

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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`.

This function has been deprecated in favor of the `QUEUE_MEMBER()` function.

### Syntax

```
QUEUE_MEMBER_COUNT(queueName)
```

### Arguments

- `queueName`

### See Also

- Asterisk 10 Application_Queue
- Asterisk 10 Application_QueueLog
- Asterisk 10 Application_AddQueueMember
- Asterisk 10 Application_RemoveQueueMember
- Asterisk 10 Application_PauseQueueMember
- Asterisk 10 Application_UnpauseQueueMember
- Asterisk 10 Function_QUEUE_VARIABLES
- Asterisk 10 Function_QUEUE_MEMBER
- Asterisk 10 Function_QUEUE_MEMBER_COUNT
- Asterisk 10 Function_QUEUE_EXISTS
- Asterisk 10 Function_QUEUE_WAITING_COUNT
- Asterisk 10 Function_QUEUE_MEMBER_LIST
- Asterisk 10 Function_QUEUE_MEMBER_PENALTY

### Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

---

### Asterisk 10 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`
Asterisk 10 Function_QUEUE_MEMBER_PENALTY

QUEUE_MEMBER_PENALTY()

Synopsis

Gets or sets queue members penalty.

Description

Gets or sets queue members penalty.

This function has been deprecated in favor of the QUEUE_MEMBER() function

Syntax

```
QUEUE_MEMBER_PENALTY(queuename,interface)
```

Arguments

- `queuename`
- `interface`

See Also

Asterisk 10 Application_Queue
Asterisk 10 Application_QueueLog
Asterisk 10 Application_AddQueueMember
Asterisk 10 Application_RemoveQueueMember
Asterisk 10 Application_PauseQueueMember
Asterisk 10 Application_UnpauseQueueMember
Asterisk 10 Function_QUEUE_VARIABLES
Asterisk 10 Function_QUEUE_MEMBER
Asterisk 10 Function_QUEUE_MEMBER_COUNT
Asterisk 10 Function_QUEUE_EXISTS
Asterisk 10 Function_QUEUE_WAITING_COUNT
Asterisk 10 Function_QUEUE_MEMBER_LIST
Asterisk 10 Function_QUEUE_MEMBER_PENALTY
Asterisk 10 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 - Maximum 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 10 Application_Queue  
Asterisk 10 Application_QueueLog  
Asterisk 10 Application_AddQueueMember  
Asterisk 10 Application_RemoveQueueMember  
Asterisk 10 Application_PauseQueueMember
Asterisk 10 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 *queue name*.

**Syntax**

```
QUEUE_WAITING_COUNT([queue name])
```

**Arguments**

- `queue name`

**See Also**

Asterisk 10 Application_Queue
Asterisk 10 Application_QueueLog
Asterisk 10 Application_AddQueueMember
Asterisk 10 Application_RemoveQueueMember
Asterisk 10 Application_PauseQueueMember
Asterisk 10 Application_UnpauseQueueMember
Asterisk 10 Function_QUEUE_VARIABLES
Asterisk 10 Function_QUEUE_MEMBER
Asterisk 10 Function_QUEUE_MEMBER_COUNT
Asterisk 10 Function_QUEUE_EXISTS
Asterisk 10 Function_QUEUE_WAITING_COUNT
Asterisk 10 Function_QUEUE_MEMBER_LIST
Asterisk 10 Function_QUEUE_MEMBER_PENALTY

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
Asterisk 10 Function _QUOTE

QUOTE()

Synopsis

Quotes a given string, escaping embedded quotes as necessary

Description

Example: \$\{QUOTE(ab"c"de)} will return "abcde"

Syntax

QUOTE(string)

Arguments

- string

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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
Asterisk 10 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[,value[,delim1|field[,delim2]]])

Arguments

- family
- fieldmatch
- value
- 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 10 Function_REALTIME_STORE
Asterisk 10 Function_REALTIME_DESTROY
Asterisk 10 Function_REALTIME_FIELD
Asterisk 10 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.
REALTIME_DESTROY()

Synopsis

RealTime Destroy Function.

Description

This function acts in the same way as REALTIME(....) does, except that it destroys the matched record in the RT engine.

Syntax

```
REALTIME_DESTROY(family,fieldmatch[,value[,delim1[,delim2]]])
```

Arguments

- family
- fieldmatch
- value
- delim1
- delim2

See Also

Asterisk 10 Function_REALTIME
Asterisk 10 Function_REALTIME_STORE
Asterisk 10 Function_REALTIME_FIELD
Asterisk 10 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 `value`. When written to, the REALTIME_FIELD() function performs identically to the REALTIME() function.

Syntax
REALTIME_HASH(family, fieldmatch, value)

Arguments

- family
- fieldmatch
- value

See Also

Asterisk 10 Function_REALTIME
Asterisk 10 Function_REALTIME_STORE
Asterisk 10 Function_REALTIME_DESTROY
Asterisk 10 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 value 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, value)
```

Arguments

- family
- fieldmatch
- value

See Also

Asterisk 10 Function_REALTIME
Asterisk 10 Function_REALTIME_STORE
Asterisk 10 Function_REALTIME_DESTROY
Asterisk 10 Function_REALTIME_FIELD
Asterisk 10 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 10 Function_REALTIME
Asterisk 10 Function_REALTIME_DESTROY
Asterisk 10 Function_REALTIME_FIELD
Asterisk 10 Function_REALTIME_HASH

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 field are the following:

Unknown
Call Forwarding Busy
Call Forwarding No Reply
Callee is Unavailable
Time of Day
Do Not Disturb
Call Deflection
Follow Me
Called DTE Out-Of-Order
Callee is Away
Call Forwarding By The Called DTE
Call Forwarding Unconditional

The allowable values for the xxx-name-charset field are the following:

Unknown
ISO8859-1
Withdrawn
ISO8859-2
ISO8859-3
ISO8859-4
ISO8859-5
ISO8859-7
ISO10646 Bmp String
ISO10646 UTF-8 String
### Syntax

```
REDIRECTING(datatype[,i])
```

### Arguments

- **datatype** - The allowable datatypes are:
  - `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`
  - `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-10-r340810.

### Asterisk 10 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-10-r340810.

**Asterisk 10 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.

The replacement only occurs in the output. The original variable is not altered. 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-10-r340810.

**Asterisk 10 Function_SET**

**SET()**
Synopsis

SET assigns a value to a channel variable.

Description

Syntax

```plaintext
SET(varname[,value])
```

Arguments

- `varname`
- `value`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 Function_SHA1

SHA1()

Synopsis

Computes a SHA1 digest.

Description

Generate a SHA1 digest via the SHA1 algorithm.

Example: Set(sha1hash=${SHA1(junky)})

Sets the asterisk variable `sha1hash` to the string `60fa5675b9303eb62f99a9cd47f9f5837d18f9a0` which is known as his hash

Syntax

```plaintext
SHA1(data)
```

Arguments

- `data` - Input string

See Also

Import Version
Asterisk 10 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 The variables used in this space are separate from the general namespace of the channel and thus None - SHARED(foo) and The variables used in this space are separate from the general namespace of the channel and thus None - 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-10-r340810.

Asterisk 10 Function_SHELL

SHELL()

Synopsis

Executes a command as if you were at a shell.
**Description**

Returns the value from a system command

**Example:**

Set(foo=${SHELL(echo \bar)})

When using the SHELL() dialplan function, your \SHELL\ is /bin/sh, which may differ as to the underlying shell, depending upon your production platform. Also keep in mind that if you are using a common path, you should be mindful of race conditions that could result from two calls running SHELL() simultaneously. When using the SHELL() dialplan function, your \SHELL\ is /bin/sh, which may differ as to the underlying shell, depending upon your production platform. Also keep in mind that if you are using a common path, you should be mindful of race conditions that could result from two calls running SHELL() simultaneously.

**Syntax**

```
SHELL(command)
```

**Arguments**

- **command** - This is the argument to the function, the command you want to pass to the shell.

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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**

```plaintext
SHIFT(varname[,delimiter])
```

**Arguments**

- `varname`
- `delimiter`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.

**Syntax**

```plaintext
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-10-r340810.

**Asterisk 10 Function_SIPCHANINFO**

**SIPCHANINFO()**
Synopsis

Gets the specified SIP parameter from the current channel.

Description

Syntax

```
SIPCHANINFO(item)
```

Arguments

- `item`
  - `peerip` - The IP address of the peer.
  - `recvip` - The source IP address of the peer.
  - `from` - The URI from the `From` header.
  - `uri` - The URI from the `Contact` header.
  - `useragent` - The useragent.
  - `peername` - The name of the peer.
  - `t38passthrough` - 1 if T38 is offered or enabled in this channel, otherwise 0.

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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.
- codecs - The configured codecs.
- status - Status (if qualify=yes).
- regexten - Registration extension.
- 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 id for 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-10-r340810.

Asterisk 10 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:
    - D - Direct Calls
    - A - Forward All Calls
    - B - Forward Busy Calls
    - N - Forward No Answer Calls

See Also

Asterisk 10 Function_SMDI_MSG_RETRIEVE

Import Version
Asterisk 10 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 10 Function_SMDI_MSG

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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(key1val1[,key2val2[,...]])
```

Arguments

- `keyval`
  - `key1`
  - `val1`
- `keyvaln`
  - `key2`
  - `val2`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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-10-r340810.
Asterisk 10 Function_SPEECHENGINE

SPEECHENGINE()

Synopsis

Change a speech engine specific attribute.

Description

Changes a speech engine specific attribute.

Syntax

```
SPEECHENGINE(name)
```

Arguments

- `name`

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 Function_SPEECHGRAMMAR

SPEECHGRAMMAR()

Synopsis

Gets the matched grammar of a result if available.

Description

Gets the matched grammar of a result if available.

Syntax

```
SPEECHGRAMMAR([nbest_number,result_number])
```

Arguments

- `nbest_number`
- `result_number`

See Also
Asterisk 10 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

Asterisk 10 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
Asterisk 10 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-10-r340810.

Asterisk 10 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**
**Asterisk 10 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**

`sprintf(3)`

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

---

**Asterisk 10 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-10-r340810.

---

**Asterisk 10 Function_STAT**

**STAT()**
Synopsis

Does a check on the specified file.

Description

Syntax

```
STAT(flag, filename)
```

Arguments

- **flag** - Flag may be one of the following: d - Checks if the file is a directory. e - Checks if the file exists. f - Checks if the file is a regular file. m - Returns the file mode (in octal) s - Returns the size (in bytes) of the file A - Returns the epoch at which the file was last accessed. C - Returns the epoch at which the inode was last changed. M - Returns the epoch at which the file was last modified.

- **filename**

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

Asterisk 10 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 `strptime(3)`. It also supports the following format:

`%nq` - 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
Asterisk 10 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

Asterisk 10 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 effectively delete that substring. If max-replacements is specified, this function will stop after performing replacements max-replacements times.

The replacement only occurs in the output. The original variable is not altered. 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-10-r340810.

**Asterisk 10 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. This parameter is dependant upon operating system. This parameter is dependant upon operating system.
  - totalram - Total usable main memory size in KiB. This parameter is dependant upon operating system. This parameter is dependant upon operating system.
  - freeram - Available memory size in KiB. This parameter is dependant upon operating system. This parameter is dependant upon operating system.
  - bufferram - Memory used by buffers in KiB. This parameter is dependant upon operating system. This parameter is dependant upon operating system.
  - totalswap - Total swap space still available in KiB. This parameter is dependant upon operating system. This parameter is dependant upon operating system.
  - other - Other system information.
See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 10 Application_GotoIfTime**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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 absolute 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 i extension if it exists, and if not the call would be terminated. The default timeout is 10 seconds.

Syntax

```plaintext
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-10-r340810.

**Asterisk 10 Function_TOLOWER**

**TOLOWER()**

Synopsis

Convert string to all lowercase letters.

Description

Example: `${TOLOWER(Example)}` returns "example"

Syntax

```plaintext
TOLOWER(string)
```

Arguments

- `string`
Asterisk 10 Function_TOUPPER

TOUPPER()

Synopsis

Convert string to all uppercase letters.

Description

Example: ${TOUPPER(Example)} returns "EXAMPLE"

Syntax

```
TOUPPER(string)
```

Arguments

- `string`

Asterisk 10 Function_TRYLOCK

TRYLOCK()

Synopsis

Attempt to obtain a named mutex.

Description

Attempts to grab a named lock exclusively, and prevents other channels from obtaining the same lock. Returns 1 if the lock was available or 0 otherwise.

Syntax

```
TRYLOCK(lockname)
```
Arguments

- lockname

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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.
It is generally unnecessary to unlock in a hangup routine, as any locks held are automatically freed when the channel is destroyed. It is generally unnecessary to unlock in a hangup routine, as any locks held are automatically freed when the channel is destroyed.

**Syntax**

```plaintext
UNLOCK(lockname)
```

**Arguments**

- `lockname`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 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**

```plaintext
UNSHIFT(varname[,delimiter])
```

**Arguments**

- `varname`
- `delimiter`

**See Also**

**Import Version**

This documentation was imported from Asterisk version SVN-branch-10-r340810.

### Asterisk 10 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-10-r340810.

**Asterisk 10 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-10-r340810.

**Asterisk 10 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.

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-10-r340810.

Asterisk 10 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 (right now fixed at 999999).
- **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-10-r340810.

**Asterisk 10 Function VMCOUNT**

**VMCOUNT()**

Synopsis

Count the voicemails in a specified mailbox.

Description

Count the number of voicemails in a specified mailbox, you could also specify the `context` and the mailbox `folder`.

Example:

```plaintext
exten => s,1,Set(foo=${VMCOUNT(125)})
```

Syntax

```
VMCOUNT(vmbox[@context][,folder])
```

Arguments

- `vmbox`
- `context` - If not specified, defaults to default.
- `folder` - If not specified, defaults to INBOX

See Also

Import Version

This documentation was imported from Asterisk version SVN-branch-10-r340810.

**Asterisk 10 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-10-r340810.